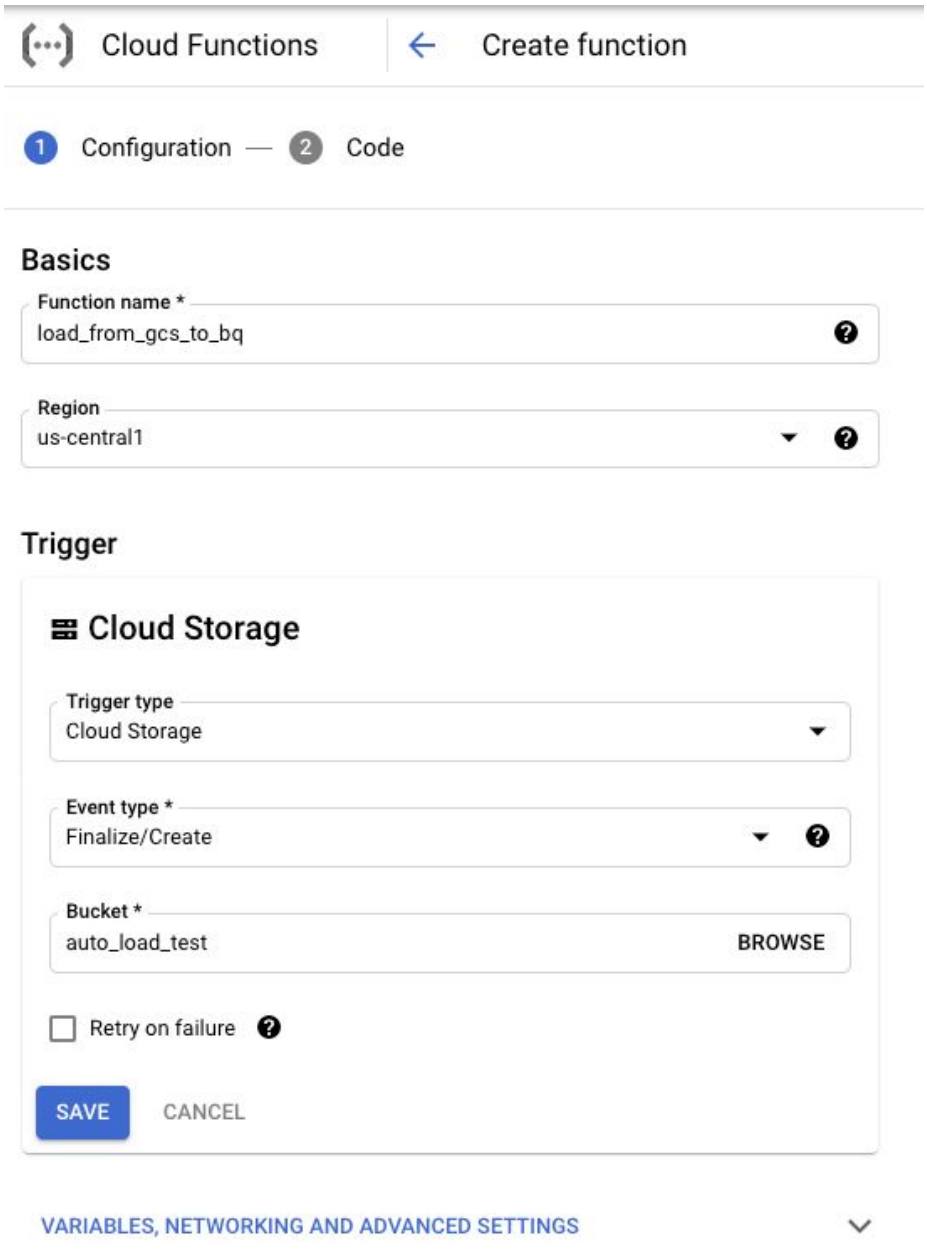
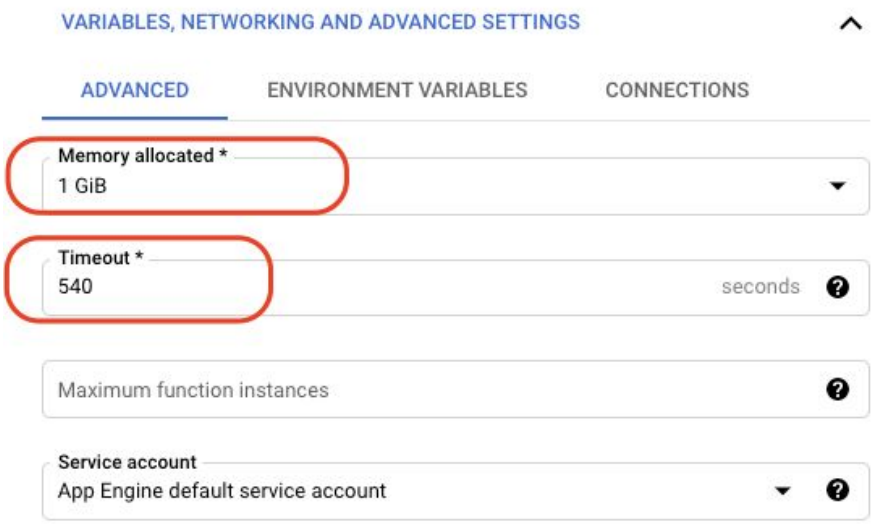
用Cloud Functions导入数据到BigQuery

Author: eugeneyu@google.com

首先在谷歌云控制台创建一个Cloud Functions函数。参照下面截图填写相应配置。其中“Bucket”选择源桶。“Event Type”选择“Finalize/Create”。



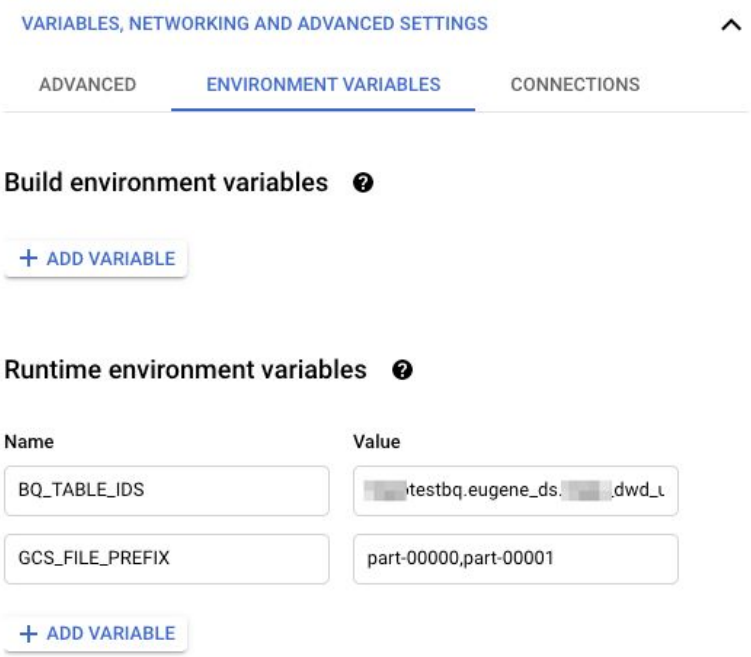
在下面的高级配置中，适当增大Cloud Functions的分配内存和超时设置（最长9分钟）。当导入的文件较大时，配置大一些的内存和超时可以防止出错。



在环境变量配置中创建新的环境变量，指定导入文件的特定前缀和BigQuery的目的表。如果有多个不同前缀文件，要分布导入不同的目的表，可以用相同的顺序列表填写。函数代码会按照列表顺序，将不同前缀文件数据导入到不同的BigQuery目的表。

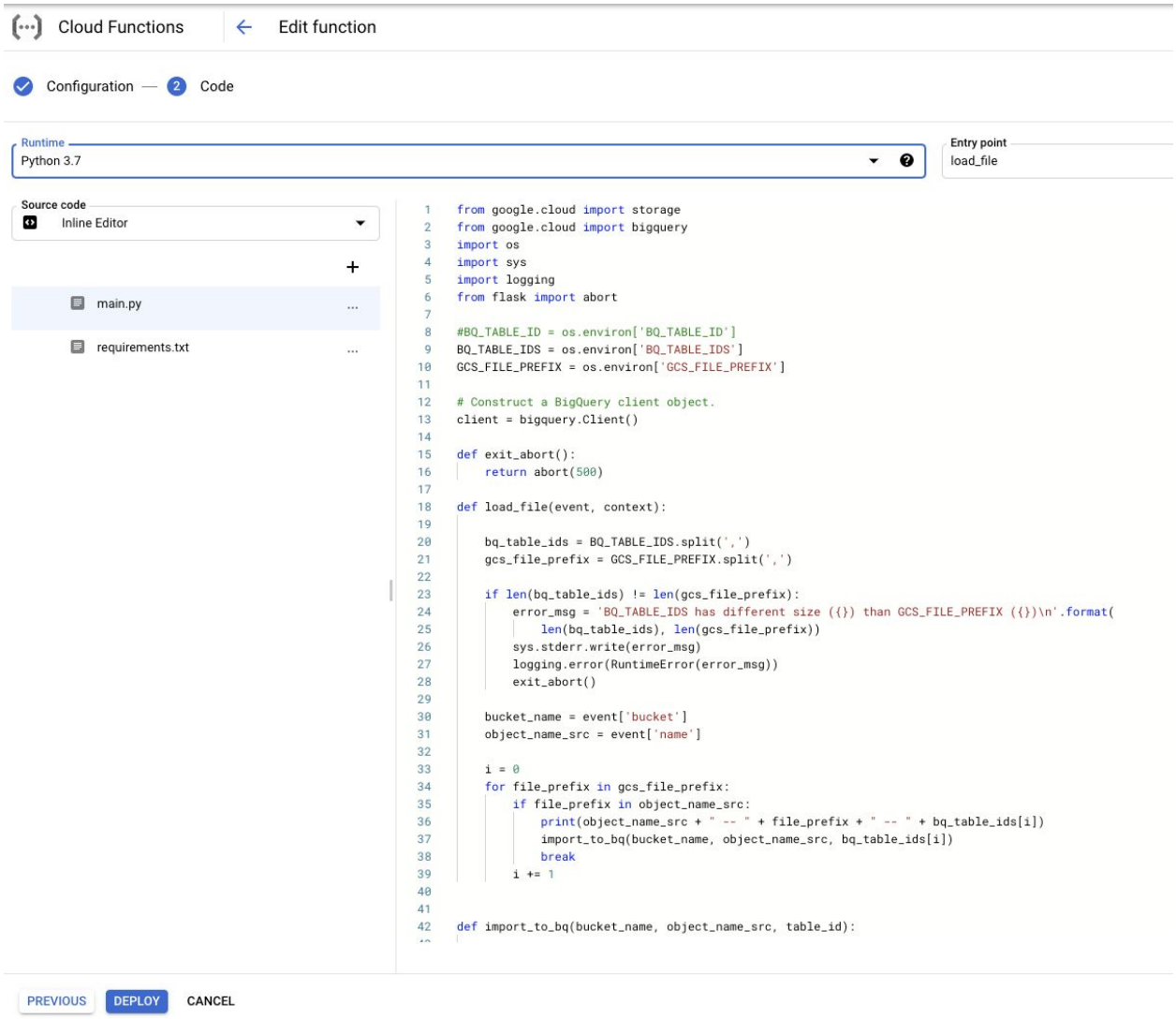
- Name: GCS_FILE_PREFIX
- Value: 需要导入的文件内的关键字前缀，如“user_id,produc_id”

- Name：BQ_TABLE_IDS
- Value：目的表名列表，用逗号隔开，表名格式为 `your-project.your_dataset.your_table_name`



在下一步配置函数代码。首先运行时选择Python 3.7，Entry Point是初始调用的函数，选择代码里的入口函数名。

因为上面“Source code”选择了“Inline editor”，所以可以用内建的网页代码编辑器输入代码。将示例代码填入“main.py”标签的代码输入框。



本例的代码可以在Github上查看。地址是

https://github.com/eugeneyu/cloud-demos/blob/master/cloud-functions/load_from_gcs_to_bq.py

也可以从下面拷贝。

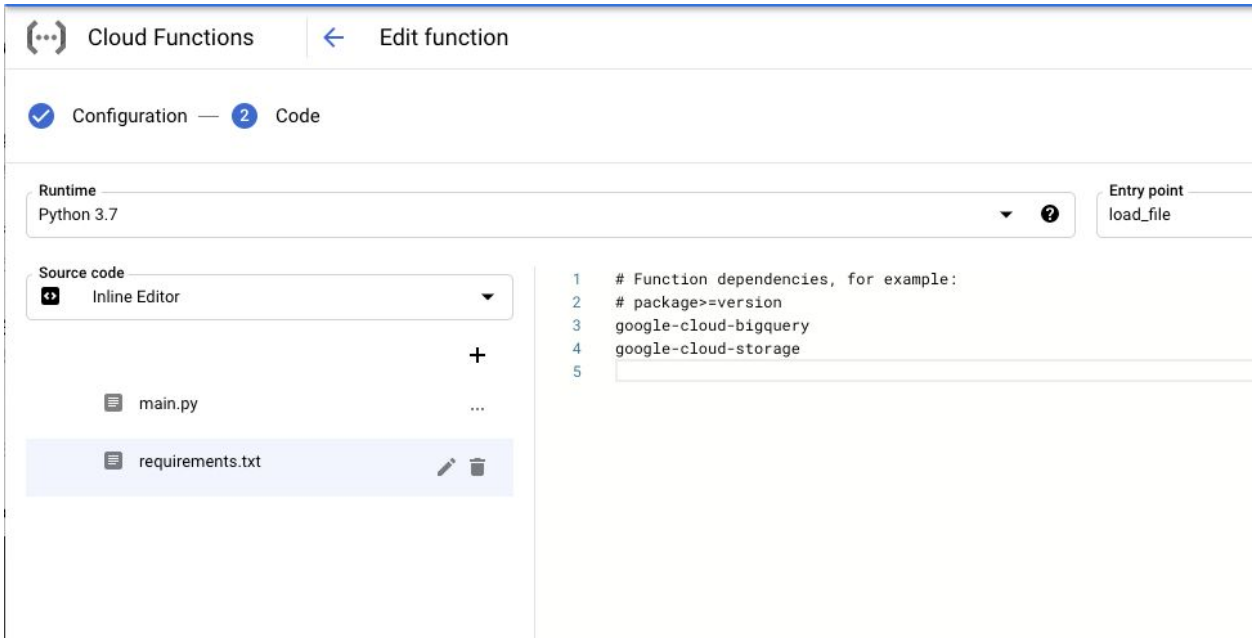
```
from google.cloud import storage
from google.cloud import bigquery
import os
import sys
import logging
from flask import abort

BQ_TABLE_IDS = os.environ['BQ_TABLE_IDS']
```

```
GCS_FILE_PREFIX = os.environ['GCS_FILE_PREFIX']

# Construct a BigQuery client object.
client = bigquery.Client()

def exit_abort():
    return abort(500)

def load_file(event, context):

    bq_table_ids = BQ_TABLE_IDS.split(',')
    gcs_file_prefix = GCS_FILE_PREFIX.split(',')

    if len(bq_table_ids) != len(gcs_file_prefix):
        error_msg = 'BQ_TABLE_IDS has different size ({}) than
GCS_FILE_PREFIX ({})\n'.format(
            len(bq_table_ids), len(gcs_file_prefix))
        sys.stderr.write(error_msg)
        logging.error(RuntimeError(error_msg))
        exit_abort()

    bucket_name = event['bucket']
    object_name_src = event['name']

    i = 0
    for file_prefix in gcs_file_prefix:
        if file_prefix in object_name_src:
            print(object_name_src + " -- " + file_prefix + " -- " +
bq_table_ids[i])
            import_to_bq(bucket_name, object_name_src, bq_table_ids[i])
            break
        i += 1


def import_to_bq(bucket_name, object_name_src, table_id):

    job_config =
bigquery.LoadJobConfig(source_format=bigquery.SourceFormat.ORC)

    destination_table = client.get_table(table_id)

    row_num_before_load = destination_table.num_rows
```

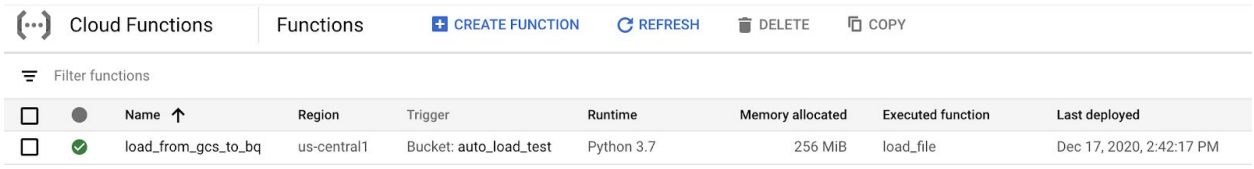
```
    # uri is like
"gs://cloud-samples-data/bigquery/us-states/us-states.orc"
    uri = "gs://" + bucket_name + "/" + object_name_src

    load_job = client.load_table_from_uri(
        uri, table_id, job_config=job_config
    )  # Make an API request.

    load_job.result()  # Waits for the job to complete.

    destination_table = client.get_table(table_id)
    row_num_after_load = destination_table.num_rows
    print("Loaded {} rows.".format(row_num_after_load-row_num_before_load))
```

然后在“requirements.txt”标签输入框加上两行，指定要用到的外部库：

```
google-cloud-bigquery
google-cloud-storage
```



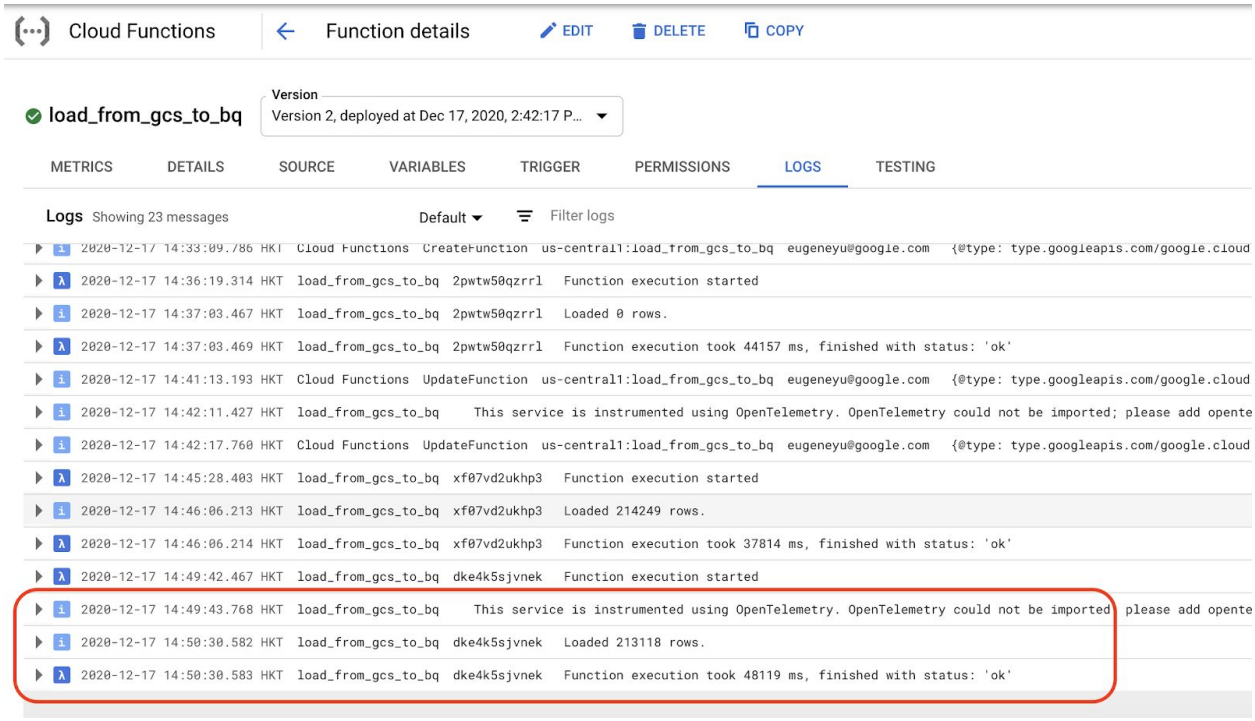
然后点击部署，等待几分钟可以看到函数已经部署成功。



然后拷贝或上传一个ORC文件到函数监控的存储桶，来触发函数执行。

```
gsutil cp
gs://hago_test2/xxx_dwd_user_uid_day_detail/dt=20201205/part-00000-3dbb42bb
-6f27-428f-8788-38ccc7509e86-c000
gs://auto_load_test/xxx_dwd_user_uid_day_detail/dt=20201205/part-00000-3dbb
42bb-6f27-428f-8788-38ccc7509e86-c000
```

然后在Cloud Functions函数详情页可以看到执行的日志，以及其它监控状态。



这时也可以到BigQuery控制台查看目的表的更新时间和行数，确认导入成功。