Cloud Spanner - Loading Data and Performing Backups

Objective

To learn and practice:

- Loading data into **Cloud Spanner** using different methods.
- Performing **database backups** within Google Cloud Spanner.
- Understanding data handling at scale on a **managed relational database service**.

Tools Used

- **Google Cloud Console**
- **Cloud Shell**
- **gcloud CLI**
- **Python client library**
- **Dataflow**
- **Cloud Storage**

Task 1: Explore the Instance

- Navigated to **Cloud Spanner > banking-instance > banking-db**.
- Verified existing **Customer table schema**.
- Confirmed it was empty using:

```
SELECT * FROM Customer;
```

Task 2: Insert Data using DML

Executed a **DML insert** using gcloud:

```
gcloud spanner databases execute-sql banking-db --instance=banking-instance \

--sql="INSERT INTO Customer (CustomerId, Name, Location) VALUES ('bdaaaa97-1b4b-4e58-b4ad-84030de92235', 'Richard Nelson', 'Ada Ohio')"
```

Verified the inserted row in Console > Customer table > Data.

Task 3. Insert data through a client library

The optimal way to access Spanner is via a programmatic interface. There are a wide variety of client libraries including **C++, C#, Go, Java, Node.js, PHP, Python and Ruby**.

1. In the Cloud Shell enter the following command to invoke the **Nano** text editor and create a new empty configuration file named **insert.py**.

```
nano insert.py
```

```
from google.cloud import spanner from google.cloud.spanner_v1 import param_types

INSTANCE_ID = "banking-instance" DATABASE_ID = "banking-db"

spanner_client = spanner.Client() instance = spanner_client.instance(INSTANCE_ID)
database = instance.database(DATABASE_ID)

def insert_customer(transaction): row_ct = transaction.execute_update( "INSERT INTO
Customer (CustomerId, Name, Location)" "VALUES ('b2b4002d-7813-4551-b83b-
366ef95f9273', 'Shana Underwood', 'Ely Iowa')" ) print("{} record(s)
inserted.".format(row_ct))

database.run_in_transaction(insert_customer)
```

3. Press **Ctrl+X** to exit Nano, **Y** to confirm the update, and press **Enter** to save your changes.
4. Run the python code.

```
python3 insert.py
```

5. Refresh the Cloud Console, or click on a different item on the left menu and then click again on **Data** and you will see the new row in your database.

Task 4. Insert batch data through a client library

A more optimal way to load data into Spanner is doing so in batches. All of the client libraries support batch loading. This example uses Python.

1. In the Cloud Shell enter the following command to invoke the **Nano** text editor and create a new empty configuration file named **batch_insert.py**.

```
nano batch_insert.py
```

2. Paste the code block listed below.

```
from google.cloud import spanner from google.cloud.spanner_v1 import param_types
```

```
INSTANCE_ID = "banking-instance" DATABASE_ID = "banking-db"

spanner_client = spanner.Client() instance = spanner_client.instance(INSTANCE_ID)
database = instance.database(DATABASE_ID)

with database.batch() as batch: batch.insert( table="Customer", columns=("CustomerId",
"Name", "Location"), values=[ ('edfc683f-bd87-4bab-9423-01d1b2307c0d', 'John Elkins',
'Roy Utah'), ('1f3842ca-4529-40ff-acdd-88e8a87eb404', 'Martin Madrid', 'Ames Iowa'),
('3320d98e-6437-4515-9e83-137f105f7fbc', 'Theresa Henderson', 'Anna Texas'),
('6b2b2774-add9-4881-8702-d179af0518d8', 'Norma Carter', 'Bend Oregon'),

    ],
)


print("Rows inserted")
```

3. Press **Ctrl+X** to exit Nano, **Y** to confirm the update, and press **Enter** to save your changes.

Task 5. Load data using Dataflow

Dataflow is a Google Cloud service for streaming and batch data processing at large scale. Dataflow uses multiple workers to run data processing in parallel. The way in which data is processed is defined using **pipelines** that transform data from its origin (**sources**) to its destination (**sinks**).

There are connectors for **Spanner** that allow you to connect a database as a **source** or a **sink** in Dataflow.

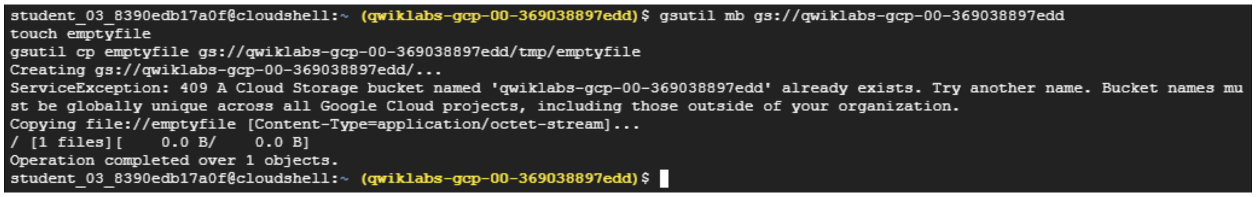
In order to load big amounts of data, you can use the serverless distributed power of **Dataflow** to read data from a source (for example, a CSV file in **Google Cloud Storage**) and load it into your **Spanner** database using a sink connector.

1. To prepare for the Dataflow job, in the Cloud Shell run these commands to create a bucket in your project and a folder with an empty file inside it.

```
gsutil mb gs://Project ID

touch emptyfile

gsutil cp emptyfile gs://Project ID/tmp/emptyfile
```

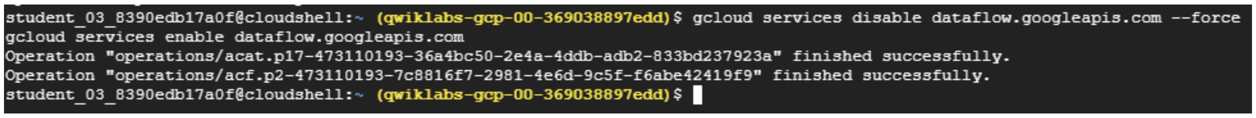


```
student_03_8390edb17a0f@cloudshell:~ (qwiklabs-gcp-00-369038897edd)$ gsutil mb gs://qwiklabs-gcp-00-369038897edd
touch emptyfile
gsutil cp emptyfile gs://qwiklabs-gcp-00-369038897edd/tmp/emptyfile
Creating gs://qwiklabs-gcp-00-369038897edd/...
ServiceException: 409 A Cloud Storage bucket named 'qwiklabs-gcp-00-369038897edd' already exists. Try another name. Bucket names mu
st be globally unique across all Google Cloud projects, including those outside of your organization.
Copying file://emptyfile [Content-Type=application/octet-stream]...
/ [1 files][    0.0 B/    0.0 B]
Operation completed over 1 objects.
student_03_8390edb17a0f@cloudshell:~ (qwiklabs-gcp-00-369038897edd)$
```

2. To ensure that the proper APIs and permissions are set, execute the following block of code in the Cloud Shell.

gcloud services disable dataflow.googleapis.com --force

gcloud services enable dataflow.googleapis.com



```
student_03_8390edb17a0f@cloudshell:~ (qwiklabs-gcp-00-369038897edd)$ gcloud services disable dataflow.googleapis.com --force
gcloud services enable dataflow.googleapis.com
Operation "operations/acat.p17-473110193-36a4bc50-2e4a-4ddb-adb2-833bd237923a" finished successfully.
Operation "operations/acf.p2-473110193-7c8816f7-2981-4e6d-9c5f-f6abe42419f9" finished successfully.
student_03_8390edb17a0f@cloudshell:~ (qwiklabs-gcp-00-369038897edd)$
```

3. From the Console, open the navigation menu (☰) > **View All Products**. Under **Analytics** section, click **Dataflow**.
4. On the top of the screen, click **Create Job From Template**.
5. Place the following values in the template:
a. **Job Name**: spanner-load
b. **Regional endpoint**: `default-region`
6. Scroll down the **Dataflow template** selector and you will see all the different blueprints you can use with Dataflow. Of course, you can also create your own tailored pipelines, using the Beam SDK.

There are two main types of templates:

- **Stream** will create a pipeline for data that is flowing and is processed continuously (for example, online orders from a website).
- **Batch** will process a dataset that has a beginning and an end (for example, files stored in Google Cloud Storage).

In your scenario, you will load data into Spanner banking database from a CSV file with over 150,000 rows.

7. Select the **Text Files on Cloud Storage to Cloud Spanner** template.
8. Place the following values in the template:

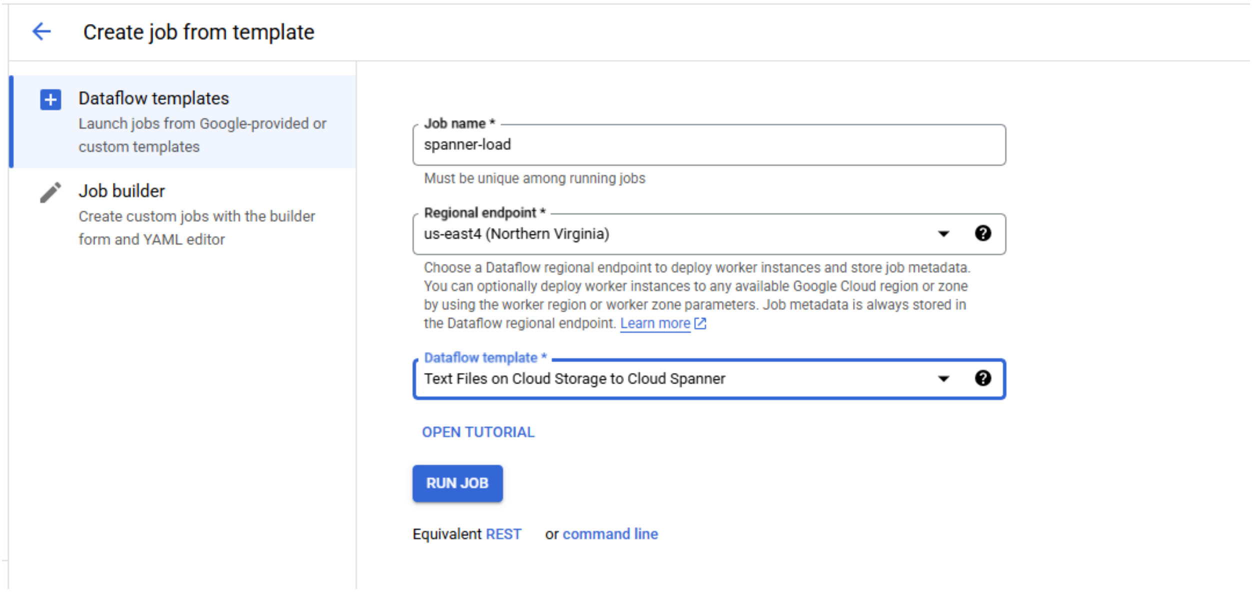
Item	Value

Cloud Spanner Instance Id	**banking-instance**
Cloud Spanner Database Id	**banking-db**
Text Import Manifest file	**cloud-training/OCBL372/manifest.json**

For the **Temporary Location** parameter input the following valu

Project ID/tmp

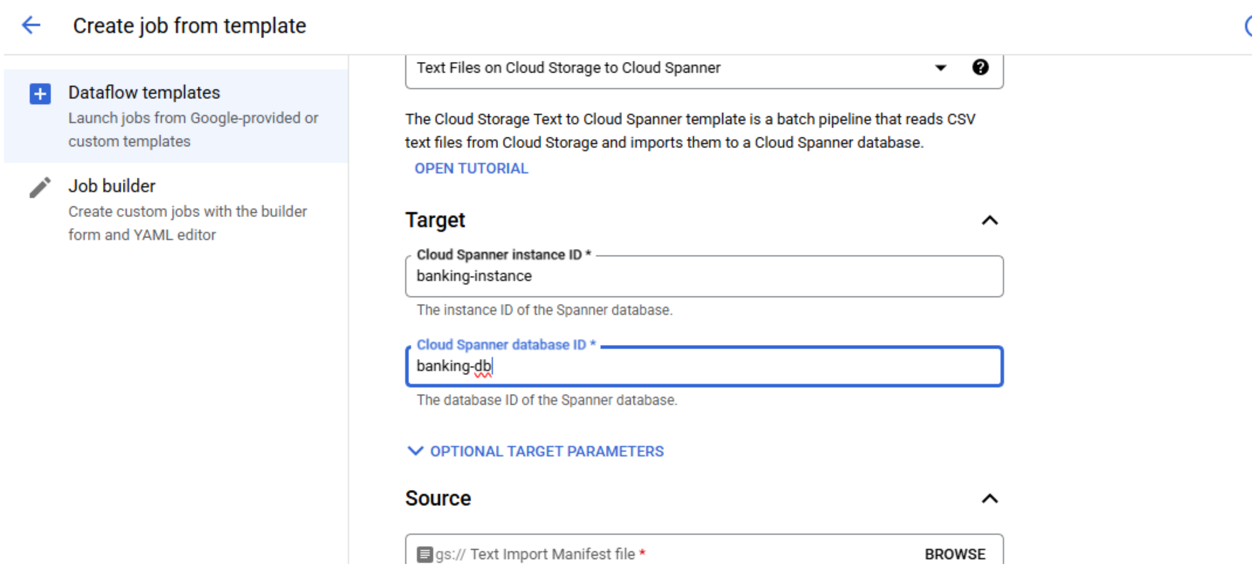
10. Expand **Optional Parameters**.
11. Uncheck **Use default machine type**.
12. Under **General purpose**, choose the following:
    a. Series: **E2**
    b. Machine type: **e2-medium (2 vCPU, 4 GB memory)**
13. Click **Run Job** to start the pipeline.
14. The process will take around 12 to 16 minutes. You will see Dataflow go through multiple stages, first starting up the workers and analyzing the pipeline from the template. Then it will read the manifest file and will start processing the CSV file.


Create job from template
Dataflow templates
Launch jobs from Google-provided or custom templates
Job builder
Create custom jobs with the builder form and YAML editor
OPEN TUTORIAL
RUN JOB
Equivalent REST or command line

Job name *
spanner-load
Must be unique among running jobs

Regional endpoint *
us-east4 (Northern Virginia)
Choose a Dataflow regional endpoint to deploy worker instances and store job metadata. You can optionally deploy worker instances to any available Google Cloud region or zone by using the worker region or worker zone parameters. Job metadata is always stored in the Dataflow regional endpoint. Learn more

Dataflow template *
Text Files on Cloud Storage to Cloud Spanner



Create job from template

Dataflow templates
Launch jobs from Google-provided or custom templates
Job builder
Create custom jobs with the builder form and YAML editor

Text Files on Cloud Storage to Cloud Spanner

The Cloud Storage Text to Cloud Spanner template is a batch pipeline that reads CSV text files from Cloud Storage and imports them to a Cloud Spanner database.

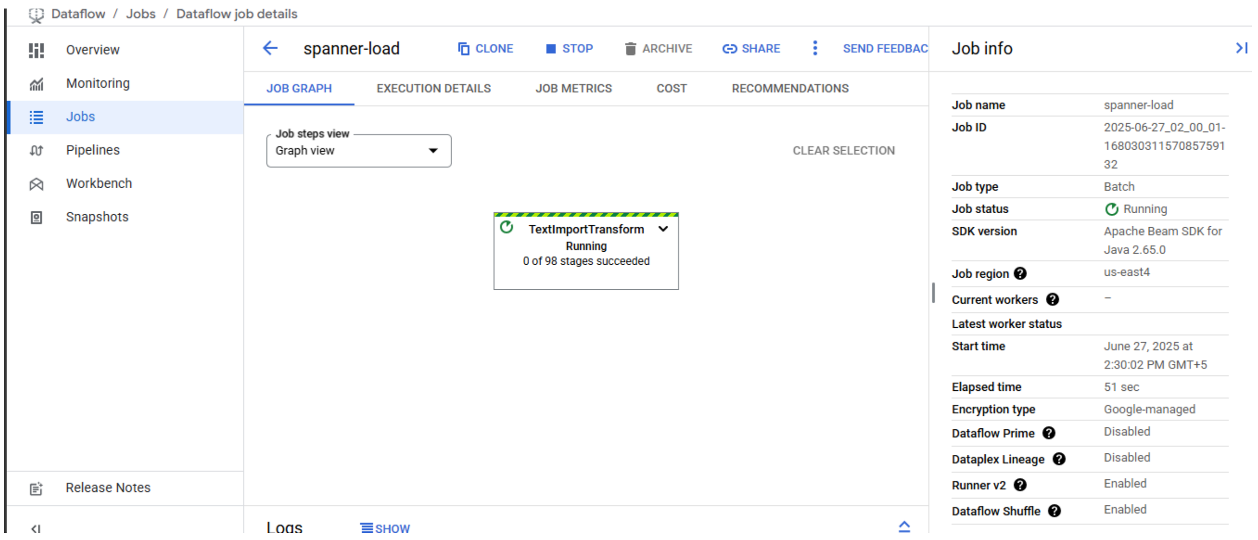
Target

Cloud Spanner instance ID *: banking-instance
The instance ID of the Spanner database.

Cloud Spanner database ID *: banking-db
The database ID of the Spanner database.

Source

gs:// Text Import Manifest file *



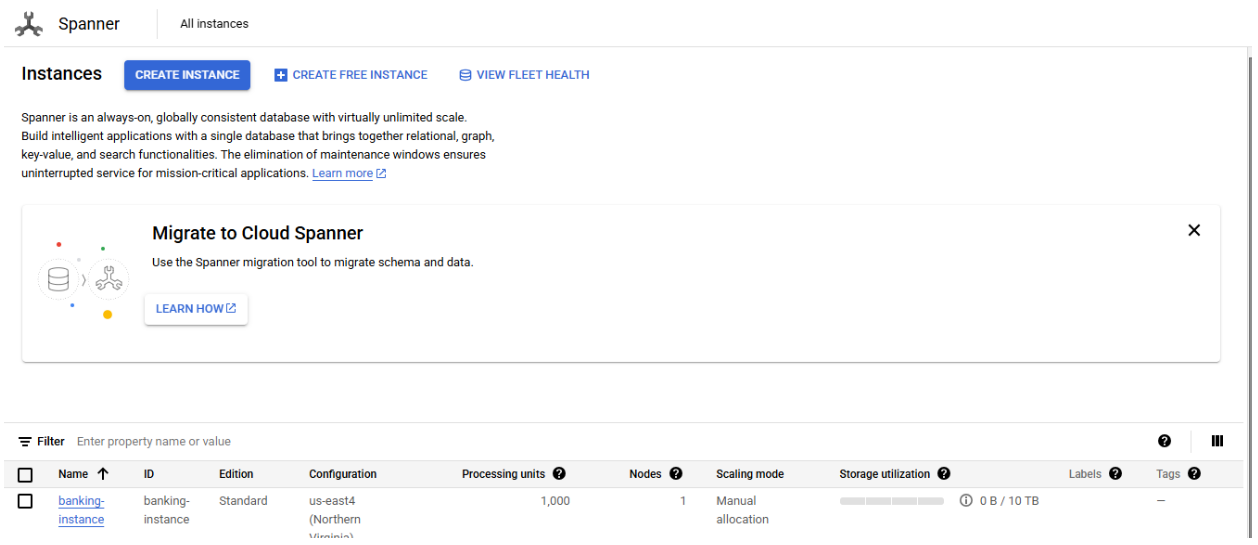
spanner-load

Job steps view: Graph view

TextImportTransform
Running
0 of 98 stages succeeded

Job info

Job name	spanner-load
Job ID	2025-06-27_02_00_01-16803031157085759132
Job type	Batch
Job status	Running
SDK version	Apache Beam SDK for Java 2.65.0
Job region	us-east4
Current workers	–
Latest worker status	
Start time	June 27, 2025 at 2:30:02 PM GMT+5
Elapsed time	51 sec
Encryption type	Google-managed
Dataflow Prime	Disabled
Dataplex Lineage	Disabled
Runner v2	Enabled
Dataflow Shuffle	Enabled



Instances

Spanner is an always-on, globally consistent database with virtually unlimited scale. Build intelligent applications with a single database that brings together relational, graph, key-value, and search functionalities. The elimination of maintenance windows ensures uninterrupted service for mission-critical applications. Learn more

Migrate to Cloud Spanner

Use the Spanner migration tool to migrate schema and data.

Name	ID	Edition	Configuration	Processing units	Nodes	Scaling mode	Storage utilization	Labels	Tags
banking-instance	banking-instance	Standard	us-east4 (Northern Virginia)	1,000	1	Manual allocation	0 B / 10 TB		–

Task 6. Backup your database

Using **Dataflow** as explained above is a way to create backups of your data. But **Spanner** has its own tool for backups. You can backup a **Spanner** database from the Cloud Console, client libraries or **gcloud** commands. Check the previous links for documentation.

In this lab, you will use the **Cloud Console** to backup your database.

1. Select **Backup/Restore** from the left menu.
2. Click **Create Backup**.
3. Place or select the following values in the wizard:

Item	**Value**
Database Name	**banking-db**
Backup Name	**banking-backup-001**
Expiration Date	**1 year**

4. Click **Create**.
5. The backup will take around 15 minutes to complete and will appear in the **Backups** list while being created.