

Lecture: Cloud SQL Hands On

Google Cloud Platform data-engineer SQL Create a MySQL instance

Instance info

Instance ID
Choice is permanent. Use lowercase letters, numbers, and hyphens. Start with a letter.
test-sql-instance

Root password
Set a password for the root user. Learn more
test-password Generate
No password

Location
For better performance, keep your data close to the services that need it.

Region
Choice is permanent
us-central1 (Iowa)

Zone
Can be changed at any time
us-central1-a

Database version
MySQL 5.7

Check how **machine type** affect the **network throughput** rate:

Configuration options

1 Connectivity
Public IP enabled

2 Machine type and storage

Machine type
For better performance, choose a machine type with enough memory to hold your largest table

db-n1-standard-1
vCPUs 1 Memory 3.75 GB Change

Network throughput (MB/s) 250 of 2,000

Configuration options

1 Connectivity
Public IP enabled

2 Machine type and storage

Machine type
For better performance, choose a machine type with enough memory to hold your largest table

db-n1-highmem-96
vCPUs 96 Memory 624 GB

Network throughput (MB/s)

Storage type
Choice is permanent.

- SSD (Recommended)
Most popular choice. Lower latency than HDD data throughput.
- HDD
Lower performance than SSD with lower storage capacity.

Storage capacity
10 – 30720 GB. Higher capacity improves performance by the machine type. Capacity can't be decreased later.

100 GB

- db-g1-small
1vCPU, 1.7 GB
- db-n1-standard-1
1vCPU, 3.75 GB
- db-n1-standard-2
2vCPU, 7.5 GB
- db-n1-standard-4
4vCPU, 15 GB
- db-n1-standard-8
8vCPU, 30 GB
- db-n1-standard-16
16vCPU, 60 GB
- db-n1-standard-32
32vCPU, 120 GB
- db-n1-standard-64
64vCPU, 240 GB
- db-n1-standard-96
96vCPU, 360 GB
- db-n1-highmem-2
2vCPU, 13 GB
- db-n1-highmem-4
4vCPU, 26 GB
- db-n1-highmem-8
8vCPU, 52 GB
- db-n1-highmem-16
16vCPU, 104 GB
- db-n1-highmem-32
32vCPU, 208 GB
- db-n1-highmem-64
64vCPU, 416 GB
- db-n1-highmem-96
96vCPU, 624 GB

And the size of **disk** (storage) can affect performance:

Storage type ⓘ
Choice is permanent.

- SSD (Recommended)**
Most popular choice. Lower latency than HDD with higher QPS and data throughput.
- HDD**
Lower performance than SSD with lower storage rates.

Storage capacity ⓘ
10 – 30720 GB. Higher capacity improves performance, up to the limits set by the machine type. Capacity can't be decreased later.

10 GB

Enable automatic storage increases
If enabled, whenever you're nearing capacity, storage will be incrementally (and permanently) increased. [Learn more](#)

Disk throughput (MB/s) ⓘ
Read: 4.8 Max: 240.0 Write: 4.8 Max: 72.0

IOPS ⓘ
Read: 300 Max: 15,000 Write: 300 Max: 4,500

Storage type ⓘ
Choice is permanent.

- SSD (Recommended)**
Most popular choice. Lower latency than HDD with higher QPS and data throughput.
- HDD**
Lower performance than SSD with lower storage rates.

Storage capacity ⓘ
10 – 30720 GB. Higher capacity improves performance, up to the limits set by the machine type. Capacity can't be decreased later.

10 GB

Enable automatic storage increases
If enabled, whenever you're nearing capacity, storage will be incrementally (and permanently) increased. [Learn more](#)

Disk throughput (MB/s) ⓘ
Read: 48.0 Max: 240.0 Write: 48.0 Max: 72.0

IOPS ⓘ
Read: 3,000 Max: 15,000 Write: 3,000 Max: 4,500

When create a SQL instance:

We can view ‘datasets’, ‘backups’, and ‘replicas’ in this instance.

Google Cloud Platform data-engineer Search products and resources

SQL Overview EDIT IMPORT EXPORT RESTART STOP DELETE CLONE

MASTER INSTANCE

- test-sql-instance**
- MySQL 5.7
- CPU utilization ▾

1 hour 6 hours 12 hours 1 day 2 days 4 days 7 days 14 days 30 days May 20, 2020 2:53 PM

100%
80%
60%
40%
20%
0%

2:50 2:55 3 PM 3:05 3:10 3:15 3:20 3:25 3:30 3:35 3:40 3:45

Connect to this instance

Public IP address: 104.198.252.8 Instance connection name: data-engineer-274519:us-central1:test-sql-instance

- Connect using Cloud Shell
- Connect from a Compute Engine VM instance
- See all connection methods

Suggested actions

- Create a backup
- Enable high availability

Configuration

vCPUs	Memory	SSD storage
1	3.75 GB	10 GB

- Database version is MySQL 5.7
- Auto storage increase is enabled
- Automated backups are enabled
- Point-in-time recovery is enabled
- Located in us-central1-a
- No database flags set
- No labels set
- Not highly available (zonal)

→ Edit configuration

Lecture: Importing Data

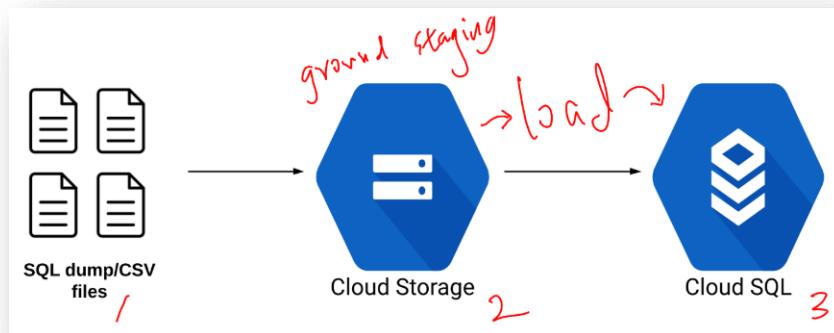
In this lesson, we will cover how to bulk import data into a Cloud SQL session, including SQL dump and CSV files. Detailed steps of what was performed in the lesson are below:

We will be using a sample data GitHub repo

here: <https://github.com/linuxacademy/googledataengineer>

The original source of this data from Google's GitHub can be viewed

here: <https://github.com/GoogleCloudPlatform/training-data-analyst>



1. Clone our sample data to Cloud Shell:

- Open Cloud Shell.
- Clone the data from *repo* to Cloud Shell.
- `git clone https://github.com/linuxacademy/googledataengineer`

2. Create the **Cloud Storage bucket** (as staging ground) to copy data:

- `gsutil mb -l (your_region) gs://(bucket_name)`

```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gsutil mb -l us-central1 gs://sql-practice
Creating gs://sql-practice/...
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$
```

The screenshot shows the Google Cloud Platform Storage browser interface. The left sidebar has a 'Storage' tab highlighted with a blue circle. The main area shows a table with one row of data. The columns are 'Name' (with an upward arrow icon), 'Created', 'Location type', 'Location', and 'Default storage class'. The single row contains the value 'sql-practice' under 'Name', 'May 20, 2020, 7:56:32 PM' under 'Created', 'Region' under 'Location type', 'us-central1 (lo...' under 'Location', and 'Standard' under 'Default storage class'. A blue circle highlights the 'sql-practice' entry in the table.

3. Browse to the CloudSQL sample data directory:

- `cd googledataengineer/CPB100/lab3a/cloudsql`

4. Copy all data into your cloud bucket:

- o `gsutil cp * gs://(bucket_name)` (* means transfer all the files)

```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gsutil cp * gs://sql-practice
Copying file://accommodation.csv [Content-Type=text/csv]...
Copying file://rating.csv [Content-Type=text/csv]...
Copying file://table_creation.sql [Content-Type=application/x-sql]...
- [3 files] [ 14.2 KiB/ 14.2 KiB]
Operation completed over 3 objects/14.2 KiB.
```

The screenshot shows the Google Cloud Platform Storage interface. A blue box highlights the 'Storage' icon in the sidebar. Another blue box highlights the 'sql-practice' bucket name in the top navigation bar. The main area displays the 'Objects' tab with three files listed:

Name	Size	Type	Storage class	Last modified	Public access	Encryption
accommodation.csv	4.78 KB	text/csv	Standard	5/20/20, 8:00:02 PM UTC-4	Not public	Google-managed ke
rating.csv	8.7 KB	text/csv	Standard	5/20/20, 8:00:02 PM UTC-4	Not public	Google-managed ke
table_creation.sql	770 B	application/x-sql	Standard	5/20/20, 8:00:02 PM UTC-4	Not public	Google-managed ke

5. From **cloud storage bucket**, Import an SQL dump file into **Cloud SQL**:

- o From Cloud SQL, click on the instance and click the **Import** button

The screenshot shows the Google Cloud Platform SQL interface. A blue box highlights the 'SQL' icon in the sidebar. Another blue box highlights the 'IMPORT' button in the top navigation bar. The main area shows the 'MASTER INSTANCE' section for 'test-sql-instance' with MySQL 5.7. The 'CPU utilization' chart has a blue box highlighting the '1 hour' time range. The date at the bottom right is May 20, 2020.

- o Click **Browse**, select **bucket**, browse to `table_creation.sql`, and click **Select**

The screenshot shows the 'Import data from Cloud Storage' page in the Google Cloud Platform SQL interface. On the left, there's a sidebar with 'MASTER INSTANCE' options: Overview (selected), Connections, Users, Databases, Backups, and Replicas. The main area has a 'Source' section with a 'Browse' button highlighted by a blue oval. Below it, under 'Format of import', 'SQL' is selected (radio button highlighted). A note says: 'A plain text file with a sequence of SQL commands, like the output of mysqldump'. There's also a 'CSV' option with a note: 'If your Cloud Storage file is a CSV file, select CSV. The CSV file should be a plain text file with one line per row and comma-separated fields.'

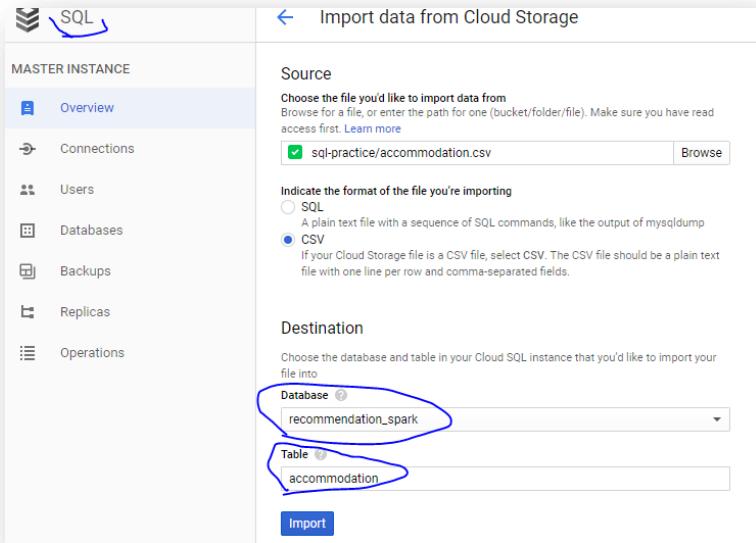
- Under the "Format of import" options, make sure **SQL** is selected as the import format (should be default).

This screenshot shows the 'Select a file to import' dialog. It has a 'Location' field containing 'sql-practice' with a blue oval around it. Below it is a list of files: 'accommodation.csv', 'rating.csv', and 'table_creation.sql', with 'table_creation.sql' highlighted by a blue oval.

- Click **Import**.

6. Import CSV tables into SQL database (recommendation_spark):

- From Cloud SQL, click on the instance and click the **Import** button.
- Click **Browse**, select **bucket**, browse to **accommodation.csv**, and click **Select**.
- Expand **advanced options**, and from the Database drop-down menu, select **recommendation_spark**.
- Set the Table name to **Accommodation**.



- Click **Import**.
- Perform the same actions for the `rating.csv` file as well, setting the Table name to **Rating**.

7. Connect to your Cloud SQL instance:
- Click **Connect using Cloud Shell**.

Connect to this instance

Public IP address
104.198.252.8

Instance connection name
data-engineer-274519:us-central1:test-sql-instance

Connect using Cloud Shell

Connect from a Compute Engine VM instance

See all connection methods

Configuration

vCPUs	Memory	SSD
1	3.75 GB	10

Database version is MySQL 5.7

Auto storage increase is enabled

Automated backups are enabled

Point-in-time recovery is enabled

Located in us-central1-a

No database flags set

No labels set

Not highly available (zonal)

[Edit configuration](#)

- In Cloud Shell, hit enter once the command is **populated**.

```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gcloud sql connect test-sql-instance --user=root
t --quiet
Whitelisting your IP for incoming connection for 5 minutes...done.
```

- Enter the root password when prompted.

```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gcloud sql connect test-sql-instance --user=root
t --quiet
Whitelisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 1485  
Server version: 5.7.25-google-log (Google)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
```

- Now we are in MySQL!!

```
leonf_info@cloudshell:~/googledataengineer/CPB100/lab3a/cloudsql (data-engineer-274519)$ gcloud sql connect test-sql-instance --user=root
t --quiet
Whitelisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 1485  
Server version: 5.7.25-google-log (Google)

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |
```

8. View tables and table data:

- Switch to database:

- `use recommendation_spark;`

```
mysql> use recommendation_spark |  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
Database changed |
```

- o View tables:

- `show tables;`

```
mysql> show tables;
+-----+
| Tables_in_recommendation_spark |
+-----+
| Accommodation                   |
| Rating                          |
| Recommendation                  |
+-----+
3 rows in set (0.03 sec)
```

- o View contents of one of the tables:

- `select * from Rating;`

```
mysql> select * from Accommodation;
+----+-----+-----+-----+-----+-----+-----+
| id | title           | location      | price | rooms | rating | type   |
+----+-----+-----+-----+-----+-----+-----+
| 1  | Comfy Quiet Chalet | Vancouver    | 50    | 3     | 3.1   | cottage |
| 10 | Sizable Calm Country House | Auckland    | 650   | 9     | 4.9   | mansion |
| 11 | Homy Quiet Shanty | Melbourne   | 50    | 1     | 2.8   | cottage |
| 12 | Beautiful Peaceful Villa | Seattle     | 90    | 2     | 2.1   | house   |
| 13 | Enormous Peaceful Fortress | Melbourne  | 3300  | 12    | 2.3   | castle  |
| 14 | Colossal Peaceful Palace | Melbourne  | 1200  | 21    | 1.5   | castle  |
| 15 | Vast Private Fort | London      | 1300  | 18    | 2.6   | castle  |
| 16 | Large Calm House | Melbourne  | 45    | 3     | 4.1   | house   |
| 17 | Small Quiet Cabin | Vancouver  | 250   | 2     | 3.5   | cottage |
+----+-----+-----+-----+-----+-----+-----+
```



Lecture: Instance Configuration (Exam!)

In this lesson, we will go hands-on with creating a **Bigtable** instance (**expensive!**), and explore how to resize instances and some basic table interaction. The steps we took to interact with our table via **cbt** command line are below for your reference.

The screenshot shows the Google Cloud Platform interface for managing Bigtable instances. At the top, there's a navigation bar with the 'Google Cloud Platform' logo, a dropdown for 'data-engineer', a search bar, and various status icons. Below the navigation is a header with 'Bigtable' (circled in blue), 'Instances', and a 'CREATE INSTANCE' button. A 'SHOW INFO PANEL' link is on the right. The main content area displays a message: 'Cloud Bigtable is a fully managed, wide-column NoSQL database that offers low latency and replication for high availability. You can provision Cloud Bigtable instances for your workload, then use the Bigtable HBase client to develop applications, as well as other standard open-source big data tools.' followed by a 'Learn more' link. Below this is a notification: 'You've got a new instance! Connect to it with the cbt command-line tool. [Learn more](#)'. The main table lists instances with columns: 'Instance ID' (checkbox), 'Instance name' (dropdown), 'Application profiles', 'Zones', 'Nodes', and 'Storage utilization'. The first row, 'example-instance', has its 'Instance ID' column circled in blue. The table shows one node and no storage utilized.

1. Install the **cbt** command in Google SDK:

- **gcloud components update**

```
ERROR: (gcloud.components.update)
You cannot perform this action because the Cloud SDK component manager
is disabled for this installation. You can run the following command
to achieve the same result for this installation:

sudo apt-get update && sudo apt-get --only-upgrade install google-cloud-sdk-skaffold kubectl google-cloud-sdk-anthos-auth google-cloud-s
dk-minikube google-cloud-sdk google-cloud-sdk-app-engine-grpc google-cloud-sdk-kind google-cloud-sdk-pubsub-emulator google-cloud-sdk-ap
p-engine-go google-cloud-sdk-firebase-emulator google-cloud-sdk-cloud-build-local google-cloud-sdk-datastore-emulator google-cloud-sdk-
kpt google-cloud-sdk-app-engine-python google-cloud-sdk-spanner-emulator google-cloud-sdk-cbt google-cloud-sdk-bigtable-emulator google-
cloud-sdk-app-engine-python-extras google-cloud-sdk-datalab google-cloud-sdk-app-engine-java
```

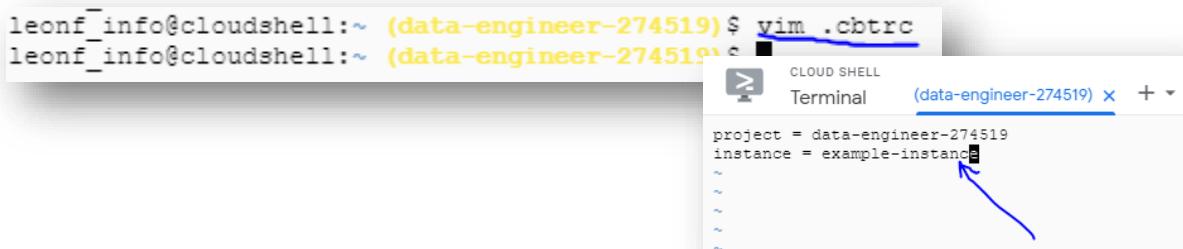
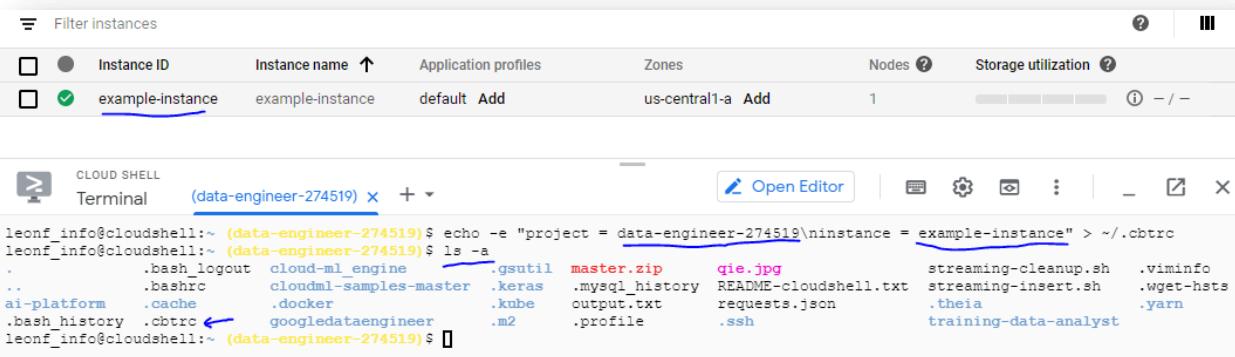
- **gcloud components install cbt**

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud components install cbt
ERROR: (gcloud.components.install)
You cannot perform this action because the Cloud SDK component manager
is disabled for this installation. You can run the following command
to achieve the same result for this installation:

sudo apt-get install google-cloud-sdk-cbt
```

2. Configure `cbt` to use your project and instance via `.cbtrc` file:

- echo -e "project = [PROJECT_ID]\ninstance = [INSTANCE_ID]" > ~/.cbtrc



3. Create a table:

- cbt createtable my-table

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt createtable my-table  
2020/05/21 21:50:30 -creds flag unset, will use gcloud credential
```

4. List the table (now we only have ‘my-table’):

- cbt ls

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt ls  
2020/05/21 21:51:15 -creds flag unset, will use gcloud credential  
my-table
```

5. Add a column family:

- `cbt createfamily my-table cf1`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt createtable my-table cfl  
2020/05/21 21:52:42 -creds flag unset, will use gcloud credential
```

6. List column family

- `cbt ls my-table`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt ls my-table
2020/05/21 21:53:19 -creds flag unset, will use gcloud credential
Family Name      GC Policy
-----      -----
cf1            <never>
```



7. Add value to *row 1*, using column family *cf1* and column qualifier *c1*:

- `cbt set my-table r1 cf1:c1=test-value`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt set my-table r1 cf1:c1=test-value
2020/05/21 21:54:17 -creds flag unset, will use gcloud credential
```

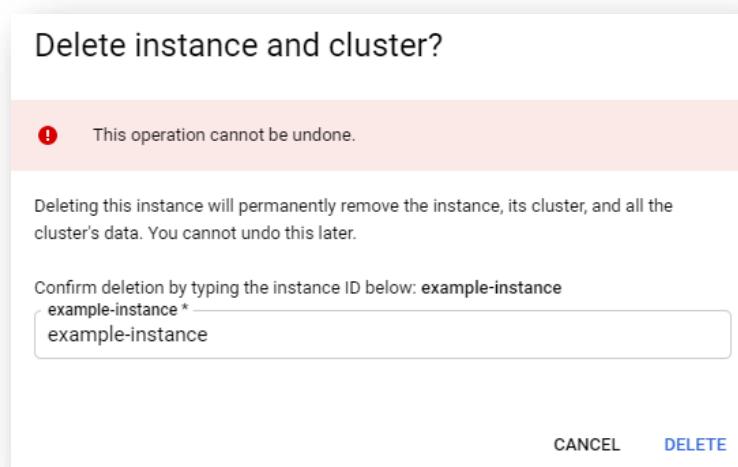
8. Use the `cbt read` command to read the data you added to the table:

- `cbt read my-table`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cbt read my-table
2020/05/21 21:54:45 -creds flag unset, will use gcloud credential
-----
r1
  cf1:c1          @ 2020/05/21-21:54:18.618000
    "test-value"
```

9. Delete the table (if not deleting instance):

- `cbt deletetable my-table`



Lecture: Hands On and Viewing Examples (**Spanner**)

Create a Spanner instance, and populate another instance using Python scripts in order to exam a more complete database example.

The below steps ran from Cloud Shell will create a Spanner instance and populate Singer and Album data for you.

[Create an instance](#)

Name your instance
An instance has both a name and an ID. The name is for display purposes only. The ID is a permanent and unique identifier.

Instance name * Name must be 4-30 characters long

Instance ID * Lowercase letters, numbers, hyphens allowed

Choose a configuration
Determines where your nodes and data are located. Permanent. Affects cost, performance, and replication.
[Compare region configurations](#)

Regional Multi-region

us-central1 (Iowa)

Allocate nodes
Add nodes to increase data throughput and queries per second (QPS). Affects billing.

Nodes *

NODE GUIDANCE

CREATE **CANCEL**

Create a **database** before we can add any **table**:

Google Cloud Platform data-engineer

Spanner

example-instance

example-instance

ID Configuration
example-instance us-central1

OVERVIEW MONITORING IMPORT/EXPORT BACKUP/RESTORE

Nodes 1 CPU utilization (mean) — Operations Read: —/s Write: —/s Throughput Read: —/s Write: —/s Total database storage — Total backups storage —

No databases yet. Create a database to get started.

CREATE DATABASE Cloud Spanner documentation

Spanner

example-instance

Create a database in example-instance

1 Name your database
Enter a permanent name for your database.
Name

Continue

2 Define your database schema (optional)

Cancel

Now put **table** into the **database**:

The image contains three screenshots from the Google Cloud Platform Spanner interface:

- Screenshot 1: Create a database in example-instance**
 - Shows the "Name your database" field set to "example_database".
 - Shows the "Define your database schema (optional)" section with a note: "Add tables and indexes to define your initial schema. You can add these anytime, but it's fastest to add them during database creation."
 - A blue arrow points to the "+ Add table" button.
 - Buttons: "Create" and "Cancel".
- Screenshot 2: Create a table**
 - Shows the "Name your table" field set to "example_table".
 - Shows the "Add columns" section with a column named "column_ID".
 - A blue circle highlights "column_ID: INT64".
 - A modal window titled "New column" shows:
 - Name: "column_input"
 - Type: "INT64"
 - NOT NULL checkbox (unchecked)
 - Buttons: "Done" and "Cancel".
- Screenshot 3: Set primary key**
 - Shows the "Set primary key" section with a note: "Each table requires a primary key of one or more columns."
 - A blue circle highlights "Key type: Single column".
 - A blue circle highlights the "Column" dropdown set to "column_ID".
 - Buttons: "Create" and "Cancel".

After all setting, click 'create':

Screenshot showing the final step of creating the database:

- Shows the "Name your database" field set to "example_database".
- Shows the "Define your database schema (optional)" section with a note: "Add tables and indexes to define your initial schema. You can add these anytime, but it's fastest to add them during database creation."
- A blue arrow points to the "Name" input field, which has "example_table" typed into it.
- A blue arrow points to the "Create" button.
- Buttons: "Create" and "Cancel".

View the table we created:

The screenshot shows the Google Cloud Spanner console. On the left, there's a sidebar with 'Spanner' at the top, followed by 'example-instance' and 'example_database'. Under 'example_database', there are 'Query' and 'example_table'. The main area is titled 'example_database' with tabs for 'OVERVIEW', 'MONITORING', 'QUERY STATS', 'IMPORT/EXPORT', and 'BACKUP/RESTORE'. The 'OVERVIEW' tab is selected. It displays metrics like CPU utilization (mean), Operations (Read: -/s, Write: -/s), Throughput (Read: -/s, Write: -/s), Total database storage, and Total backups storage. Below this is a 'Tables' section with a table header: 'Name ↑', 'Indexes', and 'Interleaved in'. A single row is listed: 'example_table' with no indexes and no interleaving. A blue circle highlights the 'Tables' section, and another blue circle highlights the 'example_table' row.

Add some data:

The screenshot shows the Google Cloud Spanner console. The left sidebar shows 'example-instance' and 'example_database'. Under 'example_database', there are 'Query' and 'example_table'. The main area is titled 'Table details' for 'example_table' with tabs for 'SCHEMA' and 'INDEXES'. A blue circle highlights the 'DATA' tab. A separate modal window is open with the title 'Insert a row in example_table'. It shows a tree view: 'Spanner' > 'example-instance' > 'example_database' > 'example_table'. It has sections for 'Primary key column' (set to 'column_ID: INT64' with value '001') and 'Columns' (set to 'column_input: INT64' with value '002'). There are 'Save' and 'Cancel' buttons at the bottom. A blue circle highlights the 'Insert a row in example_table' button.

The screenshot shows the Google Cloud Spanner console. The left sidebar shows 'example-instance' and 'example_database'. The main area is titled 'Instance details' for 'example-instance' with tabs for 'CREATE DATABASE', 'EDIT INSTANCE', and 'DELETE INSTANCE'. A blue circle highlights the 'DELETE INSTANCE' button. A confirmation dialog box is overlaid with the title 'Delete instance?'. It contains a warning message: 'Deleting an instance permanently removes the instance and all its databases. You cannot undo this later.' Below it is a text input field with the placeholder 'Confirm deletion by typing the instance ID below: example-instance'. The user has typed 'example-instance' into this field. At the bottom of the dialog are 'CANCEL' and 'DELETE' buttons.

Another example with more data

1. Link Google Cloud documentation to follow along with for the Python example:
 - o <https://cloud.google.com/spanner/docs/getting-started/python/>

The screenshot shows a web browser displaying the 'Getting started with Cloud Spanner in Python' documentation. The page has a header with 'Cloud Spanner > Documentation' and three stars for rating. A 'Send feedback' button is in the top right. The main content area has a title 'Objectives' and a paragraph describing the tutorial's steps. Below the paragraph is a bulleted list of objectives:

- Create a Cloud Spanner instance and database.
- Write, read, and execute SQL queries on data in the database.
- Update the database schema.

2. Clone the GitHub repository to run scripts, and browse to the correct directory:

- o `git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ git clone https://github.com/GoogleCloudPlatform/python-docs-samples.git
Cloning into 'python-docs-samples'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 31385 (delta 0), reused 4 (delta 0), pack-reused 31378
Receiving objects: 100% (31385/31385), 39.80 MiB | 42.76 MiB/s, done.
Resolving deltas: 100% (16846/16846), done.
```

- o `cd python-docs-samples/spanner/cloud-client`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd python-docs-samples/spanner/cloud-client
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ █
```

3. Create a Python environment and install dependencies:

4. `virtualenv env`

```
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ virtualenv env
created virtual environment CPython3.7.3.final.0-64 in 759ms
  creator CPython3Posix(dest=/home/leonf_info/python-docs-samples/spanner/cloud-client/env, clear=False, global=False)
  seeder FromAppData(download=False, pip=True, setuptools=True, wheel=True, via='copy', app_data_dir=/home/leonf_info/.local/share/virtualenv/seed/app-data/v1.0.1)
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
```

5. `source env/bin/activate`

```
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ source env/bin/activate
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ █
```

6. pip install -r requirements.txt

```
leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ source env/bin/activate
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ pip install -r requirements.txt
Ignoring futures: markers 'python_version < "3"' don't match your environment
Collecting google-cloud-spanner==1.16.0
  Downloading google_cloud_spanner-1.16.0-py2.py3-none-any.whl (249 kB)
    |████████| 249 kB 2.8 MB/s
Collecting google-cloud-core<2.0.dev,>=1.0.3
  Downloading google_cloud_core-1.3.0-py2.py3-none-any.whl (26 kB)
Collecting grpc-google-iam-v1<0.13dev,>=0.12.3
  Downloading grpc-google-iam-v1-0.12.3.tar.gz (13 kB)
Collecting google-api-core[grpc,grpcgcp]<2.0.0dev,>=1.14.0
```

7. Create a Spanner instance named *test-instance*:

- `gcloud spanner instances create test-instance --config=regional-us-central1 --description="Test Instance" --nodes=1`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ gcloud spanner instances create test-instance --config=regional-us-central1 --description="Test Instance" --nodes=1  
Creating instance...done.
```

8. Create a database and insert data using the Python scripts from our GitHub clone:

- o `python snippets.py test-instance -database-id example-db create_database`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ python snippets.py test-instance --database-id example-db create_database
Waiting for operation to complete...
Created database example-db on instance test-instance
```

- o `python snippets.py test-instance --database-id example-db insert_data`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ python snippets.py test-instance --database-id example-db insert_data
Inserted data.
```

9. Run a query to read the values of all columns from the Albums table:

- `gcloud spanner databases execute-sql example-db --instance=test-instance --sql='SELECT SingerId, AlbumId, AlbumTitle FROM Albums'`

```
(env) leonf_info@cloudshell:~/python-docs-samples/spanner/cloud-client (data-engineer-274519)$ gcloud spanner databases execute-sql example-db --instance=test-instance --sql='SELECT SingerId, AlbumId, AlbumTitle FROM Albums'
```

SingerId	AlbumId	AlbumTitle
1	1	Total Junk
1	2	Go, Go, Go
2	1	Green
2	2	Forever Hold Your Peace
2	3	Terrified

Check all the tables:

Albums table is **child** table of Singers table

The screenshot shows the Google Cloud Spanner interface for a database named 'example-db'. On the left, the 'OVERVIEW' tab is selected, displaying CPU utilization (mean) and operations (Read: -/s, Write: -/s). Below this, the 'Tables' section lists two tables: 'Albums' and 'Singers'. The 'Albums' table has an 'Indexes' column entry of '-' and an 'Interleaved in' column entry of 'Singers', which is circled in blue. The 'Singers' table also has an 'Indexes' column entry of '-' and an 'Interleaved in' column entry of 'Albums', which is also circled in blue. To the right, the 'Table details' page for the 'Albums' table is shown. It includes tabs for 'SCHEMA', 'INDEXES', and 'DATA', with 'DATA' being the active tab. The schema table shows three columns: 'SingerId (inherited)' (INT64, No), 'AlbumId' (INT64, No), and 'AlbumTitle' (STRING(MAX), Yes). A note at the bottom states 'Interleaved in: Singers'.

Name	Indexes	Interleaved in
Albums	-	Singers
Singers	-	Albums

Column	Type	Nullable
SingerId (inherited)	INT64	No
AlbumId	INT64	No
AlbumTitle	STRING(MAX)	Yes

Lecture: Pub/Sub Hands On

Demonstration of Cloud Pub/Sub using both the web console and gcloud command line.

Web console demonstration:

1. Create a topic called *my-topic*:

- o `gcloud pubsub topics create my-topic`
- o We can also assign access: project wide, topic wide, and subscription wide.

The screenshot shows three windows from the Google Cloud Platform Pub/Sub interface:

- Create a topic**: A modal window where the "Topic ID" field is set to "my-topic". It also includes sections for "Encryption" (Google-managed key selected) and "Topic name" (projects/data-engineer-274519/topics/my-topic).
- Add members to "my-topic"**: A modal window titled "Add members and roles for "my-topic" resource". It lists "New members" and a table of existing members with roles like "Owner" and "Viewer".
- Topics**: A list of topics in the main dashboard, showing one topic named "my-topic".

2. Create a subscription to topic *my-topic*:

- o `gcloud pubsub subscriptions create --topic my-topic mySub1`

The screenshot shows two windows from the Google Cloud Platform Pub/Sub interface:

- Topic details**: A window showing "Topic name" as projects/data-engineer-274519/topics/my-topic. It has tabs for "SUBSCRIPTIONS" and "CREATE SUBSCRIPTION".
- Add subscription to topic**: A modal window where the "Subscription ID" is set to "mySub1". Other settings include "Delivery type" (Pull selected), "Subscription expiration" (31 days), "Acknowledgement deadline" (10 seconds), and "Message retention duration" (7 days).

The screenshot shows the 'Subscription details' page for a subscription named 'mysub1'. The left sidebar lists 'Topics', 'Subscriptions' (which is selected), 'Schemas', 'Lite Topics', and 'Lite Subscriptions'. The main area displays the subscription name 'projects/data-engineer-274519/subscriptions/mysub1' and the topic name 'projects/data-engineer-274519/topics/my-topic'. Below this is a chart titled 'Unacked message count' showing a flat line at zero across the time range from 4:55 to 5:30.

3. Publish a message to your topic:

- o `gcloud pubsub topics publish my-topic --message "hello"`

The screenshot shows the 'Topic details' page for a topic named 'my-topic'. The left sidebar lists 'Topics' (selected), 'Subscriptions', 'Schemas', 'Lite Topics', and 'Lite Subscriptions'. The main area shows a message titled 'Publish message' with the topic name 'projects/data-engineer-274519/topics/my-topic' and a message body containing 'Hello'. A note indicates that export options have moved to the 'Create subscription' dropdown menu under the Subscriptions section. The message body field is highlighted with a blue circle. A red box highlights the 'Key' field in the 'Message attributes' section, which has a validation error message 'Key is required'. A blue box highlights the '+ ADD AN ATTRIBUTE' button.

4. Retrieve message with your subscription, acknowledge receipt and remove message from the queue:

- o `gcloud pubsub subscriptions pull --auto-ack mySub1`
- o (We have to use command line to do this step)

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud pubsub subscriptions pull --auto-ack mysub1
```

DATA	MESSAGE_ID	ATTRIBUTES	DELIVERY_ATTEMPT
hello	448325824473138		

5. Cancel subscription:

- o `gcloud pubsub subscriptions delete mySub1`

Topic ID ↑	Encryption	Topic name
<input checked="" type="checkbox"/> my-topic	Google-managed	projects/data-engineer-274519/topics/my-topic

6. Simulated traffic ingest:

- o Clone GitHub data to Cloud Shell (or other SDK environments), and browse to the *publish* folder:
- o `cd ~ git clone https://github.com/linuxacademy/googledataengineer`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~
leonf_info@cloudshell:~ (data-engineer-274519)$ git clone https://github.com/linuxacademy/googledataengineer
fatal: destination path 'googledataengineer' already exists and is not an empty directory.
```

- o `cd ~/googledataengineer/courses/streaming/publish`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/googledataengineer/courses/streaming/publish
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ls
download_data.sh  pubsub_pull.py  README.txt  send_sensor_data.py  sensor_obs2008.csv.gz
```

7. Create a topic called *sandiego*:

- o `gcloud pubsub topics create sandiego`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gcloud pubsub topics create sandiego
Created topic [projects/data-engineer-274519/topics/sandiego].
```

Topic ID ↑	Encryption
<input checked="" type="checkbox"/> sandiego	Google-managed

8. Create a subscription to topic sandiego:

- o `gcloud pubsub subscriptions create --topic sandiego mySub1`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gcloud pubsub subscriptions create --topic sandiego mySub1
Created subscription [projects/data-engineer-274519/subscriptions/mySub1].
```

9. Run script to download sensor data:

- o `./download_data.sh`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ vim download_data.sh
```

```
gsutil cp gs://la-gcloud-course-resources/sandiego/sensor_obs2008.csv.gz ~
```

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./download_data.sh
Copying gs://la-gcloud-course-resources/sandiego/sensor_obs2008.csv.gz...
\ [1 files] [ 34.6 MiB/ 34.6 MiB]
Operation completed over 1 objects/34.6 MiB. ↵
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ls
download_data.sh  pubsub_pull.py  README.txt  send_sensor_data.py  sensor_obs2008.csv.gz ↗
```

(Optional). If you need to authenticate a shell to ensure we have the right permissions:

- o `gcloud auth application-default login`

10. View script info:

- o `vim ./send_sensor_data.py` or use viewer of your choice

11. Run python script to simulate one hour of data per minute:

- o `./send_sensor_data.py --speedFactor=60 --project=YOUR-PROJECT-ID`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./send_sensor_data.py --spec
*****
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in  seconds or on any key.
```

If you receive an error: `google.cloud.pubsub` cannot be found OR ‘`ImportError: No module named iterator`’, run the below `pip` command to install components then try again:

- o `sudo pip install -U google-cloud-pubsub`

```

leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ sudo pip install -U google-cloud-pubsub
*****
Python 2 is deprecated. Upgrade to pip3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-pip-warning file.
The command will automatically proceed in 5 seconds or on any key.
*****
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 in January 2021. More details about Python 2 support in https://pip.pypa.io/en/latest/development/release-process/#python-2-support
Collecting google-cloud-pubsub
  Downloading google-cloud-pubsub-1.5.0-py2.py3-none-any.whl (137 kB)
    100% |██████████| 137.0MB 2.7 MB/s
Requirement already satisfied, skipping upgrade: grpc-google-iam<1.13dev,p==0.12.3 in /usr/local/lib/python2.7/dist-packages (from google-cloud-pubsub) (0.12.3)
Requirement already satisfied, skipping upgrade: enum34<1.1.6,>=1.1.4.0 in /usr/local/lib/python2.7/dist-packages (from google-cloud-pubsub) (1.1.10)
Requirement already satisfied, skipping upgrade: google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub (1.14.0)
Requirement already satisfied, skipping upgrade: googleapis-common-protos<2.0dev,>=1.6.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (1.51.0)
Requirement already satisfied, skipping upgrade: google-auth<2.0dev,>>0.4.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (1.15.0)
Requirement already satisfied, skipping upgrade: protobuf<3.4.0,>=3.0.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (44.1.0)
Requirement already satisfied, skipping upgrade: requests<3.0.0,>=2.22.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (3.12.1)
Requirement already satisfied, skipping upgrade: requests-oauthlib<1.0.0,>=0.8.0 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (2.23.0)
Requirement already satisfied, skipping upgrade: pytz in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (2020.1)
Requirement already satisfied, skipping upgrade: future<=3.2.0, python_version < "3.2" in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (3.3.0)
Requirement already satisfied, skipping upgrade: rsa<4.1,>=3.1.4 in /usr/local/lib/python2.7/dist-packages (from google-auth<2.0dev,>>0.4.0>google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (4.0)
Requirement already satisfied, skipping upgrade: certifi<2020.6.16,>=2019.11.3 in /usr/local/lib/python2.7/dist-packages (from google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (2020.6.16)
Requirement already satisfied, skipping upgrade: urllib3<1.25.0,>=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (1.25.9)
Requirement already satisfied, skipping upgrade: certifi<2017.4.17,in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (2020.4.5.1)
Requirement already satisfied, skipping upgrade: chardet<4,>=3.0.2 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (3.0.4)
Requirement already satisfied, skipping upgrade: idna<3,>=2.8 in /usr/local/lib/python2.7/dist-packages (from requests<3.0.0dev,>=2.18.0>google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (2.9)
Requirement already satisfied, skipping upgrade: pyasn1<0.1.3 in /usr/local/lib/python2.7/dist-packages (from rsa<4.1,>=3.1.4>google-auth<2.0dev,>>0.4.0>google-api-core[grpc]<1.17.0,>=1.14.0>google-cloud-pubsub) (0.4.8)

Attempting uninstall: google-api-core
  Found existing installation: google-api-core 1.17.0
  Uninstalling google-api-core-1.17.0:
    Successfully uninstalled google-api-core-1.17.0
Successfully installed google-api-core-1.16.0 google-cloud-pubsub-1.5.0

```

```

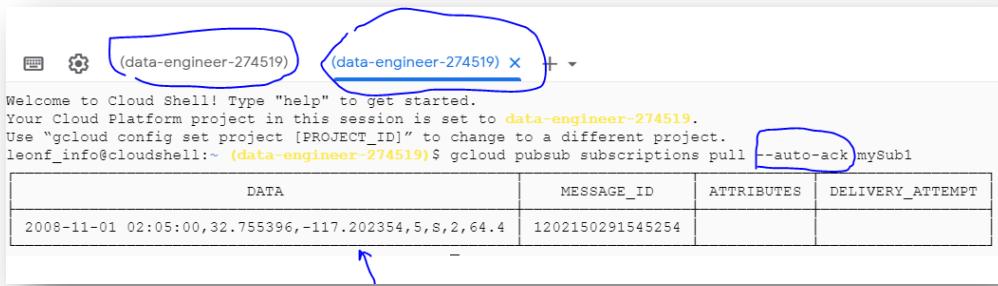
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./send_sensor_data.py --speedFactor=60 --project=data-engineer-274519
*****
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset
T

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in 5 seconds or on any key.
*****
INFO: Reusing pub/sub topic sandiego
INFO: Sending sensor data from 2008-11-01 00:00:00
INFO: Publishing 477 events from 2008-11-01 00:00:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:05:00
INFO: Sleeping 5.0 seconds

```

12. Open new Cloud Shell tab (using + symbol):

- Pull the message using the subscription *mySub1*
 - `gcloud pubsub subscriptions pull --auto-ack mySub1`



13. Create a new subscription and pull messages with it:

- `gcloud pubsub subscriptions create --topic sandiego mySub2`
- `gcloud pubsub subscriptions pull --auto-ack mySub2`

```

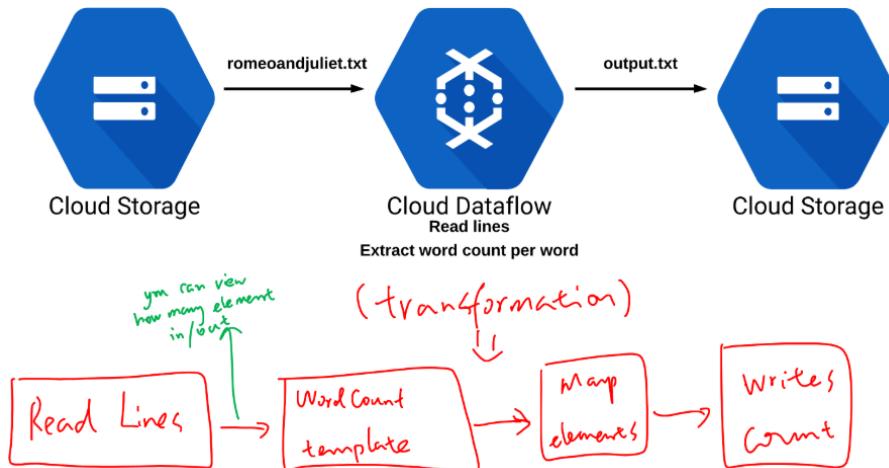
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud pubsub subscriptions create --topic sandiego mySub2
Created subscription [projects/data-engineer-274519/subscriptions/mySub2].
leonf_info@cloudshell:~ (data-engineer-274519)$ gcloud pubsub subscriptions pull --auto-ack mySub2
T

```

DATA	MESSAGE_ID	ATTRIBUTES	DELIVERY_ATTEMPT
2008-11-01 05:30:00,32.784015,-117.16195,163,s,1,77.1	1202128459831768		

Lecture: Template Hands On

In the first of our two hands-on demonstrations, we are going to use a predefined template to conduct a word count on a classic Shakespeare play using Dataflow.



1. Create a storage bucket:

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gsutil mb gs://word-count-test-1
Creating gs://word-count-test-1/...
```

2. Create a job:

- (specify the input and output location in storage buckets)

Dataflow

[Jobs](#)

[Notebooks](#)

[Create job from template](#)

Make sure all fields are correct to continue

Job name *
word-count

Must be unique among running jobs. Use lowercase letters, numbers, and hyphens (-).

Regional endpoint *
us-central1

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 *
Word Count

Batch pipeline. Reads text from Cloud Storage, tokenizes text lines into individual words, and performs frequency count on each of the words.

Required parameters

Input file(s) in Cloud Storage *
gs://dataflow-samples/shakespeare/romeoandjuliet.txt

The input file pattern Dataflow reads from. Use the example file (gs://dataflow-samples/shakespeare/kinglear.txt) or enter the path to your own using the same format: gs://<your-bucket>/<your-file>.txt

Output Cloud Storage file prefix *
gs://word-count-test-1/word-count

Path and filename prefix for writing output files. Ex: gs://<your-bucket>/counts

Temporary location *
gs://word-count-test-1/temp

Path and filename prefix for writing temporary files. Ex: gs://<your-bucket>/temp

Encryption

Google-managed key
No configuration required

Customer-managed key
Manage via Google Cloud Key Management Service

[SHOW OPTIONAL PARAMETERS](#)

[RUN JOB](#)

[Job details](#) [BACK TO OLD JOB PAGE](#) [STOP](#) [MAX TIME](#)

[HIDE SIDE PANEL](#)

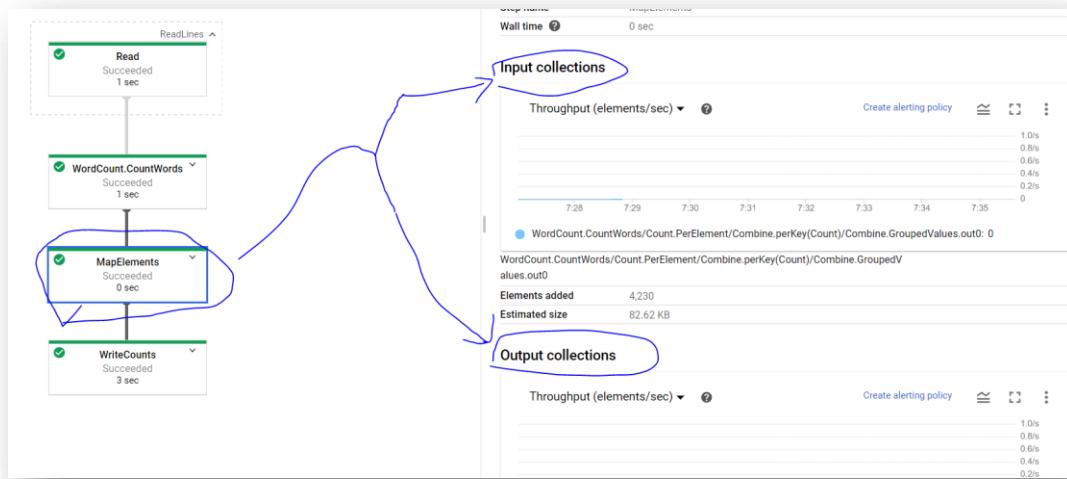
Job info

Job name	word-count
Job ID	2020-05-22_16-27-00-18307047650230711316
Job type	batch
Job status	Running
SDK version	Apache Beam SDK for Java 2.20.0
Region	us-central1
Start time	May 22, 2020 at 7:27:01 PM GMT-4
Elapsed time	6 min 33 sec
Encryption type	Google-managed key

Resource metrics

Current vCPUs	1
Total vCPU time	0.02 vCPU hr
Current memory	3.75 GB
Total memory time	0.074 GB hr
Current PD	250 GB
Total PD time	4.939 GB hr
Current SSD PD	0 B
Total SSD PD time	0.0B hr

Watch the ‘batch processing’ job:



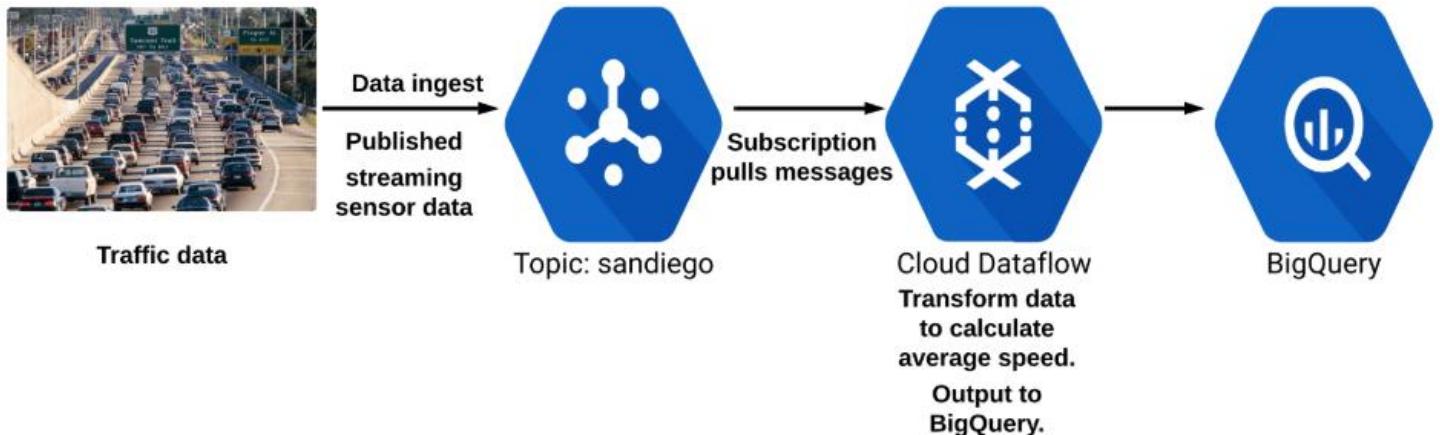
3. Check the result:

The screenshot shows the Google Cloud Storage interface. The left sidebar has 'Storage' selected. The main area shows 'Bucket details' for 'word-count-test-1'. Under 'Objects', there is a table listing three files: 'temp/' (Folder), 'word-count-00000-of-00003' (text/plain, 14.4 KB, Standard, 5/22/2020), and 'word-count-00001-of-00003' (text/plain, 13.38 KB, Standard, 5/22/2020). A blue circle highlights the 'temp/' folder entry.

The screenshot shows the Google Cloud Storage interface. The left sidebar has 'Storage' selected. The main area shows 'Object details' for the file 'word-count-00000-of-00003' in the 'word-count-test-1' bucket. The file's content is displayed as a list of words and their counts: Montague: 25, child's: 1, feather'd: 1, mourners: 1, scatter'd: 1, she: 96, importuned: 1, lisping: 1, sudden: 7, More: 7, aspired: 1, 'twill: 1, out: 48, mask: 3, believe: 3, extreme: 1.

Lecture: Streaming Ingest Pipeline Hands On

How to take our streaming ingest of traffic sensor data from the previous section, and run it through a **Dataflow** pipeline to calculate average speeds and output it to **BigQuery**.



The command line reference for what we are demonstrating is below.

1. Create **BigQuery dataset** for processing pipeline output:

- o `bq mk --dataset $DEVSHELL_PROJECT_ID:demos`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq mk --dataset $DEVSHELL_PROJECT_ID:demos
Dataset 'data-engineer-274519:demos' successfully created.
```

2. Create **Cloud Storage bucket** for Dataflow staging:

- o `gsutil mb gs://$DEVSHELL_PROJECT_ID`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gsutil mb gs://$DEVSHELL_PROJECT_ID
Creating gs://data-engineer-274519/...
```

3. Create a topic and publish messages ([sandiego example](#)):

- o `cd ~/googledataengineer/courses/streaming/publish`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/googledataengineer/courses/streaming/publish
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$
```

- o `gcloud pubsub topics create sandiego`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ gcloud pubsub topics create sandiego
Created topic [projects/data-engineer-274519/topics/sandiego].
```

- `./download_data.sh`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./download_data.sh
Copying gs://la-gcloud-course-resources/sandiego/sensor_obs2008.csv.gz...
\ [1 files] [ 34.6 MiB/ 34.6 MiB]
Operation completed over 1 objects/34.6 MiB.
```

- `sudo pip install -U google-cloud-pubsub`

```
Installing collected packages: google-api-core, google-cloud-pubsub
  Attempting uninstall: google-api-core
    Found existing installation: google-api-core 1.17.0
      Uninstalling google-api-core-1.17.0:
        Successfully uninstalled google-api-core-1.17.0
Successfully installed google-api-core-1.16.0 google-cloud-pubsub-1.5.0
```

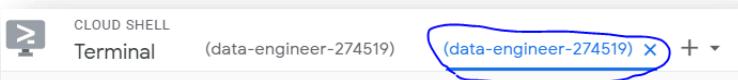
- `./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_ID`

```
leonf_info@cloudshell:~/googledataengineer/courses/streaming/publish (data-engineer-274519)$ ./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_ID
*****
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in seconds or on any key.
*****
INFO: Reusing pub/sub topic sandiego
INFO: Sending sensor data from 2008-11-01 00:00:00
INFO: Publishing 477 events from 2008-11-01 00:00:00
INFO: Sleeping 5.0 seconds
```

In previous example, we create mySub1 and mySub2.
But this time, we create Dataflow pipeline instead

4. Open a new Cloud Shell tab



- Browse to the Dataflow directory and run the script to create a pipeline, passing along **our project ID, storage bucket, and Average Speeds file** to construct the pipeline.
- `cd ~/googledataengineer/courses/streaming/process/sandiego`
**(we use the 'run_oncloud.sh' to kick off the pipeline
It will open some Java file to calculate the Average)**

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/googledataengineer/courses/streaming/process/sandiego
leonf_info@cloudshell:~/googledataengineer/courses/streaming/process/sandiego (data-engineer-274519)$ ls
create_cbt.sh delete_cbt.sh install_quickstart.sh pom.xml run_oncloud.sh src
```

- o `./run_oncloud.sh $DEVSHELL_PROJECT_ID $DEVSHELL_PROJECT_ID AverageSpeeds`

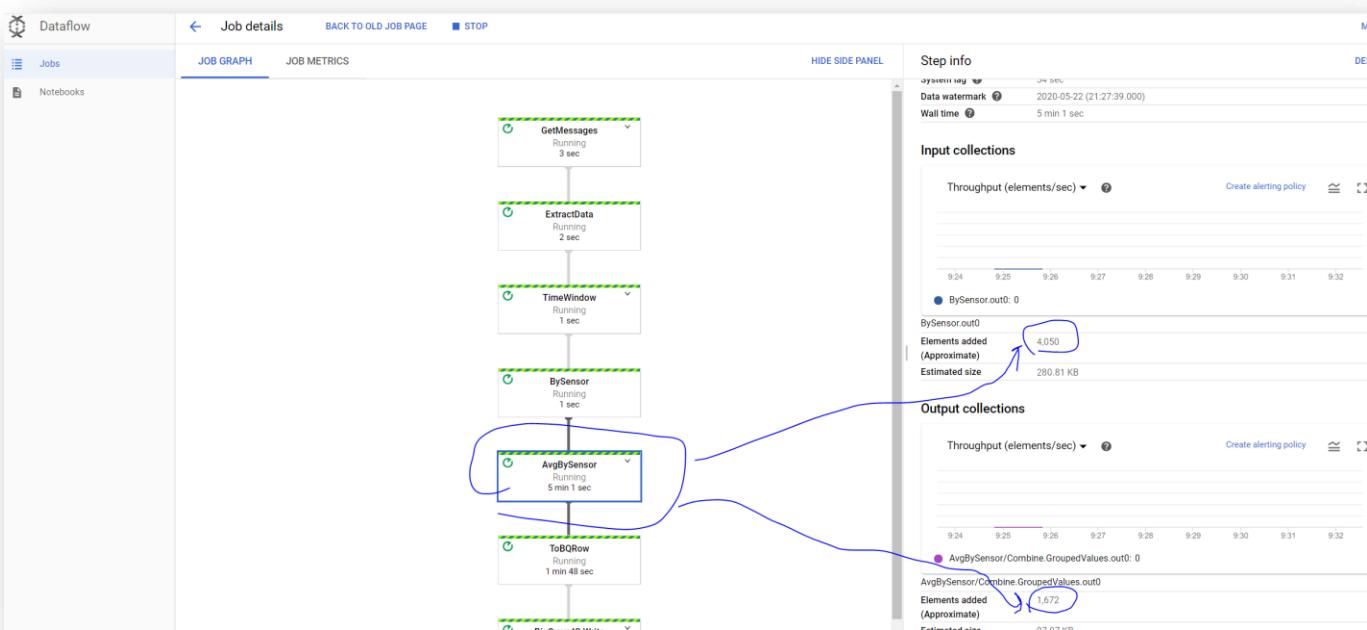
```
> gcloud dataflow jobs --project=data-engineer-274519 cancel --region=us-central1 2020-05-22_18_23_35-10788448147967333524
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:12 min
[INFO] Finished at: 2020-05-22T21:23:36-04:00
[INFO] -----
```

Now, the pipeline is built successfully.

Go back to Dataflow to confirm it:

The screenshot shows the Google Cloud Dataflow web interface. In the top navigation bar, 'Dataflow' is selected. Below it, there are two tabs: 'Jobs' (which is active) and 'Notebooks'. On the right side of the screen, there is a search bar labeled 'Filter jobs' and a list of jobs. One job, 'averagespeeds-leonf0info-0523012323-e4171a92', is highlighted with a blue oval and a blue arrow pointing down to the 'JOB GRAPH' section. Other jobs listed include 'word-count', 'cloud-dataprep-random-sample-5407245-by-leonfinfo', 'cloud-dataprep-first-rows-sample-5407243-by-leonfinfo', and 'averagespeeds-leonf0info-0501004131-933899c'.

The transform is happening:



Run some SQL on BigQuery:

(Even the data is in streaming buffer, we can see nothing in BigQuery, we can still write SQL query)

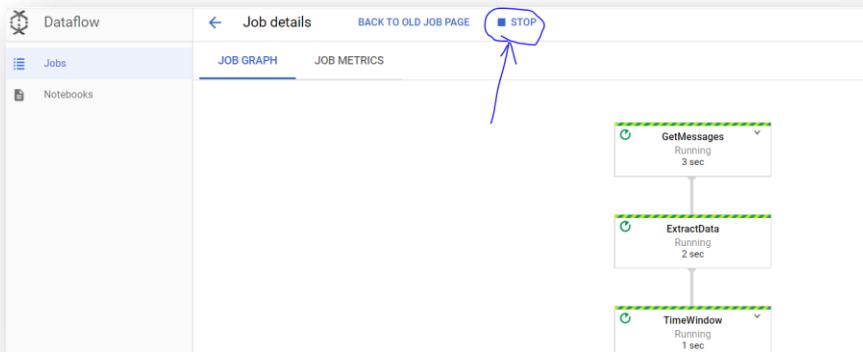
The screenshot shows the BigQuery web interface. On the left, there's a sidebar with links like 'Query history', 'Saved queries', 'Job history', 'Transfers', 'Scheduled queries', 'Reservations', and 'BI Engine'. Below that is a 'Resources' section with a 'data-engineer-274519' project selected, showing 'demos' and 'average_speeds'. The main area is the 'Query editor' with the following SQL code:

```
1 SELECT * FROM `data-engineer-274519.demos.average_speeds`  
2 ORDER BY timestamp DESC LIMIT 100
```

The 'Query results' section shows the output of the query:

Row	timestamp	latitude	longitude	highway	direction	lane	speed	sensord
1	2020-05-23 01:26:06.279 UTC	32.671747	-117.111237	5	S	4	62.05000000000000	32.671747,
2	2020-05-23 01:26:06.260 UTC	32.743125	-117.184141	5	S	4	64.0	32.743125,
3	2020-05-23 01:26:06.259 UTC	32.848811	-117.235604	5	S	4	64.55	32.848811,
4	2020-05-23 01:26:06.256 UTC	32.833981	-117.233282	5	S	2	76.25	32.833981,

Stop the job:



Go to PubSub to delete the **Job** and the **subscriptions**:

The screenshot shows the Pub/Sub 'Topics' page. On the left, there's a sidebar with 'Pub/Sub', 'Topics', 'Subscriptions', 'Snapshots', 'Lite Topics', and 'Lite Subscriptions'. The main area shows a table with one row:

Topic ID	Encryption
sandiego	Google-managed

Lecture: Configure Dataproc Cluster and Submit Job – Part 1

- ❖ Creating a **Dataproc cluster** and **submitting a sample Spark job** to the cluster.
- ❖ Google-provided Dataproc initialization scripts:
<https://console.cloud.google.com/storage/browser/dataproc-initialization-actions/>
- ❖ Reference for example job submission:
<https://cloud.google.com/dataproc/docs/quickstarts/quickstart-console>
- ❖ Creating a dataproc cluster: `gcloud dataproc clusters create [cluster_name] --zone [zone_name]`

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gsutil mb -l us-central1 gs://$DEVSHELL_PROJECT_ID-data
Creating gs://data-engineer-274519-data/...
ServiceException: 409 Bucket data-engineer-274519-data already exists.
```

Lecture: Configure Dataproc Cluster and Submit Job – Part 2

This is the second half of our hands-on demonstration. Command line notes of actions taken in this lesson are below:

1. Updating a cluster with new workers/preemptible machines:
 - o `gcloud dataproc clusters update [cluster_name] --num-workers [#] --num-preemptible-workers [#]`
 - o SOCKS proxy configuration
2. From the local machine, SSH to the master to enable port forwarding:
 - o `gcloud compute ssh master-host-name --project=project-id --zone=master-host-zone -- -D 1080 -N`
3. Open new terminal window and launch the web browser with parameters (varies by OS/browser):
 - o `"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" --proxy-server="socks5://localhost:1080" --host-resolver-rules="MAP * 0.0.0.0 , EXCLUDE localhost" --user-data-dir=/tmp/cluster1-m`
 - Browse to `http://[master]:port`
 - 8088 - Hadoop
 - 9870 - HDFS
 - Using Cloud Shell (must use for each port)
 - o `gcloud compute ssh master-host-name --project=project-id --zone master-host-zone -- -4 -N -L port1:master-host-name:port2`
4. Use Web Preview to choose port (8088/9870).

Lecture: Interacting with BigQuery

Create a dataset:

The screenshot shows the BigQuery web interface. On the left, there's a sidebar with links like 'Query history', 'Saved queries', 'Job history', 'Transfers', 'Scheduled queries', 'Reservations', 'BI Engine', 'Resources' (with '+ ADD DATA'), and a search bar. The main area has a 'Query editor' tab with a 'COMPOSE NEW QUERY' button. Below it is a 'Datasets and tables available' section. A modal window titled 'Create dataset' is open on the right. It contains fields for 'Dataset ID' (set to 'new_dataset'), 'Data location (Optional)' (set to 'United States (US)'), 'Default table expiration' (radio buttons for 'Never' and 'Number of days after table creation'), and an 'Encryption' section with options for 'Google-managed key' (selected) and 'Customer-managed key'. A blue arrow points from the 'CREATE DATASET' button in the modal to the 'CREATE DATASET' button in the main interface.

Or using bq command:

This screenshot shows the BigQuery web interface and a Cloud Shell terminal. In the Cloud Shell, a user runs the command `bq mk --dataset data-engineer-274519:new_dataset`. The output shows the dataset was successfully created. A blue arrow points from the terminal output back to the 'CREATE DATASET' button in the BigQuery interface above.

Setting different permissions (IAM):

The screenshot shows the BigQuery web interface for the 'new_dataset' dataset. On the left, there's a sidebar with 'data-engineer-274519' and 'demos'. The main area shows 'data-engineer-274519:new_dataset' with a 'COPY DATA' button. A blue arrow points from the 'COPY DATA' button to the 'Dataset permissions' section on the right. This section includes a 'Dataset permissions' table and a 'DATASET PERMISSIONS' tab. Under 'DATASET PERMISSIONS', there's a 'Add members' field with a dropdown menu showing roles: 'Owner', 'Editor', 'Viewer', 'BigQuery Admin', 'BigQuery Data Editor', 'BigQuery Data Owner', 'BigQuery Data Viewer', 'BigQuery Metadata Viewer', 'BigQuery Data Editor (1 member)', 'BigQuery User', 'Security Admin', and 'Security Reviewer'. Other tabs include 'AUTHORIZED VIEWS'.

Add public dataset:

The screenshot shows the BigQuery web interface. In the top navigation bar, there is a dropdown for the project "data-engineer-274519". Below it, under the "demos" folder, a new dataset named "new_dataset" is selected. The main panel displays the details for "data-engineer-274519:new_dataset". A blue circle highlights the "Labels" section, which is currently set to "None". A blue arrow points from the "Labels" section to the "Labels" input field.

Write Query (Standard SQL mode):

This screenshot shows the BigQuery UI with a focus on a specific table. On the left, the sidebar lists datasets and tables, with "Kyc" in the "new_dataset" folder highlighted by a blue circle. The main area shows an "Unsaved query" editor with the following SQL code:

```
1 SELECT
2   acct_type,
3   acct_rep
4 FROM
5   `data-engineer-274519.new_dataset.Kyc`
6 LIMIT
7   1000
```

A blue arrow points from the "SQL dialect" dropdown menu to the "Standard" option, which is selected. Another blue arrow points from the "Processing location" dropdown to "United States (US)". The "QUERY TABLE" button is also circled in blue. The table schema is shown below:

Field name	Type	Mode	Policy tags	Description
client_id	STRING	NULLABLE		

This screenshot shows the BigQuery UI with a focus on running a query and viewing its results. The "Unsaved query" editor contains the same SQL code as before:

```
1 SELECT
2   acct_type,
3   acct_rep
4 FROM
5   `data-engineer-274519.new_dataset.Kyc`
6 WHERE client_age=52.0
7 LIMIT
8   1000
```

A blue arrow points from the "More" dropdown menu to the "Format" option. The "Query editor" panel shows the valid query and its execution status. The "Query results" panel shows the output of the query:

Row	acct_type	acct_rep
1	RSP	05G2

Searching multiple tables with wildcards

Query across multiple, similarly named tables

- FROM `project.dataset.table_prefix*`

Filter further in WHERE clause

Query using bq command line format:

```
leonf.info@cloudshell:~ (data-engineer-274519)$ bq query --use_legacy_sql=false 'select acct_type, acct_rep from `data-engineer-274519.n  
ew_dataset.Kyc`'  
  
Waiting on bqjob_r38e00fa8ac7a651f_00000172483f51e2_1 ... (0s) Current status: DONE  
+-----+-----+  
| acct_type | acct_rep |  
+-----+-----+  
| RSP      | TFW1    |  
| Cash     | 05G2    |  
| RSP      | 05G2    |  
| RSP      | 05G2    |  
| RSP      | 05G2    |  
+-----+-----+
```

Create a VIEW:

The screenshot shows the BigQuery web interface. A query is being run, and the results are displayed in a table. A blue arrow points from the 'Save view' button in the main toolbar to a 'Save view' dialog box. Another blue circle highlights the 'Dataset name' field in the dialog box.

Query results

Row	acct_type	acct_rep
1	RSP	TFW1
2	Cash	05G2
3	RSP	05G2
4	RSP	05G2
5	RSP	05G2

Save view

The destination dataset for a saved view must be in the same region as the source, otherwise a 'Dataset not found' error will be returned.

Destination
Project name: data-engineer
Dataset name: new_dataset

Table name: kyc_views

CANCEL SAVE

If we query from the VIEW, it would be more much faster:

Processing location: US

Run Save query Save view Schedule query More

This query will process 5 rows

kyc_views

QUERY VIEW COPY VIEW DELETE VIEW

Schema Details

Field name	Type	Mode	Policy tags ⓘ	Description
acct_type	STRING	NULLABLE		
acct_rep	STRING	NULLABLE		

Edit schema

Cache a Query (per user only, somebody else can not use it at the same time):

Query settings

Query engine

BigQuery engine

Cloud Dataflow engine
Deploy your data processing pipelines on the Cloud Dataflow service.

Destination

Save query results in a temporary table

Set a destination table for query results

Project name: data-engineer Dataset name: demos

Table name: Letters, numbers, underscores, and template system characters allowed

Destination table write preference

Write if empty

Append to table

Overwrite table

Results size

Allow large results (no size limit)

Resource management

Job priority

Interactive

Batch

Cache preference

Use cached results

User Defined Functions (UDF)

- Combine SQL code with JavaScript/SQL functions
- Combine SQL queries with programming logic
- Allow much more complex operations (loops, complex conditionals)
- WebUI only usable with Legacy SQL



Lecture: Load and Export Data

Prepare data in **Storage Bucket**:

The screenshot shows the 'Bucket details' page for 'bq-demo-1'. The left sidebar has 'Storage' selected. The main area shows the 'Objects' tab with a file named 'kyc.csv' listed. The file is 846 B in size, has type 'application/vnd.ms-excel', and is in 'Standard' storage class. A blue circle highlights the 'Storage' tab in the sidebar, another highlights the 'bq-demo-1' bucket name, and a third highlights the 'kyc.csv' file.

Create a new table in **BigQuery**:

The screenshot shows the 'Dataset info' page for 'data-engineer-274519.new_dataset'. The left sidebar shows datasets like 'demos' and 'new_dataset'. The main area shows the dataset's ID, creation date, and location. A blue arrow points from the 'CREATE TABLE' button at the top right towards the 'Labels' section. Another blue circle highlights the 'CREATE TABLE' button.



Lecture: Load and Export Data (Using Wildcard)

Prepare data in **Storage Bucket**:

Name	Size	Type	Storage
accommmodation_2018.csv	4.78 KB	application/vnd.ms-excel	Standard
accommmodation_2019.csv	4.78 KB	application/vnd.ms-excel	Standard
accommmodation_2020.csv	4.78 KB	application/vnd.ms-excel	Standard
kyc.csv	846 B	application/vnd.ms-excel	Standard

Create a new table in **BigQuery** and load multiple files into **ONE** table:

The screenshot shows the BigQuery web interface. On the left, the 'Resources' sidebar lists datasets: 'data-engineer-274519' (containing 'demos' and 'new_dataset') and 'KYC'. The 'new_dataset' dataset is selected. Inside 'new_dataset', there is a table named 'all_years' which is circled in blue. The main area shows the 'all_years' table with the following schema and data:

Row	int64_field_0	string_field_1	string_field_2	int64_field_3	int64_field_4	double_field_5	string_field_6
1	5	Homy Quiet Shack	Paris	50	1	1.1	cottage
2	5	Homy Quiet Shack	Paris	50	1	1.1	cottage
3	5	Homy Quiet Shack	Paris	50	1	1.1	cottage
4	11	Homy Quiet Shanty	Melbourne	50	1	2.8	cottage
5	11	Homy Quiet Shanty	Melbourne	50	1	2.8	cottage
6	11	Homy Quiet Shanty	Melbourne	50	1	2.8	cottage
7	36	Comfy Private Shanty	NYC	80	1	3.7	cottage
8	36	Comfy Private Shanty	NYC	80	1	3.7	cottage
9	36	Comfy Private Shanty	NYC	80	1	3.7	cottage

- When Exporting, the maximum size is 1GB. If exporting is > 1GB, then use wildcard

The screenshot shows the BigQuery web interface with the 'Table Info' panel open. The 'Table ID' is 'bigquery-public-data:baseball.games_wide'. The 'Storage' tab is circled in blue. An export dialog is open, showing the 'Export to Google Cloud Storage' options. The 'Export format' is set to 'CSV' and 'Compression' is set to 'None'. The 'Google Cloud Storage URI' field contains 'gs://pw-bq-demo/baseball/games_wide*.csv' and is also circled in blue. The 'OK' button is visible at the bottom of the dialog.

Below the 'Table Info' panel, the 'Bucket details' for 'pw-bq-demo' are shown. The 'Storage' tab in the sidebar is circled in blue. The bucket contains objects named 'games_post_wide.csv' and 'games_post_wide2.csv'.

Lecture: Load and Export Data (Using Command Line)

Loading Data:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq load --source_format=CSV new_dataset.kyc_from_command_line gs://bq-demo-1/accommodation_2018.csv
BigQuery error in load operation: Error processing job 'data-engineer-274519:bqjob_r422ea8856ce64e31_000001724d03d644_1': No schema specified on job or table.
```

We need Schema:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq load --source_format=CSV new_dataset.accom_from_command_line gs://bq-demo-1/accommodation_2018.csv
Index:INTEGER,Name:STRING,Location:STRING,Size:INTEGER,Room:INTEGER,Price:FLOAT,Type:STRING
Waiting on bqjob_rdb563f4599a2fef_000001724d0782ef_1 ... (0s) Current status: DONE
```

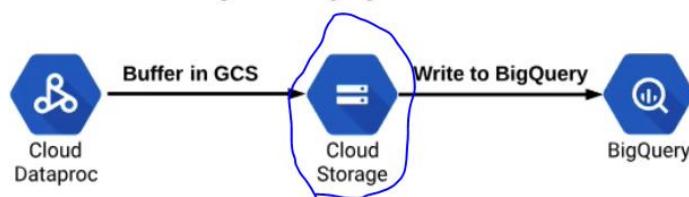
Exporting Data:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bq extract 'data-engineer-274519:new_dataset.all_years' gs://bq-demo-1/all_years_extract.csv
Waiting on bqjob_r1a6d629662014360_000001724d159bee_1 ... (0s) Current status: DONE
```

The screenshot shows the 'Bucket details' page for 'bq-demo-1'. On the left, a sidebar menu has 'Storage' circled in blue. The main area shows the bucket name 'bq-demo-1' and tabs for 'Objects', 'Overview', 'Permissions', and 'Bucket Lock'. Below is a file list with a search bar and a 'Filter by prefix...' input. The files listed are: accommodation_2018.csv, accommodation_2019.csv, accommodation_2020.csv, all_years_extract.csv (circled in blue), and kyc.csv.

Connecting to/from other Google Cloud services

- Dataproc - Use BigQuery connector (installed by default), job uses Cloud Storage for staging





Lecture: Streaming Insert Example

Revisit our streaming Dataflow/PubSub pipeline and take a closer look at the end result, which is a streaming insert into BigQuery. For easy setup, the below commands and scripts will quickly create your pipeline so we can focus on BigQuery, as well as cleanup when finished.

Quick setup:

```
cd
```

```
gsutil cp -r gs://gcp-course-exercise-scripts/data-engineer/* .
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ gsutil cp -r gs://gcp-course-exercise-scripts/data-engineer/* .
Copying gs://gcp-course-exercise-scripts/data-engineer/streaming-cleanup.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/streaming-insert.sh...
- [2 files][ 1.4 KiB/ 1.4 KiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/1-local-train.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/2-single-instance-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/3-distributed-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/4-hyperparameter-distributed.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/5-create-model-send-predictions.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/AI Platform.pdf...
Copying gs://gcp-course-exercise-scripts/data-engineer/ai-platform/AI-Platform-Commands.md...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/1-local-train.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/2-local-train-dist.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/3-single-instance-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/4-distributed-trainer.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/5-hyperparameter-distributed.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/6-create-model-send-predictions.sh...
Copying gs://gcp-course-exercise-scripts/data-engineer/cloud-ml_engine/7-superfun-script.sh...
\ [16 files] [133.5 KiB/133.5 KiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Operation completed over 16 objects/133.5 KiB.
```

leonf_info@cloudshell:~ (data-engineer-274519)\$ ls

ai-platform	googledataengineer	python-docs-samples	requests.json	training-data-analyst
cloud-ml_engine	master.zip	qie.jpg	streaming-cleanup.sh	
cloudml-samples-master	output.txt	README-cloudshell.txt	streaming-insert.sh	

```
#!/bin/bash
leonf_info@cloudshell:~ (data-engineer-274519)$ vim streaming-insert.sh
# Enable Dataflow and Pub/Sub API's if not already enabled
gcloud services enable dataflow.googleapis.com
gcloud services enable pubsub.googleapis.com
# Go to home directory and clone GitHub data
cd ~
git clone https://github.com/GoogleCloudPlatform/training-data-analyst
# Create BigQuery dataset if it does not already exist
echo **Creating BigQuery dataset**
bq mk --dataset $DEVSHELL_PROJECT_ID:demos
# Create Cloud Storage Bucket for Dataflow staging
echo **Creating staging bucket**
gsutil mb gs://$DEVSHELL_PROJECT_ID
# Create Pub/Sub topic named San Diego
echo **Creating Pub/Sub topic**
gcloud pubsub topics create sandiego
# Go to Dataflow job directory and run script to subscribe to topic and create streaming pipeline
echo **Creating Dataflow streaming pipeline**
cd ~/training-data-analyst/courses/streaming/process/sandiego
./run_oncloud.sh $DEVSHELL_PROJECT_ID $DEVSHELL_PROJECT_ID AverageSpeeds
# Install python dependencies for Pub/Sub
cd ~/training-data-analyst/courses/streaming/publish
sudo pip install -U google-cloud-pubsub
# Download sample sensor data and begin simulated stream
echo **Starting sensor stream**
./download_data.sh
./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_ID
```

bash streaming-insert.sh

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bash streaming-insert.sh
fatal: destination path 'training-data-analyst' already exists and is not an empty directory.
**Creating BigQuery dataset**
BigQuery error in mk operation: Dataset 'data-engineer-274519:demos' already exists.
**Creating staging bucket**
Creating gs://data-engineer-274519/...
ServiceException: 409 Bucket data-engineer-274519 already exists.
**Creating Pub/Sub topic**
Created topic [projects/data-engineer-274519/topics/sandiego].
**Creating Dataflow streaming pipeline**
Launching com.google.cloud.training.dataanalyst.sandiego.AverageSpeeds project=data-engineer-274519 bucket=data-engineer-274519
[INFO] Error stacktraces are turned on.
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.google.cloud.training.dataanalyst.sandiego:sandiego >-----
[INFO] Building sandiego [1.0.0,2.0.0]
[INFO] -----[ jar ]----- INFO: Reusing pub/sub topic sandiego
Downloading from apache.snapshots: https://repository.apache.org/c/ INFO: Sending sensor data from 2008-11-01 00:00:00
Downloading from central: https://repo.maven.apache.org/maven2/io/ INFO: Publishing 477 events from 2008-11-01 00:00:00
Downloaded from central: https://repo.maven.apache.org/maven2/io/ INFO: Sleeping 5.0 seconds
Downloading from apache.snapshots: https://repository.apache.org/c/ INFO: Publishing 477 events from 2008-11-01 00:05:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:10:00
INFO: Sleeping 5.0 seconds
INFO: Publishing 477 events from 2008-11-01 00:15:00
INFO: Sleeping 5.0 seconds
```

Go to Dataflow, can see the streaming pipeline:

The screenshot shows the Google Cloud Dataflow interface. On the left, there's a sidebar with 'Jobs' and 'Notebooks'. The main area displays a table of jobs. Two rows are visible: one job named 'averagespeeds-leonfinfo-0525184849-3dec1a15' which is 'Running', and another job named 'averagespeeds-leonfinfo-0523012323-e4171a92' which is 'Canceled'. Both jobs are of type 'Streaming'.

Now jump to BigQuery:

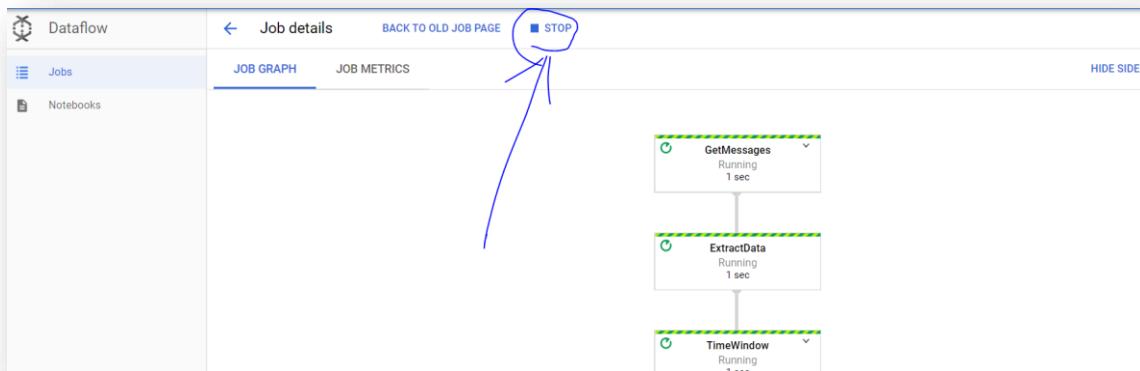
The screenshot shows the Google Cloud BigQuery interface. On the left, the 'data-engineer-274519' project navigation pane is open, showing datasets like 'demos' and 'average_speeds'. The 'average_speeds' dataset is selected. In the center, the 'average_speeds' table is shown in the 'QUERY TABLE' view. The table schema is displayed with columns: timestamp, latitude, longitude, highway, direction, lane, speed, and sensorid. Below the schema, six rows of sample data are shown. The right side of the screen shows the 'Query editor' with the following SQL query:

```
1 SELECT
2 *
3 FROM
4 `data-engineer-274519.demos.average_speeds`
5
6 WHERE
7 lane=1
8 LIMIT
9 1000
```

The 'Query results' section shows the completed query with 123.7 KB processed. An arrow points from the 'average_speeds' table in the schema view up towards the 'QUERY TABLE' view.

Clean up:

Manually stop the Dataflow job:



Run **streaming-cleanup.sh**:

```
leonf_info@cloudshell:~ (data-engineer-274519)$ ls
ai-platform      googledataengineer  python-docs-samples    requests.json      training-data-analyst
cloud-ml_engine   master.zip        qie.jpg            streaming-cleanup.sh
cloudml-samples-master output.txt    README-cloudshell.txt  streaming-insert.sh
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ vim streaming_cleanup.sh
```

```
#!/bin/bash

gcloud pubsub topics delete sandiego
gsutil rm -rf gs://$DEVSHELL_PROJECT_ID
bq rm -r -f -d $DEVSHELL_PROJECT_ID:demos
~
```

bash streaming-cleanup.sh

```
leonf_info@cloudshell:~ (data-engineer-274519)$ bash streaming-cleanup.sh
Deleted topic [projects/data-engineer-274519/topics/sandiego].
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-m/dataproc-initialization-script-0_output#1590342723259981...
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-m/dataproc-startup-script_output#1590342718630439...
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-w-0/dataproc-initialization-script-0_output#1590342775630606...
Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-w-0/dataproc-startup-script_output#1590342643266592...
/ [4 objects]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m rm ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Removing gs://data-engineer-274519/google-cloud-dataproc-metainfo/0c0b8cb5-6018-4839-a0dc-73b022d5598e/cluster-1-w-1/dataproc-initialization-script-0_output#1590342775630617
```

Lecture: AI Platform Hands On Part 1

- Go over the process of training a pre-packaged machine learning model on AI Platform, deploying the trained model, and running predictions against it.
- All commands used in these two demonstrations are below.
- If you want a set of scripts to automate much of the process, the **scripts** and a **PDF file** of the commands we used can be found at the following public link:
<https://console.cloud.google.com/storage/browser/gcp-course-exercise-scripts/data-engineer/ai-platform>
- If you want to download all scripts and files in a terminal environment, use the below command:

```
gsutil cp gs://gcp-course-exercise-scripts/data-engineer/ai-platform/* .
```

Set region environment variable

```
REGION=us-central1
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ REGION=us-central1
```

Download and unzip ML github data for demo

```
wget https://github.com/GoogleCloudPlatform/cloudml-samples/archive/master.zip
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ wget https://github.com/GoogleCloudPlatform/cloudml-samples/archive/master.zip
--2020-05-25 21:32:45-- https://github.com/GoogleCloudPlatform/cloudml-samples/archive/master.zip
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/GoogleCloudPlatform/cloudml-samples/zip/master [following]
--2020-05-25 21:32:45-- https://codeload.github.com/GoogleCloudPlatform/cloudml-samples/zip/master
Resolving codeload.github.com (codeload.github.com)... 140.82.113.10
Connecting to codeload.github.com (codeload.github.com)|140.82.113.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: 'master.zip.1'

master.zip.1                                              [=====] 24.18M 7.63MB/s   in 3.2s

2020-05-25 21:32:48 (7.63 MB/s) - 'master.zip.1' saved [25353805]
```

```
unzip master.zip
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ unzip master.zip
Archive: master.zip
54c14e0edc940a007cf045977c9e0a045a15d419
replace cloudml-samples-master/.github/ISSUE_TEMPLATE/bug_report.md? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ ls
ai-platform           googledataengineer  output.txt      README-cloudshell.txt  streaming-insert.sh
cloud-ml-engine       master.zip        python-docs-samples requests.json    training-data-analyst
Cloudml-samples-master master.zip.1     qie.jpg        streaming-cleanup.sh
```

Navigate to the cloudml-samples-master > census > estimator directory.

ALL Commands must be run from this directory

```
cd ~/cloudml-samples-master/census/estimator
```

```
leonf_info@cloudshell:~ (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ █
```

Develop and validate trainer on local machine

Get training data from public GCS bucket

1. Create a directory called 'data'
2. Download the dataset

```
mkdir data
gsutil -m cp gs://cloud-samples-data/ml-engine/census/data/* data/
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil -m cp gs://cloud-samples-data/ml-engine/census/data/* data/
Copying gs://cloud-samples-data/ml-engine/census/data/census.test.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/test.json...
Copying gs://cloud-samples-data/ml-engine/census/data/adult.data.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/census.train.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/test.csv...
Copying gs://cloud-samples-data/ml-engine/census/data/adult.test.csv...
\ [6/6 files] [ 10.7 MiB/ 10.7 MiB] 100% Done
Operation completed over 6 objects/10.7 MiB.
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd data
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator/data (data-engineer-274519)$ ls
adult.data.csv  adult.test.csv  census.test.csv  census.train.csv  test.csv  test.json
```

Set path variables for local file paths

These will change later when we use AI Platform

```
TRAIN_DATA=$(pwd)/data/adult.data.csv  
EVAL_DATA=$(pwd)/data/adult.test.csv
```

Run sample requirements.txt to ensure we're using same version of TF as sample

```
sudo pip install -r ~/cloudml-samples-master/census/requirements.txt
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator/data (data-engineer-274519)$ cd .. ↵  
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ sudo pip install -r ~/cloudml-samples-master/census/requirements.txt  
*****  
Python 2 is deprecated. Upgrade to pip3 as soon as possible.  
See https://cloud.google.com/python/docs/python2-sunset  
To suppress this warning, create an empty ~/.cloudshell/no-pip-warning file.  
The command will automatically proceed in 5 seconds or on any key.  
*****  
DEPRECATION: Python 2.7 reached the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 is no longer maintained. pip 21.0 will drop support for Python 2.7 on January 1st, 2020. pip 21.0 will drop support for this on January 1st, 2020.  
https://pip.pypa.io/en/latest/development/release-process/#python-2-support  
Collecting tensorflow<2,>=1.15.*  
  Downloading tensorflow-1.15.0-cp27-cp27mu-manylinux2010_x86_64.whl (412.3 MB)  
    |████████████████████████████████| 412.3 MB 2.6 kB/s
```

Run a local trainer

Specify output directory, set as variable

```
MODEL_DIR=output
```

Best practice is to delete contents of output directory in case data remains from previous training run

```
rm -rf $MODEL_DIR/*
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ MODEL_DIR=output  
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ rm -rf $MODEL_DIR/*
```

Run local training using gcloud (not on AI Platform)

```
gcloud ai-platform local train --module-name trainer.task --package-path trainer/ \
--job-dir $MODEL_DIR -- --train-files $TRAIN_DATA --eval-files $EVAL_DATA \
--train-steps 1000 --eval-steps 100
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform local train --module-name tra
iner.task --package-path trainer/ \
> --job-dir $MODEL_DIR -- --train-files $TRAIN_DATA --eval-files $EVAL_DATA \
> --train-steps 1000 --eval-steps 100
*****
Python 2 is deprecated. Upgrade to Python 3 as soon as possible.
See https://cloud.google.com/python/docs/python2-sunset

To suppress this warning, create an empty ~/.cloudshell/no-python-warning file.
The command will automatically proceed in seconds or on any key.
*****
WARNING:tensorflow:From /home/leonf_info/cloudml-samples-master/census/estimator/trainer/task.py:191: The name tf.logging.set_verbosity
is deprecated. Please use tf.compat.v1.logging.set_verbosity instead.

INFO:tensorflow:TF_CONFIG environment variable: {'environment': 'cloud', 'cluster': {}, 'job': {'args': ['--train-files', '/home/
leonf_info/cloudml-samples-master/census/estimator/data/adult.data.csv', '--eval-files', '/home/leonf_info/cloudml-samples-master/cens
us/estimator/data/adult.test.csv', '--train-steps', '1000', '--eval-steps', '100', '--job-dir', 'output'], 'job_name': 'trainer
task', 'task': {}}}
Model dir output
INFO:tensorflow:Using config: {'_save_checkpoints_secs': 600, '_num_ps_replicas': 0, '_keep_checkpoint_max': 5, '_task_type': 'worker',
'_global_id_in_cluster': 0, '_is_chief': True, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec object at 0x7f1096d3d
bd0>, '_model_dir': 'output', '_protocol': None, '_save_checkpoints_steps': None, '_keep_checkpoint_every_n_hours': 10000, '_service': N
one, '_session_config': allow_soft_placement: true
graph_options {
    rewrite_options {
        meta_optimizer_iterations: ONE
    }
}
, '_tf_random_seed': None, '_save_summary_steps': 100, '_device_fn': None, '_session_creation_timeout_secs': 7200, '_experimental_distribu
tute': None, '_num_worker_replicas': 1, '_task_id': 0, '_log_step_count_steps': 100, '_experimental_max_worker_delay_secs': None, '_eval
uation_master': '', '_eval_distribute': None, '_train_distribute': None, '_master': ''}
INFO:tensorflow:Not using Distribute Coordinator.
```

```
WARNING:tensorflow:Export includes no default signature!
INFO:tensorflow:Restoring parameters from output/model.ckpt-1000
INFO:tensorflow:Assets added to graph.
INFO:tensorflow:No assets to write.
INFO:tensorflow:SavedModel written to: output/export/census/temp-1590513459/saved_model.pb
INFO:tensorflow:Loss for final step: 16.124683.
```

```
# Viewing results in tensorboard
```

```
tensorboard --logdir=$MODEL_DIR --port=8080
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ tensorboard --logdir=$MODEL_DIR --port=8080
W0526 13:22:25.6994485 140590387840768 plugin_event_accumulator.py:294] Found more than one graph event per run, or there was a metagraph
containing a graph_def, as well as one or more graph events. Overwriting the graph with the newest event.
W0526 13:22:25.699318 140590387840768 plugin_event_accumulator.py:302] Found more than one metagraph event per run. Overwriting the meta
graph with the newest event.
TensorBoard 1.15.0 at http://cs-6000-devshell-vm-73658e62-4b5d-4301-afff-7f546f32260f:8080/ (Press CTRL+C to quit)
```



Run trainer on GCP AI Platform - single instance

Create regional Cloud Storage bucket used for all output and staging (make-bucket command)

```
REGION=us-central1 /  
gsutil mb -l $REGION gs://$DEVSHELL_PROJECT_ID-aip-demo
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ REGION=us-central1  
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil mb -l $REGION gs://$DEVSHELL_PROJECT_ID-aip-demo  
Creating gs://data-engineer-274519-aip-demo/...
```

Upload training and test/eval data to bucket

```
cd ~/cloudml-samples-master/census/estimator  
gsutil cp -r data gs://$DEVSHELL_PROJECT_ID-aip-demo/data
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator  
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil cp -r data gs://$DEVSHELL_PROJECT_ID-aip-demo/data  
Copying file://data/test.json [Content-Type=application/json]...  
Copying file://data/census.test.csv [Content-Type=text/csv]...  
Copying file://data/adult.data.csv [Content-Type=text/csv]...  
Copying file://data/test.csv [Content-Type=text/csv]...  
- [4 files][ 5.5 MiB/ 5.5 MiB]  
==> NOTE: You are performing a sequence of gsutil operations that may  
run significantly faster if you instead use gsutil -m cp ... Please  
see the -m section under "gsutil help options" for further information  
about when gsutil -m can be advantageous.  
  
Copying file://data/adult.test.csv [Content-Type=text/csv]...  
Copying file://data/census.train.csv [Content-Type=text/csv]...  
\ [6 files][ 10.7 MiB/ 10.7 MiB]  
Operation completed over 6 objects/10.7 MiB.
```

Set data variables to point to storage bucket files

```
TRAIN_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.data.csv  
EVAL_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.test.csv
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ EVAL_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.test.csv  
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ TRAIN_DATA=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/adult.data.csv
```

Additional set-up: Copy test.json to storage bucket (for prediction later on)

```
gsutil cp ./test.json gs://$DEVSHELL_PROJECT_ID-aip-demo/data/test.json
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil cp ./test.json gs://$DEVSHELL_PROJECT_ID-aip-demo/data/test.json
Copying file:///.../test.json [Content-Type=application/json]...
/ [1 files] [ 314.0 B/ 314.0 B]
Operation completed over 1 objects/314.0 B.
```

Set TEST_JSON to point to the same storage bucket file

```
TEST_JSON=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/test.json
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ TEST_JSON=gs://$DEVSHELL_PROJECT_ID-aip-demo/data/test.json
```

Set variables for job name and output path (for AI Platform)

```
# cloud storage staging bucket which is called as the job name
```

```
JOB_NAME=census_single_0
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_single_1
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

Submit a single process job to AI Platform

- Job name is JOB_NAME (**census_single_1**)
- Output path is our Cloud storage bucket/**job_name**
- Training and evaluation/test data is in our **Cloud Storage bucket**

```
gcloud ai-platform jobs submit training $JOB_NAME \
--job-dir $OUTPUT_PATH \
--runtime-version 1.15 \
--module-name trainer.task \
--package-path trainer/ \
--region $REGION \
-- \
--train-files $TRAIN_DATA \
--eval-files $EVAL_DATA \
--train-steps 1000 \
--eval-steps 100 \
--verbosity DEBUG
```

AI Platform						
Jobs		+ NEW TRAINING JOB <small>BETA</small>	REFRESH	CANCEL	SHOW INF	
Dashboard	Job ID	Type	HyperTune	HyperTune parameters	Target metric	Create time
AI Hub	<input type="checkbox"/> census_single_0	Custom code	No			May 26, 2020, 1:43:50 PM
Data Labeling	<input type="checkbox"/>					
Pipelines	<input type="checkbox"/>					
Notebooks	<input type="checkbox"/>					
Jobs	<input checked="" type="checkbox"/> census_prediction_1	Prediction	N/A			May 4, 2020, 2:55:55 PM

```
leonf.info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_single_0
leonf.info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
leonf.info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform jobs submit training $JOB_NAME \
> --job-dir $OUTPUT_PATH \
> --runtime-version 1.15 \
> --module-name trainer.task \
> --package-path trainer/ \
> --region $REGION \
> -- \
> --train-files $TRAIN_DATA \
> --eval-files $EVAL_DATA \
> --train-steps 1000 \
> --eval-steps 100 \
> --verbosity DEBUG
Job [census_single_0] submitted successfully.
Your job is still active. You may view the status of your job with the command

$ gcloud ai-platform jobs describe census_single_0
or continue streaming the logs with the command

$ gcloud ai-platform jobs stream-logs census_single_0
jobid: census_single_0
state: QUEUED
```

Can view streaming logs/output (stackdriver logging) with gcloud ai-platform jobs stream-logs \$JOB_NAME

The screenshot shows the AI Platform Jobs interface. On the left is a sidebar with icons for Dashboard, AI Hub, Data Labeling, Pipelines, Notebooks, and Jobs (which is selected). The main area displays two rows of job information. The first row is for a HyperTune job that completed on May 26, 2020, at 1:43:50 PM, with an elapsed time of 4 min 49 sec. The second row is for a job labeled 'N/A' that completed on May 4, 2020, at 2:55:55 PM, with an elapsed time of 8 min 39 sec. Both rows have 'Logs' buttons, which are circled in blue.

The screenshot shows the Stackdriver Logging interface. The top navigation bar includes 'CLASSIC', 'CREATE METRIC', 'CREATE SINK', 'SAVE SEARCH', and 'SHOW LIBRARY'. The left sidebar has tabs for 'Logs Viewer' (selected), 'Logs-based Metrics', 'Logs Router', and 'Logs Ingestion'. A message at the top says 'New features are available in the Logs Viewer Preview. To switch between the preview and classic interfaces, use the drop-down in the top action bar.' Below this, a search bar shows a log entry: 'resource.labels.job_id="census_single_0" timestamp>="2020-05-26T17:43:50Z"'. The main area displays a list of log entries from May 26, 2020, at 13:45:49.292 EDT, all related to 'master-replica-0' transforming feature_columns. There are buttons for 'Download logs' and 'View Options'.

When complete, inspect output path with gsutil ls -r \$OUTPUT_PATH (in cloud storage bucket with model.ckpt) and we will use those files for deployment

The screenshot shows the Google Cloud Storage Bucket details page. The left sidebar has a 'Storage' icon (circled in blue) and tabs for 'Browser', 'Transfer', 'Transfer for on-premises', 'Transfer Appliance', and 'Settings'. The main area shows the bucket 'data-engineer-274519-aip-demo'. It has tabs for 'Objects', 'Overview', 'Permissions', and 'Bucket Lock'. Below these are buttons for 'Upload files', 'Upload folder', 'Create folder', 'Manage holds', and 'Delete'. A 'Filter by prefix...' input field is present. The 'Objects' section lists three items: 'checkpoint' (130 B, Standard, 5/26/20, 1:45:48 PM UTC-4), 'eval_census-eval/' (Folder, Standard, Subject to object ACLs), and 'events.out.tfevents.1590515126.cmle-training-' (4.02 MB, Standard, 5/26/20, 1:46:05 PM UTC-4).



Run distributed training on AI Platform (on 4 machines)

Create variable for distributed job name

```
cd ~/cloudml-samples-master/census/estimator
```

```
JOB_NAME=census_dist_2
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_dist_2
```

Set new output path to Cloud Storage location using new JOB_NAME variable

```
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$
JOB_NAME
```

Submit a distributed training job

The '**--scale-tier STANDARD_1**' option is the new item that initiates distributed scaling

1 master, 4 workers, 3 parameter servers

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_dist_2
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
> --job-dir $OUTPUT_PATH \
> --runtime-version 1.15 \
> --module-name trainer.task \
> --package-path trainer/ \
> --region $REGION \
> --scale-tier STANDARD_1 \
> --
> --train-files $TRAIN_DATA \
> --eval-files $EVAL_DATA \
> --train-steps 1000 \
> --verbosity DEBUG \
> --eval-steps 100

Job [census_dist_2] submitted successfully.
Your job is still active. You may view the status of your job with the command
$ gcloud ai-platform jobs describe census_dist_2
or continue streaming the logs with the command
$ gcloud ai-platform jobs stream-logs census_dist_2
jobId: census_dist_2
state: QUEUED
```

```

gcloud ai-platform jobs submit training $JOB_NAME \
--job-dir $OUTPUT_PATH \
--runtime-version 1.15 \
--module-name trainer.task \
--package-path trainer/ \
--region $REGION \
--scale-tier STANDARD_1 \
-- \
--train-files $TRAIN_DATA \
--eval-files $EVAL_DATA \
--train-steps 1000 \
--verbosity DEBUG \
--eval-steps 100

```

	Jobs	+ NEW TRAINING JOB <small>BETA</small>	<small>REFRESH</small>	<small>CANCEL</small>
<input type="checkbox"/> Dashboard	Filter by prefix...	Type	HyperTune	HyperTune parameters
<input type="checkbox"/> AI Hub	census_dist_2	Custom code training	No	
<input type="checkbox"/> Data Labeling				
<input type="checkbox"/> Pipelines				
<input type="checkbox"/> Notebooks				
<input type="checkbox"/> Jobs	census_dist_0	Custom code training	No	

View the training input

Job Details

census_dist_0

Succeeded (6 min 55 sec)

TensorBoard TensorBoard is available from this page only for models trained with built-in TensorFlow algorithms

ML units

Training input SHOW JSON

Training output SHOW JSON

Model location gs://data-engineer-274519-aip-demo/census_dist_1

CPU **GPU** **NETWORK**

Master CPU utilization

Training input

```

{
  "scaleTier": "STANDARD_1",
  "packageUris": [
    "gs://data-engineer-274519-aip-demo/census_dist_1/packages/14b762b89b902453f2475d5"
  ],
  "pythonModule": "trainer.task",
  "args": [
    "--train-files",
    "gs://data-engineer-274519-aip-demo/data/adult.data.csv",
    "--eval-files",
    "gs://data-engineer-274519-aip-demo/data/adult.test.csv",
    "--train-steps",
    "1000",
    "--verbosity",
    "DEBUG"
  ],
  "region": "us-central1",
  "runtimeVersion": "1.15",
  "jobDir": "gs://data-engineer-274519-aip-demo/census_dist_1"
}

```

HIDE JSON

Bucket details

data-engineer-274519-aip-demo

Objects

Training input

```

{
  "name": "checkpoint",
  "size": 130 B,
  "type": "File"
}

{
  "name": "eval_census-eval/",
  "size": 4.02 MB,
  "type": "Folder"
}

{
  "name": "events.out.tfevents.1590515126.cmlc-training-",
  "size": 4.02 MB,
  "type": "File"
}

```

Prediction phase (testing it out)

Deploy a model (Named **census_2**) for prediction, setting variables in the process

The deployed mode name: census

```
cd ~/cloudml-samples-master/census/estimator
```

```
MODEL_NAME=census_2
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ cd ~/cloudml-samples-master/census/estimator
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ MODEL_NAME=census_2
```

Create the ML Engine model (a platform)

```
gcloud ai-platform models create $MODEL_NAME --regions=$REGION
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform models create $MODEL_NAME --regions=$REGION
WARNING: Using endpoint [https://ml.googleapis.com/]
Created ml engine model [projects/data-engineer-274519/models/census_2].
```

The screenshot shows the Google Cloud AI Platform interface. On the left, there's a sidebar with icons for Dashboard, AI Hub, Data Labeling, Pipelines, Notebooks, Jobs, and Models. The 'Models' icon is highlighted with a blue circle and has a blue arrow pointing to it from the bottom-left. The main content area is titled 'Models' and has a 'NEW MODEL' button. It includes a 'Region' dropdown set to 'us-central1' and a 'Filter by prefix...' input field. Below these are two tables. The first table lists models with columns: Name, Default version, Description, Endpoint, and Labels. It shows three entries: 'census' (v1, ml.googleapis.com), 'census_0' (-, ml.googleapis.com), and 'census_2' (-, ml.googleapis.com). The second table lists versions with columns: Model, Version, Description, Endpoint, and Labels. It shows three entries: 'census' (v1, ml.googleapis.com), 'census_0' (-, ml.googleapis.com), and 'census_2' (-, ml.googleapis.com).

Name	Default version	Description	Endpoint	Labels
census	v1		ml.googleapis.com	
census_0	-		ml.googleapis.com	
census_2	-		ml.googleapis.com	

Model	Version	Description	Endpoint	Labels
census	v1		ml.googleapis.com	
census_0	-		ml.googleapis.com	
census_2	-		ml.googleapis.com	

Set the job output we want to use. This example uses **census_dist_2**

The screenshot shows the 'Bucket details' page for 'data-engineer-274519-aip-demo'. The left sidebar has 'Storage' selected. The main area shows the bucket name and navigation links: 'Objects' (underlined), 'Overview', 'Permissions', and 'Bucket Lock'. Below are buttons for 'Upload files', 'Upload folder', 'Create folder', 'Manage holds', and 'Delete'. A search bar says 'Filter by prefix...'. The path 'Buckets / data-engineer-274519-aip-demo / census_dist_2 / export / census' is shown. A table lists objects, with the first entry '1590520565/' circled in blue.

Name	Size	Type	Storage class	Last modified	Public access	Encryption	Ret.
1590520565/	-	Folder	-	-	Subject to object ACLs	-	-

```
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/census_dist_2
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/census_dist_2
```

CHANGE CENSUS_DIST_1 TO USE A DIFFERENT OUTPUT FROM PREVIOUS

IMPORTANT - Look up and set full path for **export trained model binaries**

```
gsutil ls -r $OUTPUT_PATH/export
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gsutil ls -r $OUTPUT_PATH/export
gs://data-engineer-274519-aip-demo/census_dist_2/export/:
gs://data-engineer-274519-aip-demo/census_dist_2/export/
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/:
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/saved_model.pb

gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/variables/:
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/variables/
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/variables/variables.data-00000-of-00002
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/variables/variables.data-00001-of-00002
gs://data-engineer-274519-aip-demo/census_dist_2/export/census/1590520565/variables/variables.index
```

Look for directory \$OUTPUT_PATH/export/census/ and copy/paste **timestamp value (without colon)** into the below command

```
MODEL_BINARIES=gs://$DEVSHELL_PROJECT_ID-aip-
demo/census_dist_2/export/census/<timestamp> ###CHANGE ME! (blue circle)
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ MODEL_BINARIES=gs://$DEVSHELL_PROJECT_ID-aip-dem
o/census_dist_2/export/census/1590520565
```

Create Version 1 of your model

Now deploy the version 1 model

```
gcloud ai-platform versions create v1 \
--model $MODEL_NAME \
--origin $MODEL_BINARIES \
--runtime-version 1.15
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform versions create v1 \
> --model $MODEL_NAME \
> --origin $MODEL_BINARIES \
> --runtime-version 1.15
WARNING: Using endpoint [https://ml.googleapis.com/]
Creating version (this might take a few minutes).....done.
```

Name	Default version	Description	Endpoint	Last updated
census	v1		ml.googleapis.com	2020-06-15 10:12:45 UTC
census_0	-		ml.googleapis.com	2020-06-15 10:12:45 UTC
census_2	v1		ml.googleapis.com	2020-06-15 10:12:45 UTC

Send an [online prediction request](#) to our deployed model using test.json file

Results come back with a direct response ([single data point](#))

```
gcloud ai-platform predict \
--model $MODEL_NAME \
--version v1 \
--json-instances \
./test.json
```

```
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform predict \
> --model $MODEL_NAME \
> --version v1 \
> --json-instances \
> ./test.json
WARNING: Using endpoint [https://ml.googleapis.com/]
ALL_CLASS_IDS ALL_CLASSES CLASS_IDS CLASSES LOGISTIC           LOGITS           PROBABILITIES
[0, 1]          ['0', '1']  [0]          ['0']   [0.047716088593006134] [-2.9935946464538574]  [0.9522839188575745, 0.047716084867715836]
```

Send a [batch prediction job](#) using same test.json file

Results are exported to a [Cloud Storage bucket](#) location

Set job name and output path variables

```
JOB_NAME=census_prediction_2
OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ JOB_NAME=census_prediction_2
leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ OUTPUT_PATH=gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME
```

Submit the prediction job

```
gcloud ai-platform jobs submit prediction $JOB_NAME \
--model $MODEL_NAME \
--version v1 \
--data-format TEXT \
--region $REGION \
--input-paths $TEST_JSON \
--output-path $OUTPUT_PATH/predictions
```

```

leonf_info@cloudshell:~/cloudml-samples-master/census/estimator (data-engineer-274519)$ gcloud ai-platform jobs submit prediction $JOB_NAME \
--model $MODEL_NAME \
--version v1 \
--data-format TEXT \
--region $REGION \
--input-paths $TEST_JSON \
--output-path $OUTPUT_PATH/predictions

Job [census_prediction_2] submitted successfully.
Your job is still active. You may view the status of your job with the command

$ gcloud ai-platform jobs describe census_prediction_2

or continue streaming the logs with the command

$ gcloud ai-platform jobs stream-logs census_prediction_2
jobId: census_prediction_2
state: QUEUED

```

Jobs			
<input type="button" value="NEW TRAINING JOB"/> BETA <input type="button" value="REFRESH"/>			
Filter by prefix...			
<input type="checkbox"/>	Job ID	Type	HyperTune
<input type="checkbox"/>	census_prediction_2	Prediction	N/A
<input type="checkbox"/>	<input checked="" type="checkbox"/> census_dist_2	Custom code training	No
<input type="checkbox"/>	<input checked="" type="checkbox"/> census_dist_0	Custom code training	No
<input type="checkbox"/>	<input checked="" type="checkbox"/> census_single_0	Custom code	No

View results in web console at

`gs://$DEVSHELL_PROJECT_ID-aip-demo/$JOB_NAME/predictions/`

Name	Size	Type	Storage class	Last modified
prediction.errors_stats-00000-of-00001	0 B	text/plain	Standard	5/26/2023 3:44:21 UTC-4
prediction.results-00000-of-00001	219 B	text/plain	Standard	5/26/2023 3:44:14 UTC-4

Lecture: Vision API Demo

The commands we used to authenticate with our API key, create `request.json` file, and the `curl` command used to call on the API are below for your reference.

The screenshot shows two overlapping Google Cloud Platform interfaces. On the left is the 'Storage' interface for a bucket named 'picture_api_demo'. It displays a list of objects, including a single file named '1835469822.jpg'. On the right is the 'API Library' interface, which lists the 'Cloud Vision API' under the 'Image Content Analysis' category. A blue circle highlights the 'ENABLE' button for the Cloud Vision API.

1. Authenticate with your API key after **creating the API credential**:

```
export API_KEY=(your copied key)
```

The screenshot shows the 'APIs & Services' section of the Google Cloud Platform console. Under the 'Credentials' tab, the 'API key' option is highlighted with a blue circle. Below it, other credential types like 'OAuth client ID' and 'Service account' are listed. A red box highlights the 'API key' section, and a red box highlights the copied API key value: 'AIzaSyB1hX15vTTyea5skCP6PC0xqLOxNB-h2rg'.

```
leonf_info@cloudshell:~ (data-engineer-274519)$ export API_KEY=AIzaSyB1hX15vTTyea5skCP6PC0xqLOxNB-h2rg
```

2. Create your JSON file (**points to img location on bucket**) by typing:

```
vim request.json
```

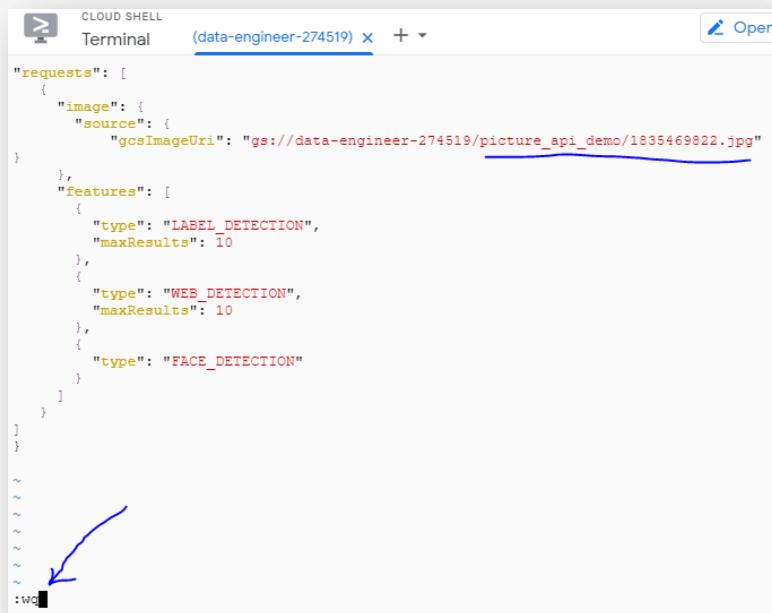
3. Then copy/paste the following text, substituting the Cloud Storage info with your actual Cloud Storage bucket and object info:

```
  "requests": [
    {
      "image": {
        "source": {
          "gcsImageUri": "gs://(your_bucket)/(your_image_file)"
        }
      },
      "features": [
        {
          "type": "LABEL_DETECTION",
          "maxResults": 10
        },
        {
          "type": "WEB_DETECTION",
          "maxResults": 10
        },
        {
          "type": "FACE_DETECTION"
        }
      ]
    }
  ]
}

CLOUD SHELL Terminal (data-engineer-274519)

"requests": [
  {
    "image": {
      "source": {
        "gcsImageUri": "gs://data-en
      }
    },
    "features": [
      {
        "type": "LABEL_DETECTION",
        "maxResults": 10
      },
      {
        "type": "WEB_DETECTION",
        "maxResults": 10
      },
      {
        "type": "FACE_DETECTION"
      }
    ]
  }
]

~
~
~
~
~
~
~
~
~
~
:wg
```



4. Use the below **curl command** to authenticate with the Vision API using your API key, and supplying the *request.json* file we just created:

```
curl -s -X POST -H "Content-Type: application/json" --data-binary @request.json  
https://vision.googleapis.com/v1/images:annotate?key=${API_KEY}
```

Lecture: Datalab Demo

```
$ datalab create [-h] [--image-name IMAGE_NAME] [--disk-name DISK_NAME]
                  [--disk-size-gb DISK_SIZE_GB] [--network-name NETWORK_NAME]
                  [--subnet-name SUBNET_NAME] [--idle-timeout IDLE_TIMEOUT]
                  [--machine-type MACHINE_TYPE] [--no-connect] [--no-swap]
                  [--no-backups] [--beta-no-external-ip] [--no-create-repository]
                  [--log-level {trace,debug,info,warn,error,fatal}]
                  [--for-user FOR_USER] [--service-account SERVICE_ACCOUNT]
                  [--port PORT] [--max-reconnects MAX_RECONNECTS]
                  [--connection-health-timeout-seconds CONNECTION_HEALTH_TIMEOUT_SECONDS]
                  [--ssh-log-level {quiet,fatal,error,info,verbose,debug,debug1,debug2,debug3}]
                  [--no-launch-browser] [--beta-internal-ip]
                  [--project PROJECT] [--quiet]
                  [--verbosity {debug,info,default,warning,error,critical,none}]
                  [--zone ZONE]
                  NAME
```

<https://cloud.google.com/datalab/docs/reference/command-line/create>

To open a datalab via command line, we need to use google **cloud shell**

Create datalab notebook:

- (1) Go over different **options**
- (2) Custom **machine type**
- (3) Separate VPC / source repository created for you

Connecting / web preview:

- (1) Authenticated as **service account**, rather than individual google account

To create a Datalab notebook using Cloud Shell, type the below command:

datalab create (instance-name)

```
leonf_info@cloudshell:~ (data-engineer-274519)$ datalab create datalab-demo
Please specify a zone from one of:
[1] us-east1-b
[2] us-east1-c
[3] us-east1-d
[4] us-east4-c
[5] us-east4-b
[6] us-east4-a
[7] us-central1-c
[8] us-central1-a ←
[9] us-central1-f
[10] us-central1-b
[11] us-west1-b
[12] us-west1-c
[13] us-west1-a
[14] europe-west4-a
```

```
[70] us-west4-c
Your selected zone: 8
Creating the instance datalab-demo
Created [https://www.googleapis.com/compute/v1/projects/data-engineer-274519/zones/us-central1-a/instances/datalab-demo].
Connecting to datalab-demo.
This will create an SSH tunnel and may prompt you to create an rsa key pair. To manage these keys, see https://cloud.google.com/compute/
docs/instances/adding-removing-ssh-keys
Waiting for Datalab to be reachable at http://localhost:8081/ ←
```

'datalab-network' is our default network

The screenshot shows the 'VPC networks' section of the Google Cloud console. On the left, there's a sidebar with icons for VPC networks, External IP addresses, Firewall rules, Routes, VPC network peering, and Shared VPC. The 'VPC networks' icon is highlighted with a blue circle. The main table lists 'VPC networks' with columns for Name, Region, Subnets, Mode, IP address ranges, Gateways, and Firewall rules. A row for 'datalab-network' is selected, with its name circled in blue. The table data is as follows:

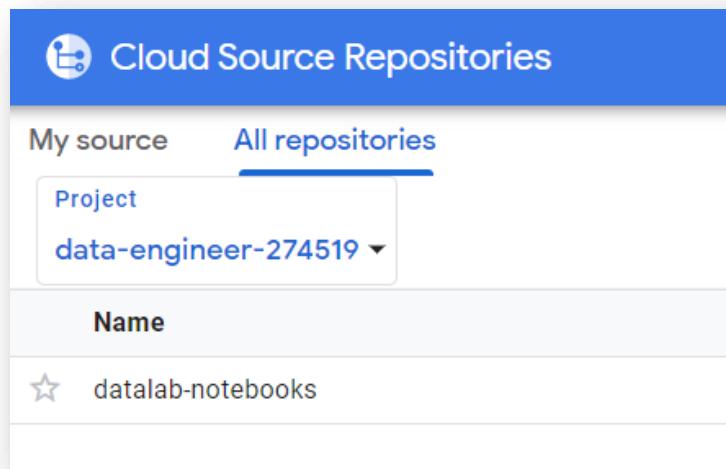
Name	Region	Subnets	Mode	IP address ranges	Gateways	Firewall rules
datalab-network	us-central1	datalab-network	Auto	10.128.0.0/20	10.128.0.1	1
	europe-west1	datalab-network		10.132.0.0/20	10.132.0.1	
	us-west1	datalab-network		10.138.0.0/20	10.138.0.1	
	asia-east1	datalab-network		10.140.0.0/20	10.140.0.1	
	us-east1	datalab-network		10.142.0.0/20	10.142.0.1	
	asia-northeast1	datalab-network		10.146.0.0/20	10.146.0.1	

'Port 22' is for the datalab network

The screenshot shows the 'Firewall rules' section of the Google Cloud console. The sidebar has icons for VPC networks, External IP addresses, Firewall rules (which is highlighted with a blue circle), Routes, VPC network peering, Shared VPC, Serverless VPC access, and Packet mirroring. The main area shows a table of firewall rules. One rule, 'datalab-network-allow-ssh', is highlighted with a blue circle and has an arrow pointing from it to the 'Network' column, which also has a blue circle around 'datalab-network'. The table data is as follows:

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network	Logs
datalab-network-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	1000	datalab-network	Off
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534	default	Off
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/9	tcp:0-65535 udp:0-65535 icmp	Allow	65534	default	Off
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534	default	Off
default-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65534	default	Off

And we have a repository call '**datalab-notebooks**' created for us



To connect to a Datalab notebook, type:

datalab connect (instance-name)

```
leonf_info@cloudshell:~ (data-engineer-274519)$ datalab connect datalab-demo
Connecting to datalab-demo.
This will create an SSH tunnel and may prompt you to create an rsa key pair. To manage these keys, see https://cloud.google.com/compute/
docs/instances/adding-removing-ssh-keys
Waiting for Datalab to be reachable at http://localhost:8081/

The connection to Datalab is now open and will remain until this command is killed.
Click on the *Web Preview* (square button at top-right), select *Change port > Port 8081*, and start using Datalab.
```

Working in a notebook

You can create a New notebook here, and share it by using the Pushing (next step)

The screenshot shows the Google Cloud Datalab interface. On the left, there is a sidebar with a navigation bar and a file tree. The file tree shows a root folder named 'datalab' containing 'docs' and 'notebooks'. On the right, there is a main workspace where a new notebook is being edited. The code cell contains the following Python code:

```
# Hello, this is a markdown sentence
print('hello world')
hello world

!pwd
/content/datalab/notebooks

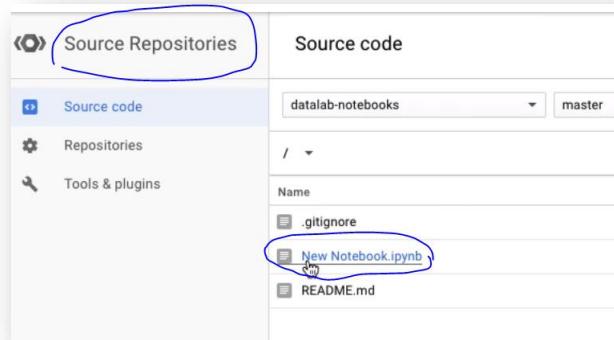
%%bash
ls -a
pwd

..
.git
.gitignore
.ipynb_checkpoints
New Notebook.ipynb
```

Pushing to source repository

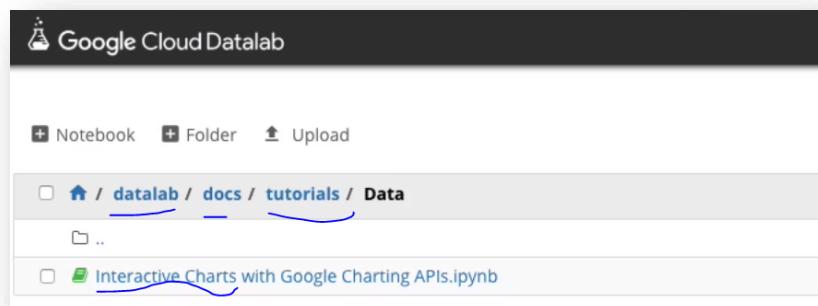
The screenshot shows a source control interface, likely GitHub, with a commit dialog open. The dialog has a 'Commit' button highlighted with a blue circle. In the background, there is a list of files ready to be committed, and a summary bar at the bottom indicates 1 file, 1 to be committed. Below the commit dialog, there is a status bar showing 'Running' and a 'Jump to file' button. At the bottom right, there is a 'master' branch indicator with a 'Push' button, also highlighted with a blue circle.

Go back to repo, now we have a new file:

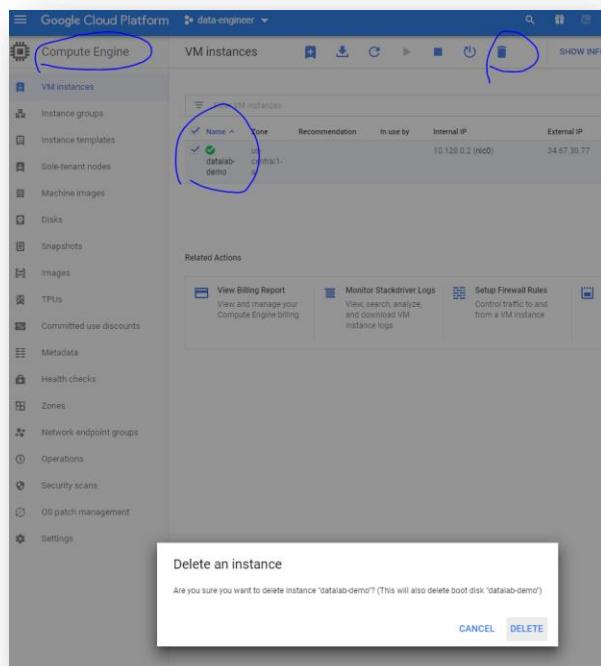


Examples

Lots of useful learning materials here



Clean up



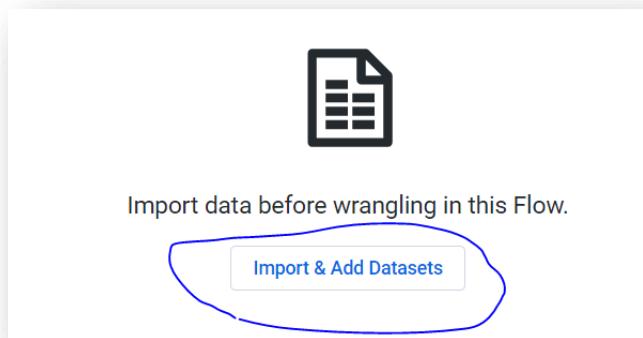
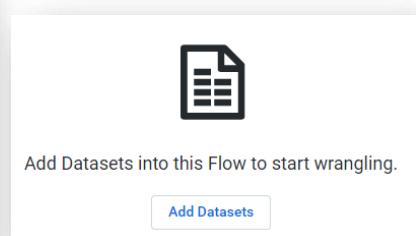
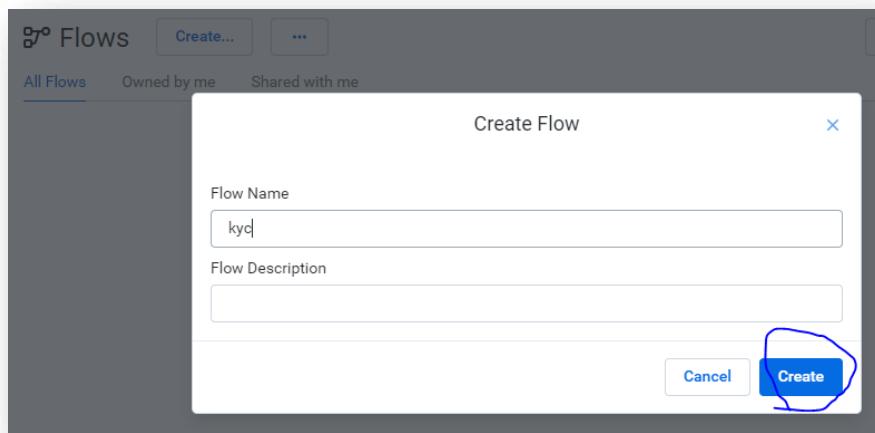
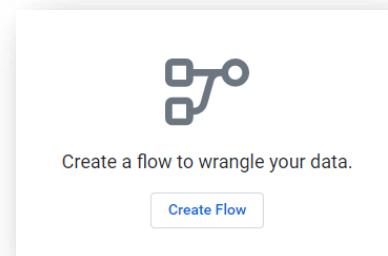
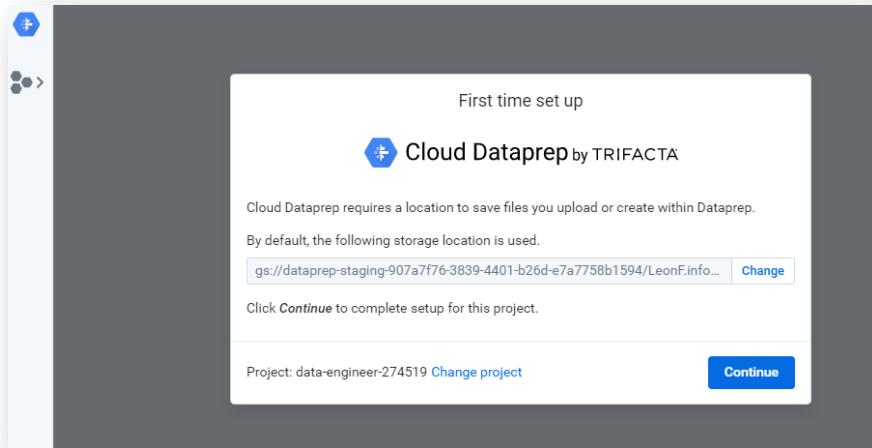
Lecture: Dataprep Demo Part 1

<https://cloud.google.com/dataprep/docs/quickstarts/quickstart-dataprep>

This is the first in a three-part demo of Dataprep.

The public cloud storage bucket we used for our datasets is `gs://la-dataprep-samples`.

NOTE: Be aware that the above data is no longer available for the Customer 360 subfolder. You can find similar examples at `gs://la-dataprep-samples`



Import Data and Add to Flow

Choose a file or folder
GCS / acp_demo

Create Dataset with Parameters

Search... (/)

Upload

GCS

BigQuery

NAME	SIZE	LAST UPDATED
kyc.csv	80MB	Today at 5:51 PM

Import & Add to Flow Cancel

1 New Dataset Clear All

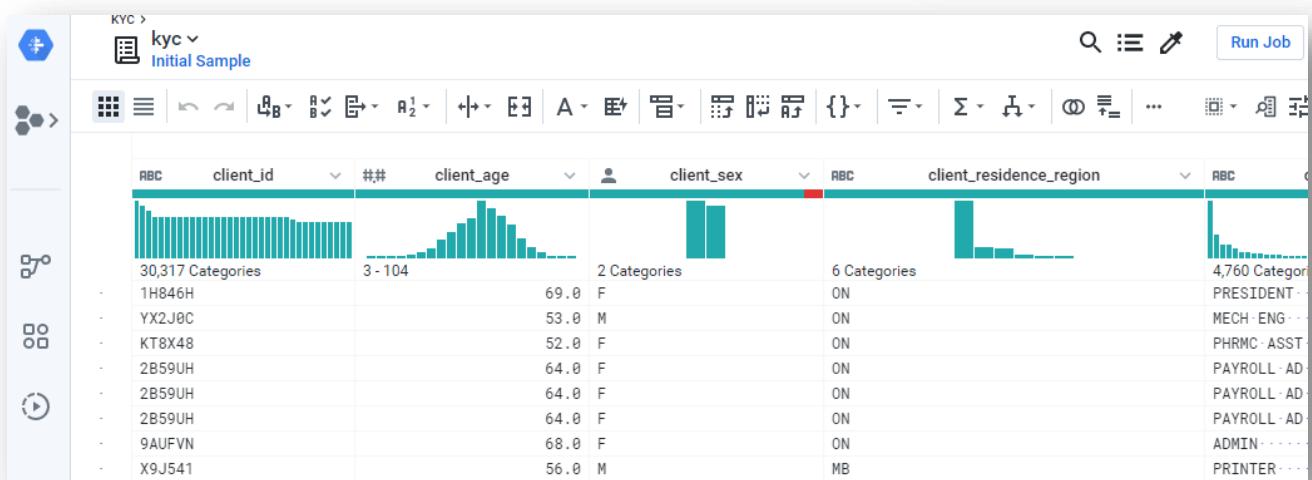
kyc.csv Add a Description

RBC client_id	## client_age	client_sex
1H846H	69.0	F
YX2J0C	53.0	M
KT8X48	52.0	F
2B59UH	64.0	F
2B59UH	64.0	F

Edit settings

Click **add new recipe**, and then **edit recipe**:

(it using cloud storage for the staging process)



Lecture: Dataprep Demo Part 2

This is the second portion of our fairly extensive Dataprep demo.

Suggestion

The screenshot shows the Dataprep interface with the 'Suggestions' panel open. The 'Replace' section contains a suggestion to replace the first occurrence of '{start}{delim}' with '' in the SessionID column. The 'Add' button for this suggestion is circled. The 'Split on values matching' section shows a preview of splitting on '{delim}'. The 'Transformation' panel on the right shows a formula being built: `ifmissing($col, DID_NOT_COMPLETE)`. The 'New Step' button in the top right of the transformation panel is also circled.

Manual recipe

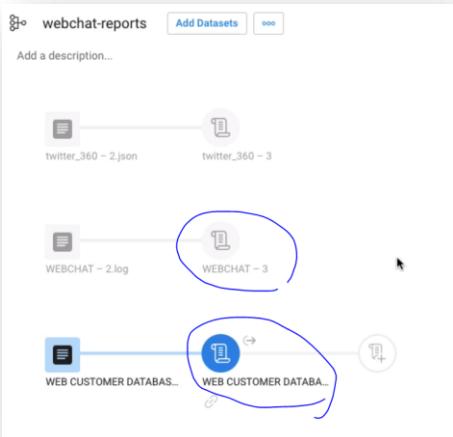
The screenshot shows the Dataprep interface with a manual recipe open. The 'New Step' button in the top right is circled. Below it, a list of 8 steps is shown:

- 1 Replace matches of `|` from client_occupation_desc with `
- 2 Replace matches of `` from client_occupation_desc with `
- 3 Replace matches of `(delim){end}` from client_occupation_desc with `
- 4 Replace matches of `(delim){end}` from client_occupation_desc with `
- 5 Set client_occupation_desc to IFMISSING(\$col, NULL())
- 6 Replace cells which equal 'US RRSP' in acct_type with 'RRSP'
- 7 Replace cells which equal 'Margin Short' in acct_type with 'Margin'

Lecture: Dataprep Demo Part 3

This is the third part of our Dataprep demo. In this lesson, we will take our joined datasets, and start a managed Dataflow job, which we will then insert into a BigQuery table.

Join all the processed datasets together



The screenshot shows the Dataprep interface with a table view on the left and a "Recipe" panel on the right. The "Recipe" panel displays a list of steps:

- 1 Delete rows where column1 == '--- Chats Export ---'
- 2 Delete rows where ISMISSING([column1])
- 3 Extract key value pairs from column1 into column2 where keys match '{alpha}+' and values are after ':'
- 4 Create new columns from 8 constants in column2
- 5 Replace '{start}{delim}' from AgentID with ''
- 6 Replace '{start}{delim}' from CustomerID with ''
- 7 Replace '&' from GeoInfo with ''
- 8 Replace '{start}{delim}' from Rating with ''

The "New Step" button in the Recipe panel is circled in blue.

The screenshot shows the Dataprep interface with a table view on the left and an "Add Step" dialog on the right. The "Transformation" dropdown in the dialog is set to "join" and is circled in blue. The dialog also contains a description of the "Join" step: "Adds additional columns from other data sources [join]".

Then it goes to a new screen page:

The screenshot shows the 'SELECT NEW DATA TO JOIN' interface. On the left, under 'CURRENT DATA', there is a list of fields for 'WEBCHAT - 3'. In the center, a search bar and a list of datasets are shown. One dataset, 'WEB CUSTOMER DATABASE - 3', is selected and highlighted with a blue circle. On the right, there are buttons for 'Cancel', 'Add to Recipe', and 'Preview', with 'Preview' also highlighted with a blue circle.

Next, set up the join key:

The screenshot shows the 'Join Keys' interface. It has two sections: 'CURRENT DATA' (WEBCHAT - 3) and 'NEW DATA' (WEB CUSTOMER DATABASE - 3). A 'Join Keys' button is highlighted with a blue circle. Below the sections, a table lists potential joining keys. The first row, '# SessionID' from WEBCHAT - 3 and '# sessionid' from WEB CUSTOMER DATABASE - 3, is selected and highlighted with a green background. Other rows include '# AgentID' from WEBCHAT - 3 and '# AgentID' from WEB CUSTOMER DATABASE - 3.

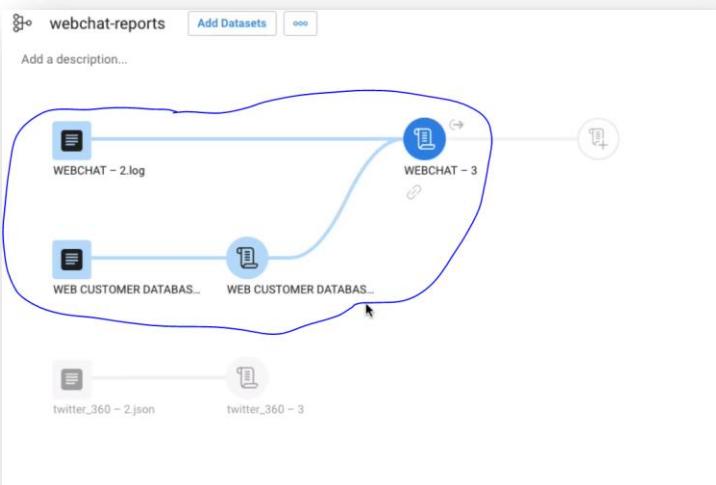
Automatically identify the joining keys for both datasets:

The screenshot shows the 'Join Keys' interface with the 'Data Select new' tab selected. The 'Join Keys' section shows a table of identified keys. The first row, '# SessionID' from WEBCHAT - 3 and '# sessionid' from WEB CUSTOMER DATABASE - 3, is highlighted with a blue circle. The second row, '# sessionid' from WEB CUSTOMER DATABASE - 3 and '# AgentID' from WEBCHAT - 3, is also highlighted with a blue circle. The third row, '# AgentID' from WEBCHAT - 3 and '# AgentID' from WEB CUSTOMER DATABASE - 3, is partially visible.

Click 'all' to add those different columns together:

The screenshot shows a data join configuration window. At the top, it says "Join Data Select new Join Keys Select keys & previous join Input Rows: WEBCHAT - 3 4,971 WEB CUSTOMER DATABASE - 3 5,566 Output Rows: 7,555 Inner Join - PREVIEW". On the left, under "Join Keys", there's a table with columns "Type", "Name", and "Source". A row for "sessionID" is selected, with "All" checked. The "Add" button is highlighted with a blue oval. On the right, a preview table shows two columns: "SessionID" and "sessionid1". The preview table has a yellow header row and contains several rows of data. A circled "2" is in the top right corner.

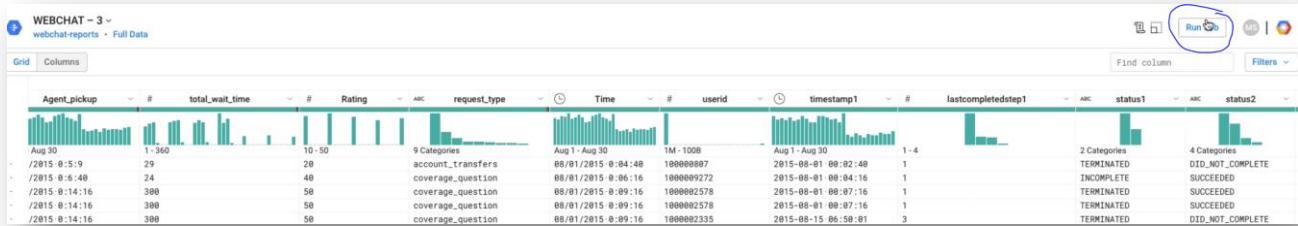
Now we have more data to work with:



Now move data to BigQuery:

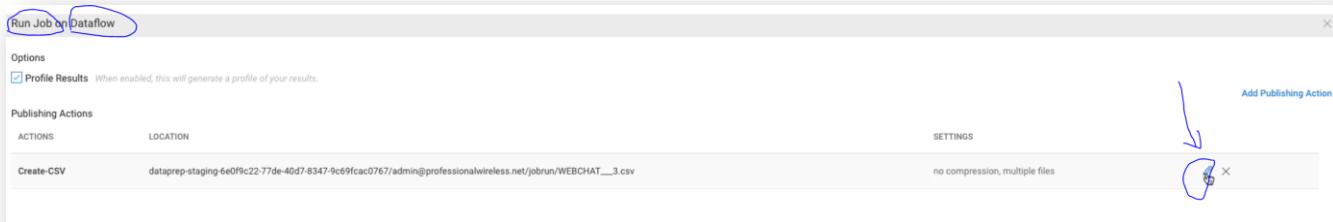
The screenshot shows the Google BigQuery interface. At the top, it says "Google BigQuery". Below that is a "COMPOSE QUERY" button. To the right is a "Queries" section with tabs for "Query History", "Saved Queries", "Project Queries", and "Sort by: Date". A dropdown menu is open, with "Create new dataset" highlighted and circled. The main area shows a sidebar with "pw-data-engineer", "bigquery-public-data", "nyc-tlc", and "Public Datasets" sections. The "Query History" tab shows a list of recent queries, each with a green checkmark and some SQL code. A blue oval highlights the "Create new dataset" option in the dropdown menu.

Back to table, click on 'run job':

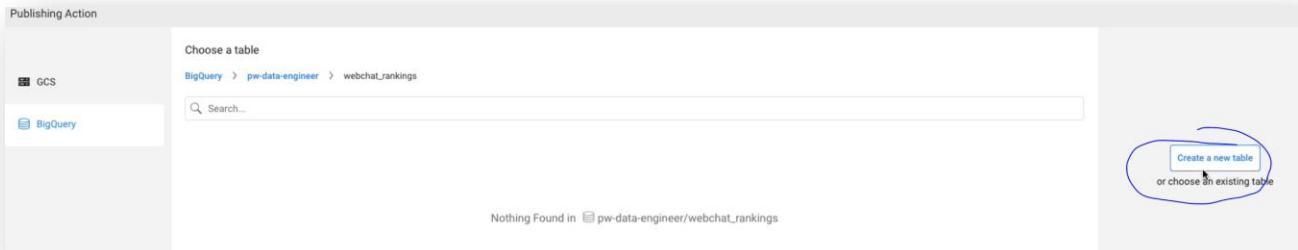
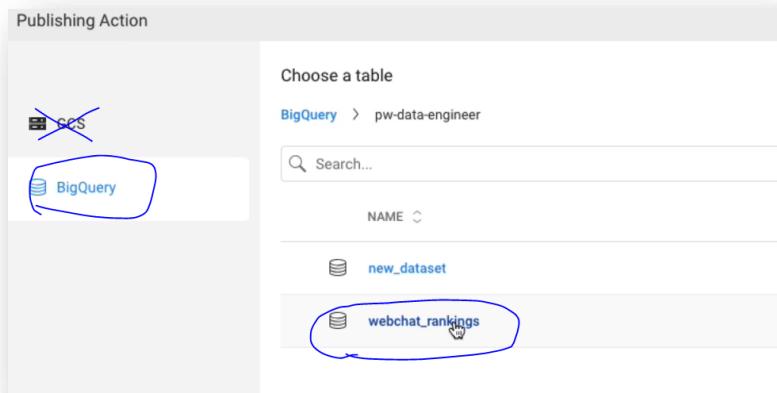


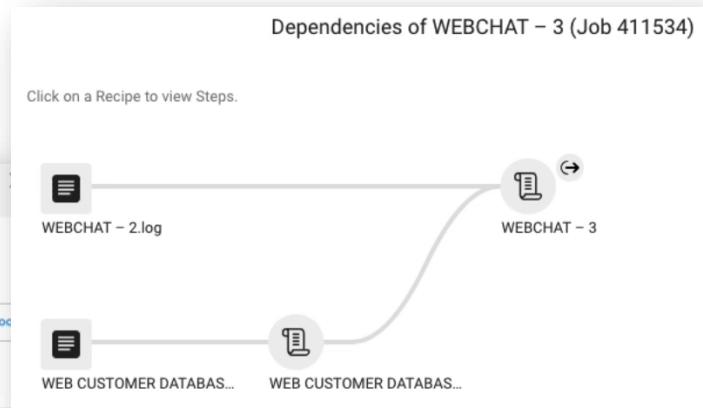
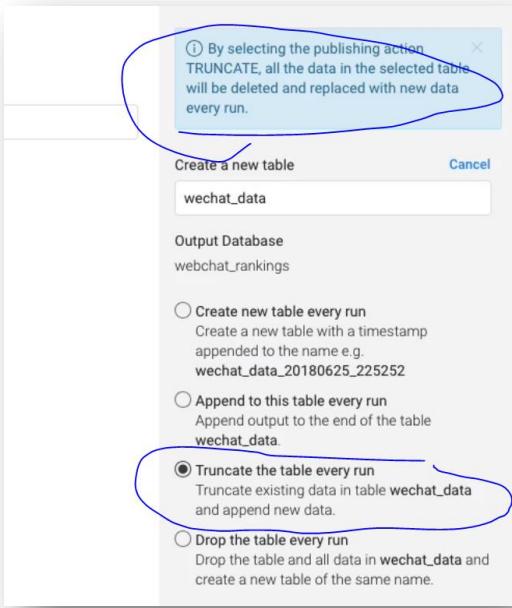
By default, it will create CSV file and will move to GCS. But this time, we don't want the default since we need to move to BigQuery.

Change the default option:



Choose **BigQuery**, create a new table, **save setting**, and click **run job**:





Details

WEBCHAT – 3

Run Job

Destinations **Jobs (1)**

Job 411534 • Transforming
started Today at 10:53 PM

View steps and dependencies

Details

WEBCHAT – 3

Run Job

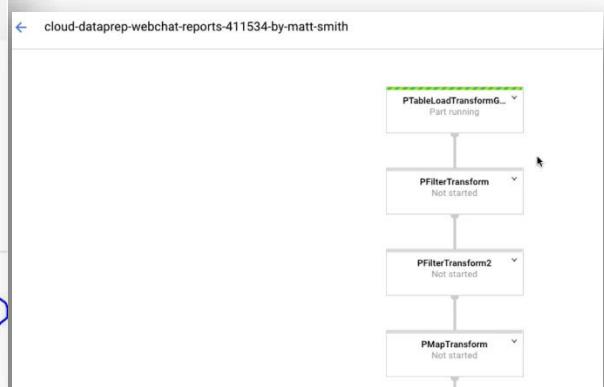
Destinations **Jobs (1)**

Job 411534 • Transforming .
started Today at 10:53 PM

More

View Dataflow Job

View steps and dependencies



Done:

Google BigQuery

Table Details: wechat_data

COMPOSE QUERY

Query History
Job History

Filter by ID or label ?

pw-data-engineer

- ▶ new_dataset
- ▼ webchat_rankings
- wechat_data** (highlighted with a blue oval)
- ▶ bigquery-public-data
- ▶ nyc-tlc

Schema Details Preview

Row	SessionID	sessionid1	AgentID	CustomerID	CustomerName	Initiate_time	Agent_pickup	total_wait_time	R
1	184686	184686	1470	1000005607	Stephen	2015-08-07T12:31:36	2015-08-07T12:32:04	28	
2	184686	184686	1131	1000006401	Ann	2015-08-18T11:33:21	2015-08-18T11:37:21	240	
3	184686	184686	1120	1000005339	Lillian	2015-08-20T02:32:43	2015-08-20T02:36:06	203	
4	184686	184686	1588	1000002962	Linda	2015-08-26T03:08:31	2015-08-26T03:11:32	181	
5	187034	187034	1403	1000009951	Charles	2015-08-13T04:25:15	2015-08-13T04:31:15	360	
6	181316	181316	1180	1000009951	Charles	2015-08-19T05:05:19	2015-08-19T05:05:49	30	
7	187164	187164	1131	1000009951	Charles	2015-08-01T17:32:32	2015-08-01T17:32:54	22	
8	187164	187164	1990	1000005875	Alan	2015-08-17T22:09:01	2015-08-17T22:10:02	61	

Lecture: Data Studio Demo

In this lesson, we will demonstrate working with Data Studio. We will take the cleaned Dataprep document from the previous section which we exported to BigQuery, and use that data source to get insights about our ratings.

The screenshot shows the Google Data Studio dashboard. The top navigation bar includes the Data Studio logo, a search bar, and user profile icons. Below the navigation is a toolbar with a 'Create' button, a 'Recent' tab (which is selected), and other tabs for 'Reports', 'Data sources', and 'Explorer'. A sidebar on the left shows 'Shared with me', 'Owned by me', and 'Trash' sections. The main content area features a 'Start with a Template' section displaying three templates: 'Blank Report' (Data Studio), 'Tutorial Report' (Data Studio), and 'Acme Marketing' (Google Analytics). Below this is a 'Create a Report' section with a large circular 'Create' button containing a blue arrow icon. At the bottom of this section is the text 'Use the Create button to add one.'

Can choose GCS or SQL

The screenshot shows the 'SELECT CONNECTOR' dialog box in Google Data Studio. The 'BigQuery' connector is highlighted with a blue oval. The dialog includes a description of BigQuery as Google's managed data warehouse and links to 'LEARN MORE' and 'REPORT AN ISSUE'. In the background, the main Data Studio interface is visible, showing a list of data sources: Attribution 360, DFP, Google Cloud Storage, Google Analytics, and BigQuery. The 'BigQuery' connector is also circled in blue on the main interface.

Detect the data type automatically, and click 'add to report'

Index	Field	Type	Aggregation	Description
14	EMAIL	RBC Text	None	
15	userid	123 Number	None	
16	FIRST_NAME	RBC Text	None	
17	STATUS	RBC Text	None	
18	status1	RBC Text	None	
19	Initiate_time	Date (YYYYMMDD)	None	

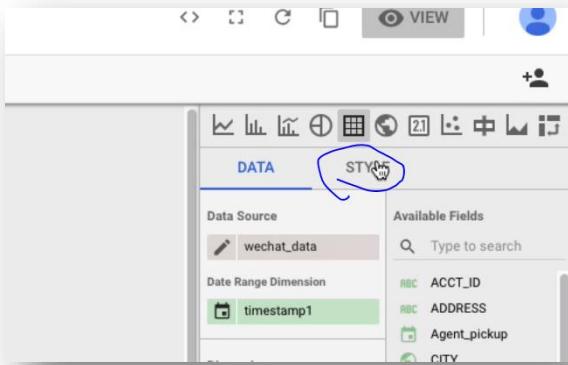
Make a table, click and drag

AgentID	Record Count	total_wait_time
1.	426	55,436
2.	399	48,750
3.	384	49,406
4.	376	46,854
5.	373	43,932
6.	364	48,227
7.	360	46,702
8.	360	45,948
9.	344	40,888
10.	344	39,867
11.	342	44,375
12.	341	45,140
13.	334	39,497
14.	334	38,331
15.	333	41,447
16.	329	47,750

Since 'adding' is the default calculation. Need to create a new field (to use aggregation), click 'create new field' at the bottom. Go to the field (column) we need to aggregate. Re-add that field to the 'metric'.

Index	Field	Type	Aggregation	Description
22	Agent_pickup	Date (YYYYMMDD)	None	
23	CUSTOMER_NAME	123 Number	None	
24	ACCT_ID	RBC Text	None	
25	INITIAL_WAIT_TIME	123 Number	Average	
26	START_DATE	Date (YYYYMMDD)	None	
27	status2	RBC Text	None	
28	sessionID	123 Number	Count	
29	CITY	City	Count	
30	ADDRESS	RBC Text	Count Distinct	
31	OCCUPATION	RBC Text	Max	

Change the table style



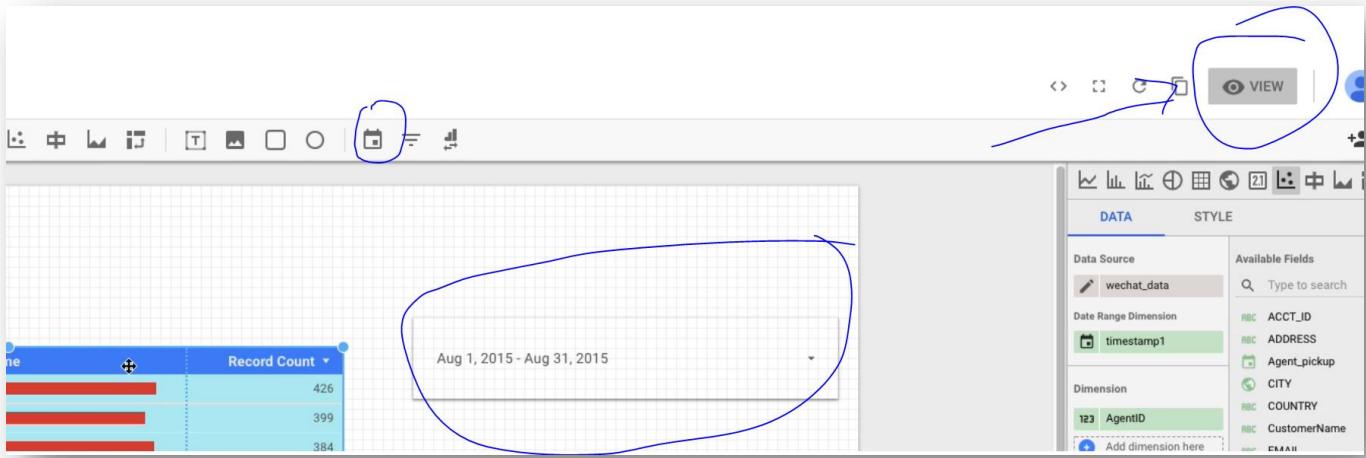
Change column to heatmap

The screenshot shows a data visualization application. The left pane displays a table with four columns: 'Agent_', 'Rating', 'total_wait_time', and 'Record Count'. The 'total_wait_time' column is highlighted with a blue arrow. The right pane shows the 'STYLE' tab settings for this column, specifically under 'Column #1'. The 'Heatmap' dropdown is selected, and a blue circle highlights it. Other settings include 'Compact Numbers' and 'Decimal Precision' set to 'auto'. The 'Number' dropdown is also visible.

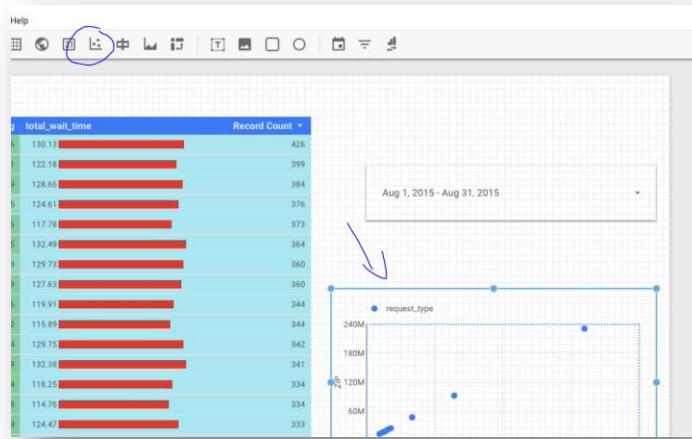
Change column to bar (with showing the number aside)

The screenshot shows a data visualization application. The left pane displays a table with four columns: 'Agent_', 'Rating', 'total_wait_time', and 'Record Count'. The 'total_wait_time' column is now represented by horizontal red bars. The right pane shows the 'STYLE' tab settings for this column, specifically under 'Column #1'. The 'Bar' dropdown is selected, and a blue circle highlights it. Other settings include 'Compact Numbers' and 'Decimal Precision' set to 'auto'. A blue circle also highlights the 'Show number' checkbox, which is checked. The 'Number' dropdown is also visible.

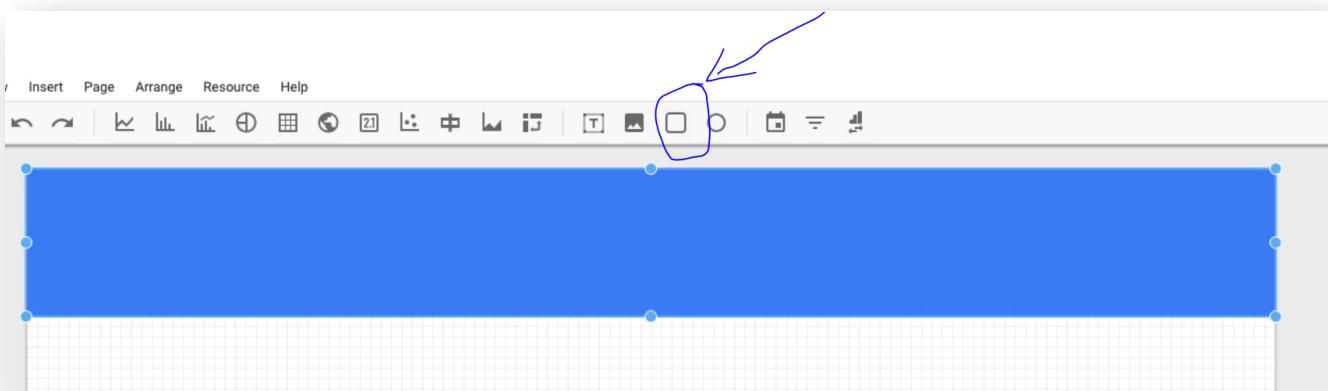
Add **time filter**, click and drag. And use the view mode to check it out



Make **scatter plot**



Make **report title**



Add control filters

A screenshot of a report interface. At the top, there's a toolbar with icons for file, edit, view, insert, page, arrange, resource, and help. Below the toolbar is a date range selector showing "Aug 1, 2015 - Aug 31, 2015". The main content area displays a table titled "request_type Rating". The table has two rows: "null" with a rating of 40 and "close_account" with a rating of 36.15. A blue circle and an arrow point to the filter icon in the top right corner of the report area.

A screenshot of a report interface showing a search bar at the top with the placeholder "Type to search". Below the search bar is a table titled "request_type (2) Record Count". The table lists two items: "account_lockout" (4K) and "coverage_question" (1.6K). Both items have checkboxes next to them, which are checked. Other items listed include "account_transfers" (840), "close_credit_card" (447), "new_account" (392), and "open_account" (346). A blue circle highlights the checkbox for "coverage_question".

Finally, report setting (for sharing)

If the table need to change frequently, untick 'enable cache'. Otherwise, just leave it.

A screenshot of a report interface titled "Untitled Report". The menu bar includes File, Edit, View, Insert, Page, Arrange, Resource, and Help. A dropdown menu is open under the "File" tab, showing options like "Share", "Report settings" (which is highlighted with a blue circle), "New report", "Open...", "Make a copy...", and "Embed report".

A screenshot of the "Report Settings" dialog box. It contains sections for "Data Source" (with a "Select Data Source" button), "Cache Control" (with a checked "Enable cache" checkbox), and "Google Analytics Tracking ID" (with a question mark icon).

Share the report to people

A screenshot of a report interface showing a "VIEW" button and a user profile icon. Below these are "Layout and Theme" settings. Under "Layout", there's a "Share the Report" button with a plus sign icon, which is highlighted with a blue circle.

Lecture: Hands On - Cloud Composer

This lesson will be a hands on tour of Cloud Composer in action. The commands we used in this lesson are duplicated below:

Step1: Enable Composer / Dataproc API **First and** Use the default settings:

The screenshot shows the 'Create environment' page for Google Cloud Composer. At the top, there's a 'Name *' field containing 'my-environment'. Below it is a 'Node configuration' section with a 'Node count *' field set to '3'. Underneath are 'Location *' (set to 'us-central1') and 'Zone' (set to 'us-central1'). A note explains that the location is where the environment will be created, and the zone is where the VMs will run the Apache Airflow software.

Done, the DAGs folder is a short-cut to storage bucket:

Name	Location	Creation time	Update time	Airflow webserver	Logs	DAGs folder	Labels
my-environment	us-central1	5/8/20, 3:14 PM	5/8/20, 3:27 PM	Airflow	Logs	DAGs	None

Storage	Bucket details	EDIT BUCKET	REFRESH BUCKET												
Browser	us-central1-my-environment-cccbc235-bucket	Objects	Overview Permissions Bucket Lock												
Transfer		Upload files Upload folder Create folder Manage holds Delete													
Transfer for on-premises		Filter by prefix...													
Transfer Appliance	Buckets / us-central1-my-environment-cccbc235-bucket / dags														
Settings	<table><thead><tr><th>Name</th><th>Size</th><th>Type</th><th>Storage class</th><th>Last modified</th><th>Public access</th></tr></thead><tbody><tr><td>airflow_monitoring.py</td><td>729 B</td><td>text/x-python</td><td>Standard</td><td>5/8/20, 3:25:21 PM UTC-4</td><td>Not public</td></tr></tbody></table>	Name	Size	Type	Storage class	Last modified	Public access	airflow_monitoring.py	729 B	text/x-python	Standard	5/8/20, 3:25:21 PM UTC-4	Not public		
Name	Size	Type	Storage class	Last modified	Public access										
airflow_monitoring.py	729 B	text/x-python	Standard	5/8/20, 3:25:21 PM UTC-4	Not public										

Also Kubernetes cluster is running at the same time:

The screenshot shows the Google Cloud Platform interface for the Kubernetes Engine. The top navigation bar includes 'Google Cloud Platform', a dropdown for 'data-engineer', and a search bar. Below the navigation is a sidebar with options: Clusters, Workloads, Services & Ingress, and Applications. The main area is titled 'Kubernetes clusters' and contains a table with one row. The row details a cluster named 'us-central1-my-environment-cccbc235-gke'. Columns include 'Name', 'Location' (us-central1-c), 'Cluster size' (3), 'Total cores' (3 vCPUs), 'Total memory' (11.25 GB), 'Notifications', and 'Labels' (goog-compo... : my-environ...). Buttons for 'CREATE CLUSTER', 'DEPLOY', 'REFRESH', and 'DELETE' are also present.

Step 2: Create another **separate** GCS bucket to output **Dataproc** results (using Project ID shell variable in composer shell environment)

- `gsutil mb -l us-central1 gs://output-$DEVSHELL_PROJECT_ID`

The screenshot shows the Google Cloud Platform Storage browser. The left sidebar has 'Storage' selected. The main area lists several buckets: 'dataproc-temp-us-central1-457820818325-3fabjohw', 'output-data-engineer-274519' (which is circled in blue), 'us-central1-my-environment-cccbc235-bucket', and 'vision_demo_api'. A terminal window at the bottom shows the command being run: `gsutil mb -l us-central1 gs://output-$DEVSHELL_PROJECT_ID`. Two blue arrows point from the text 'gsutil mb -l us-central1 gs://output-\$DEVSHELL_PROJECT_ID' in the terminal to the circled bucket name 'output-data-engineer-274519' in the Storage browser list.

Step 3: Before the workflow execute, set our Cloud Composer variables necessary.

Format

- **gcloud composer environments run (ENVIRONMENT_NAME) --location (LOCATION) variables -- --set (KEY VALUE)**
- **KEY VALUE pair example: gcp_project \$DEVSHELL_PROJECT_ID**

Project ID will again be represented by the shell variable to auto-resolve to your unique Project ID:

- **gcloud composer environments run my-environment --location us-central1 variables -- --set gcp_project \$DEVSHELL_PROJECT_ID (set variable for project ID)**
- **gcloud composer environments run my-environment --location us-central1 variables -- --set gcs_bucket gs://output-\$DEVSHELL_PROJECT_ID (set variable for cloud storage bucket)**

- **gcloud composer environments** run my-environment --location us-central1 variables -- --set gce_zone us-central1-c (set variable for zone)

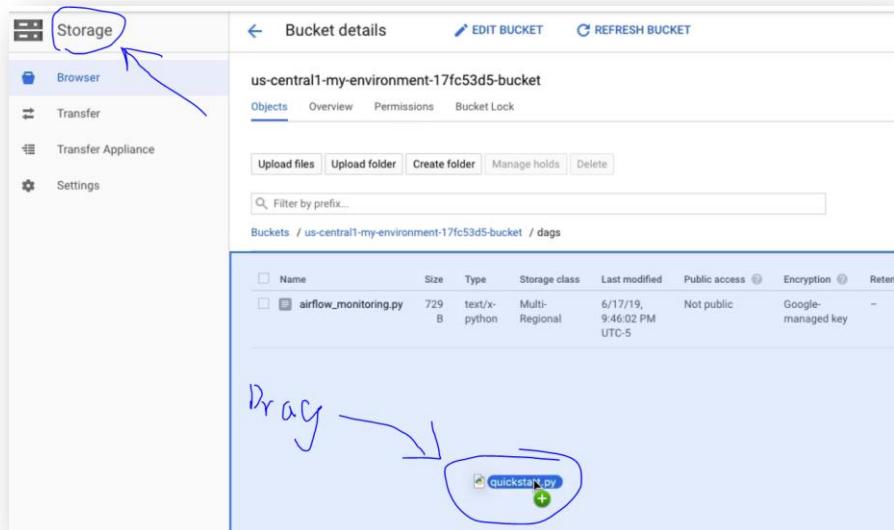
```
leonf.info@cloudshell:~ (data-engineer-274519)$ gcloud composer environments run my-environment --location us-central1 variables -- --set gce_zone gs://output-$DEVSHELL_PROJECT_ID
kubecomfig entry generated for us-central1-my-environment-cccbc235-gke.
Executing within the following Kubernetes cluster namespace: composer-1-10-2-airflow-1-10-3-cccbc235
[2020-05-09 17:08:05,812] {settings.py:185} INFO - settings.configure_orm(): Using pool settings. pool_size=5, pool_recycle=1800, pid=10043
[2020-05-09 17:08:06,170] {default_celery.py:90} WARNING - You have configured a result backend of redis://airflow-redis-service.default.svc.cluster.local:6379/0, it is highly recommended to use an alternative result backend (i.e. a database).
[2020-05-09 17:08:06,175] {_init_.py:51} INFO - Using executor CeleryExecutor
[2020-05-09 17:08:06,584] {app.py:54} WARNING - Using default Composer Environment Variables. Overrides have not been applied.
[2020-05-09 17:08:06,599] {configuration.py:556} INFO - Reading the config from /etc/airflow/airflow.cfg
[2020-05-09 17:08:06,620] {settings.py:185} INFO - settings.configure_orm(): Using pool settings. pool_size=5, pool_recycle=1800, pid=10043
```

The screenshot shows the Airflow Variables page. At the top, there are buttons for 'Choose File' (No file chosen) and 'Import Variables'. Below is a table with the following data:

	Key	Val	Is Encrypted
<input type="checkbox"/>	gce_zone	us-central1-c	<input checked="" type="radio"/>
<input type="checkbox"/>	gcp_project	pw-composer	<input checked="" type="radio"/>
<input type="checkbox"/>	gcs_bucket	gs://output-pw-composer	<input checked="" type="radio"/>

Step 4: Upload the example DAG file ([Python file](#)) to the DAG folder for Cloud Composer. A copy of the DAG file can be found at this link.

- Direct link for local download: <https://storage.googleapis.com/la-gcloud-course-resources/data-engineer/cloud-composer/quickstart.py>
- Public GCS location: gs://la-gcloud-course-resources/data-engineer/cloud-composer/quickstart.py
- GitHub link: <https://github.com/GoogleCloudPlatform/python-docs-samples/blob/b80895ed88ba86fce223df27a48bf481007ca708/composer/workflows/quickstart.py>
- Or if we have download the file to local directory, just drag it to storage bucket:



Cloud storage will automatically detect the quickstart.py:

The screenshot shows the Airflow web interface with the following details:

- Header:** Airflow logo, DAGs, Data Profiling, Browse, Admin, Docs, About, my-environment, 03:02 UTC, and a power icon.
- Notification Bar:** "DAG [] is now fresh as a daisy" with a close button.
- Section Header:** DAGs
- Search Bar:** Search: []
- DAG List Table:**

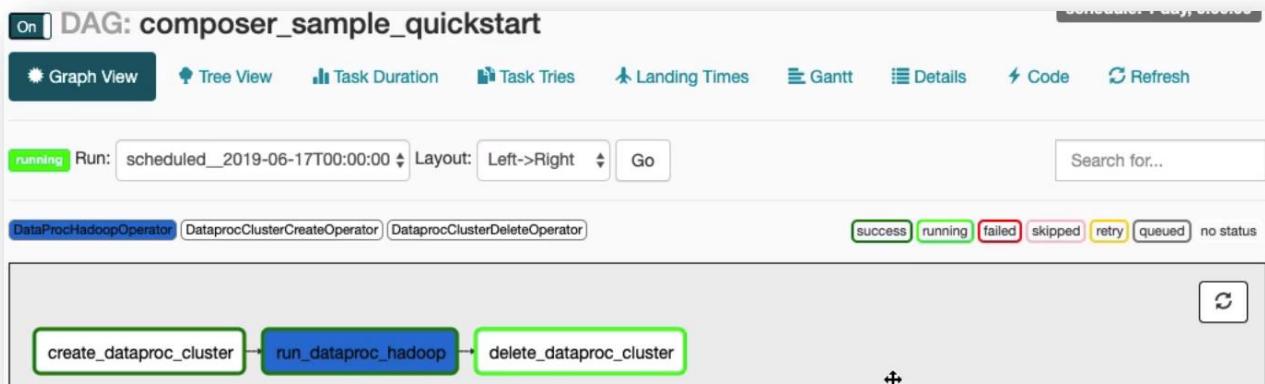
		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links	
<input checked="" type="checkbox"/>		airflow_monitoring		Airflow					
<input checked="" type="checkbox"/>		composer_sample_quickstart		Airflow					

Refresh it, it shows it is processing:

The screenshot shows the Airflow web interface with the following details:

- Header:** Airflow logo, DAGs, Data Profiling, Browse, Admin, Docs, About, my-environment, 03:02 UTC, and a power icon.
- Notification Bar:** "DAG [] is now fresh as a daisy" with a close button.
- Title:** DAGs
- Search:** A search bar labeled "Search:".
- Table:** A list of DAGs with columns: DAG, Schedule, Owner, Recent Tasks, Last Run, DAG Runs, and Links.
 - airflow_monitoring:** On, None, Airflow, 1 recent task, Last run 2019-06-17 00:00:00, 3 DAG runs.
 - composer_sample_quickstart:** On, 1 day, 0:00:00, Airflow, 1 recent task, Last run 2019-06-17 00:00:00, 1 DAG run.

Doing the Hadoop job, and delete cluster after job done:



It create a dataproc cluster and do the work right now:

The screenshot shows the Dataproc Clusters interface. On the left, there's a sidebar with 'Clusters' (selected), 'Jobs', and 'Workflows'. The main area has a 'Clusters' tab, a 'CREATE CLUSTER' button, and a 'REFRESH' button. A search bar says 'Search clusters, press Enter'. A table lists clusters, with one row highlighted: 'quickstart-cluster-20190617' (global, us-central1-c, 2 nodes, Off, dataproc-9d135d18-51cf-4f6e-98df-d73f4c599392-us).

View result in bucket:

The screenshot shows the Storage Bucket details interface. On the left, there's a sidebar with 'Storage' (selected), 'Browser', 'Transfer', 'Transfer Appliance', and 'Settings'. The main area shows 'Bucket details' for 'output-pw-composer'. It has tabs for Objects, Overview, Permissions, and Bucket Lock. Under Objects, there are buttons for Upload files, Upload folder, Create folder, Manage holds, and Delete. A filter bar says 'Filter by prefix...'. A table lists objects in the 'wordcount' directory, with one object circled: 'part-r-00000' (59 B, application/octet-stream, Regional, 6/17/19, 10:06:00 PM UTC-5, Not public, Google-managed key).

Delete composer instance and storage bucket!!!

Below are additional resources directly from Google that I highly recommend working with for even more hands-on practice:

Official Google Cloud Data Engineer practice exam:

<https://cloud.google.com/certification/practice-exam/data-engineer>

Google Cloud Solutions Center:

<https://cloud.google.com/solutions/>

Google **Codelabs** for hands-on practice across all products:

<https://codelabs.developers.google.com/>

Big Data and Machine Learning Blog:

<https://cloud.google.com/blog/big-data/>

BigQuery Tutorials:

<https://cloud.google.com/bigquery/docs/tutorials>

Dataflow Tutorials:

<https://cloud.google.com/dataflow/examples/examples-beam>

Dataproc samples and tutorials:

<https://cloud.google.com/dataproc/docs/tutorials>

Pub/Sub tutorials:

<https://cloud.google.com/pubsub/docs/tutorials>

Cloud ML Engine tutorials:

<https://cloud.google.com/ml-engine/docs/tensorflow/tutorials>

Bigtable tutorials and samples:

<https://cloud.google.com/bigtable/docs/samples>

Cloud Spanner tutorials:

<https://cloud.google.com/spanner/docs/tutorials>

Additional study resources:

SQL deep dive

Course - SQL Primer

<https://linuxacademy.com/cp/modules/view/id/52>

Machine Learning

Google Machine Learning Crash Course (free)

<https://developers.google.com/machine-learning/crash-course/>

Hadoop

Hadoop Quick Start

<https://linuxacademy.com/cp/modules/view/id/294>

Apache Beam (Dataflow)

Google's guide to designing your pipeline with Apache Beam (using Java)

<https://cloud.google.com/dataflow/docs/guides/beam-creating-a-pipeline>

NOTE: The revised version of this course (and exam) as of the March 2019 refresh has the case studies removed.