COURSE LINKS

Friday, July 29, 2022 1:42 PM

BIG DATA AND ML FUINDAMENTALS

- [x] https://partner.cloudskillsboost.google/course_templates/3?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&search_id=5773044
- [x] Resources links refer

DATA ENGINEERING ON GCP

- [x] https://partner.cloudskillsboost.google/course_templates/3?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&search_id=5773044
- [x] https://partner.cloudskillsboost.google/course_templates/54?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&search_id=5773058 (labs are not available)
- [x] https://partner.cloudskillsboost.google/course_templates/53?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&search_id=5773059
- [x] https://partner.cloudskillsboost.google/course_templates/52
- [x] https://partner.cloudskillsboost.google/course_templates/55?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&search_id=5773071

DATA FLOW(EXTRA)

- [x] https://partner.cloudskillsboost.google/course_templates/218
- [x] https://partner.cloudskillsboost.google/course_templates/229 without some java labs
- [x] https://partner.cloudskillsboost.google/course_templates/264 without some labs that don't progress

BIGQUERY QUEST

- [x] https://partner.cloudskillsboost.google/quests/147?catalog_rank=%7B"rank"%3A1%2C"num_filters"%3A0%2C"has_search"%3Atrue%7D&search_id=17444667

CREATE AND MANAGE CLOUD RESOURCES

- [x] https://partner.cloudskillsboost.google/quests/120

Perform Foundational Data, ML, and AI Tasks in Google Cloud

- [x] https://partner.cloudskillsboost.google/quests/117

ENGINEER DATA ON GCP

- [x] https://partner.cloudskillsboost.google/quests/132

TEST

- [x] https://partner.cloudskillsboost.google/course_templates/72
- [ ] https://cloud.google.com/certification/data-engineer

https://cloud.google.com/certification/guides/data-engineer/

BIG DATA AND ML FUNDAMENTALS

Friday, July 29, 2022 12:11 PM

STREAMING DATA >> PUB/SUB >> DATAFLOW >> BIGQUERY >> DATASTUDIO,LOOKER,ETC../VERTEX AI
BATCH DATA >> GCS >>DATAFLOW >> BIGQUERY >> DATASTUDIO,LOOKER,ETC../VERTEX AI

Bigquery ML Commands

- Labels (identify columns as labels and pass it to input_label_cols in options)
- Features(Data columns that are part of our SELECT statement while creating model)
- Model Object created in the bigquery dataset we specified
- Model types(linear regression,logistic regression etc…)
  (CREATE OR REPLACE MODEL <datasetname>.<modelname>
  OPTIONS(model_type=<type>) AS <TRAINING_DATASET/sql query>)
- View training progress using
  (SELECT *FROM ML.TRAINING_INFO (MODEL ' dataset.model'))
- (SELECT *FROM ML.WEIGHTS (MODEL ' dataset.model'),<query>) for weights
- (SELECT *FROM ML.EVALUATE (MODEL ' dataset.model')) for evaluating
- For predicting the using the model we need to pass the test data as a query to the model
  (SELECT *FROM ML.PREDICT (MODEL ' dataset.model'),<query>)
- ROC (Receiver Operating Characteristic) curve is in roc_auc field to see the relationships
  Accuracy,precision,recall are other fields to test the relationships

Examples:

```
CREATE OR REPLACE MODEL `ecommerce.classification_model` OPTIONS
(
model_type='logistic_reg',
labels = ['will_buy_on_return_visit']
) AS
#standardSQL SELECT * EXCEPT(fullVisitorId) FROM
#features( SELECT query data);


SELECT roc_auc,
  CASE
    WHEN roc_auc > .9 THEN 'good'
    WHEN roc_auc > .8 THEN 'fair'
    WHEN roc_auc > .7 THEN 'not great'
  ELSE 'poor' END AS model_quality
FROM
  ML.EVALUATE(MODEL ecommerce.classification_model,  (
SELECT * EXCEPT(fullVisitorId) FROM # features QUERY DATA


SELECT
*
FROM
  ml.PREDICT(MODEL `ecommerce.classification_model_2`,
    (query that has the required data)
```

OPTIONS TO BUILD ML MODELS

-- BIGQUERY ML (SQL QUERY BASED)
-- PREBUILD API'S (USE LIKE A TEMPLATE THAT HAS BEEN ALREADY CREATED)
-- AUTOML (NO CODE STUFF USING VERTEX AI)
-- CUSTOM(FULLY MANAGED BY US WITH ALL THE CODE WRIITEN BY US FROM SCRATCH)

FLEXIBITLY TO TUNE HYPERPARAMTERS IS MORE IN CUSTOM

VERTEX AI SOLUTIONS THAT ARE AVAILABLE TO USE FROM AUTOML OR WORKBENCH:

1) Feature Store (centralized repository for organizing ,storing,serving features to feed to training models)
2) Vizier(for tuning hyperparameters in complex ml models)
3) Explainable AI(interpret training performance)
4) Pipelines (monitor the prod pipelines)

WORKFLOW: DATA PREPARATION >> MODEL TRAINING >> MODEL SERVING

DATA PREPARATION

- UPLOAD DATA (Automl supports image,tabular,text,video) can be from gcs,bq etc..
- FEATURE ENGINEERING (extracting or selecting features (columns,independent var))

MODEL TRAINING

- MODEL TRAINING
  SUPERVISED LEARNING - FOR PAST DATA FOR PREDICT FUTURE NEEDS
  (CLASSIFICATION-IDENTIFY IMAGE IS DOG OR CAT,REGRESSION BASED ON PAST STOCK FUTURE STOCK PRICE PREDICT)
  UNSUPERVISED LEARNING - GROUP CUSTOMERS

BIGQUERY ML SYNTAX FOR CREATION

{CREATE MODEL | CREATE MODEL IF NOT EXISTS | CREATE OR REPLACE MODEL}
model_name[TRANSFORM (select_list)]
[OPTIONS(model_option_list)]
[AS query_statement]
model_option_list:MODEL_TYPE = { 'LINEAR_REG' |
'LOGISTIC_REG' |
'KMEANS' |
'MATRIX_FACTORIZATION' |
'PCA' |
'AUTOENCODER' |
'AUTOML_CLASSIFIER' |
'AUTOML_REGRESSOR' |
'BOOSTED_TREE_CLASSIFIER' |
'BOOSTED_TREE_REGRESSOR' |
'DNN_CLASSIFIER' |
'DNN_REGRESSOR' |
'DNN_LINEAR_COMBINED_CLASSIFIER' |
'DNN_LINEAR_COMBINED_REGRESSOR' |
'ARIMA_PLUS' |
'TENSORFLOW'}
[, INPUT_LABEL_COLS = string_array]
[, MAX_ITERATIONS = int64_value]
[, EARLY_STOP = { TRUE | FALSE }]
[, MIN_REL_PROGRESS = float64_value]
[, DATA_SPLIT_METHOD = { 'AUTO_SPLIT' | 'RANDOM' | 'CUSTOM' | 'SEQ' | 'NO_SPLIT' }]
[, DATA_SPLIT_EVAL_FRACTION = float64_value]
[, DATA_SPLIT_COL = string_value]
[, OPTIMIZE_STRATEGY = { 'AUTO_STRATEGY' | 'BATCH_GRADIENT_DESCENT' | 'NORMAL_EQUATION' }]
[, L1_REG = float64_value]
[, L2_REG = float64_value]
[, LEARN_RATE_STRATEGY = { 'LINE_SEARCH' | 'CONSTANT' }]
[, LEARN_RATE = float64_value]
[, LS_INIT_LEARN_RATE = float64_value]
[, WARM_START = { TRUE | FALSE }]
[, AUTO_CLASS_WEIGHTS = { TRUE | FALSE }]
[, CLASS_WEIGHTS = struct_array]
[, NUM_CLUSTERS = int64_value]
[, KMEANS_INIT_METHOD = { 'RANDOM' | 'KMEANS++' | 'CUSTOM' }]
[, KMEANS_INIT_COL = string_value]
[, DISTANCE_TYPE = { 'EUCLIDEAN' | 'COSINE' }]
[, STANDARDIZE_FEATURES = { TRUE | FALSE }]
[, MODEL_PATH = string_value]
[, BUDGET_HOURS = float64_value]
[, FEEDBACK_TYPE = {'EXPLICIT' | 'IMPLICIT'}]
[, NUM_FACTORS = int64_value]
[, USER_COL = string_value]
[, ITEM_COL = string_value]
[, RATING_COL = string_value]
[, WALS_ALPHA = float64_value]
[, BOOSTER_TYPE = { 'gbtree' | 'dart'}]
[, NUM_PARALLEL_TREE = int64_value]
[, DART_NORMALIZE_TYPE = { 'tree' | 'forest'}]
[, TREE_METHOD = { 'auto' | 'exact' | 'approx' | 'hist'}]
[, MIN_TREE_CHILD_WEIGHT = float64_value]
[, COLSAMPLE_BYTREE = float64_value]
[, COLSAMPLE_BYLEVEL = float64_value]
[, COLSAMPLE_BYNODE = float64_value]
[, MIN_SPLIT_LOSS = float64_value]
[, MAX_TREE_DEPTH = int64_value]
[, SUBSAMPLE = float64_value])
[, ACTIVATION_FN = { 'RELU' | 'RELU6' | 'CRELU' | 'ELU' | 'SELU' | 'SIGMOID' | 'TANH' }]
[, BATCH_SIZE = int64_value]
[, DROPOUT = float64_value]
[, HIDDEN_UNITS = int_array]
[, OPTIMIZER = { 'ADAGRAD' | 'ADAM' | 'FTRL' | 'RMSPROP' | 'SGD' }]
[, TIME_SERIES_TIMESTAMP_COL = string_value]
[, TIME_SERIES_DATA_COL = string_value]
[, TIME_SERIES_ID_COL = { string_value| string_array}]
[, HORIZON = int64_value]
[, AUTO_ARIMA = { TRUE | FALSE }]
[, AUTO_ARIMA_MAX_ORDER = int64_value]
[, NON_SEASONAL_ORDER = (int64_value, int64_value, int64_value)]
[, DATA_FREQUENCY = { 'AUTO_FREQUENCY' | 'PER_MINUTE' | 'HOURLY' | 'DAILY' | 'WEEKLY' | ... }]
[, INCLUDE_DRIFT = { TRUE | FALSE }]
[, HOLIDAY_REGION = { 'GLOBAL' | 'NA' | 'JAPAC' | 'EMEA' | 'LAC' | 'AE' | ... }]

- MODEL TRAINING
  SUPERVISED LEARNING - FOR PAST DATA FOR PREDICT FUTURE NEEDS
  (CLASSIFICATION-IDENTIFY IMAGE IS DOG OR CAT,REGRESSION BASED ON PAST STOCK FUTURE STOCK PRICE PREDICT)
  UNSUPERVISED LEARNING - GROUP CUSTOMERS
  (CLUSTERING,ASSOCIATION,DIMENTIONALITY REDUCTION)
- MODEL EVALUATION (CONFUSOIN MATRIC,PRECISOIN ,RECALL,ETC..)

MODEL SERVING

- MODEL DEPLOYMENT
  (USING ENDPOINT (LOW LATENCY RESPONSE),
  BATCH PREDICTION(ONE REQUEST AS A BATCH),
  OFFLINE PREDICTION(OFF CLOUD))
- MODEL MONITORING
  (VertexAI PIPELINES)

VERTEX AI STEPS:

1) UPLOADING DATA TO GCS,BQ,LOACL
2) TRAINING THE MODEL IN AUTOML BY UI
3) AFTER IT IS TRAINED DEPLOY IT TO ENDPOINT
4) FOR ACCESSING THE ENDPOINT FOR MODEL SCORE GET THE BEARER TOKEN FROM
   https://gsp-auth-kjyo252taq-uc.a.run.app/

   And create env variables in like
   AUTH_TOKEN=token got
   ENDPOINT="endpoint from the model deployed"
   INPUT_DATA_FILE="INPUT-JSON"
   # like

{"endpointId": "1411183591831896064", "instance": "[{age: 40.77430558, ClientID: '997', income: 44964.0106, loan: 3944.219318}]"}

5) Install the smlproxy file form google and have the input json data file ready and execute this ./smlproxy tabular -a $AUTH_TOKEN -e $ENDPOINT -d $INPUT_DATA_FILE
   (we can also use python,rest api for getting response)
6) Response
   SML Tabular HTTP Response:
   2022/07/29 09:44:04 {"model_class":"0","model_score":0.9999981}

RECALL MEANS To IDENTIFY AS MUCH AS POSSIBLE POTENTIAL CASES

[, **DATA_FREQUENCY** = { 'AUTO_FREQUENCY' | 'PER_MINUTE' | 'HOURLY' | 'DAILY' | 'WEEKLY' | ... }]
[, **INCLUDE_DRIFT** = { TRUE | FALSE }]
[, **HOLIDAY_REGION** = { 'GLOBAL' | 'NA' | 'JAPAC' | 'EMEA' | 'LAC' | 'AE' | ... }]
[, **CLEAN_SPIKES_AND_DIPS** = { TRUE | FALSE }]
[, **ADJUST_STEP_CHANGES** = { TRUE | FALSE }]
[, **DECOMPOSE_TIME_SERIES** = { TRUE | FALSE }]
[, **ENABLE_GLOBAL_EXPLAIN** = { TRUE | FALSE }]
[, **INTEGRATED_GRADIENTS_NUM_STEPS** = int64_value]
[, **CALCULATE_P_VALUES** = { TRUE | FALSE }]
[, **FIT_INTERCEPT** = { TRUE | FALSE }]
[, **CATEGORY_ENCODING_METHOD** = { 'ONE_HOT_ENCODING', 'DUMMY_ENCODING' }]

From <https://cloud.google.com/bigquery-ml/docs/reference/standard-sql/bigqueryml-syntax-create>

Modernizing Data Lakes and Data Warehouses

Saturday, July 30, 2022 11:16 AM

Data Lake:
Brings data from across the enterprise into a single location (RDMS,OFFLINE FILES,SPREADSHEETS,ETC..) >> SINGLE LOCATION OR PROJECT
-- it should handle all types of data
-- it should scale, give high throughput ingestion
-- fine grained access control
-- access easily for other tools
GCS is mainly for staging the data

Challenges: - accessing data, data quality,accuracy,computation,query performance

DATA WAREHOUSE:
Data Lake >>>> ETL >>> DATA WAREHOUSES
Bigquery provides the data warehouse solution(hardware setup internally)

Data Mart <===> datasets
Data Lake <===> (GCS,BQ,BIGTABLE,…)
Tables and views <===> same in bq
Grants <===> IAM

- CLOUD SQL(POSTGRES,SQLSERVER,MYSQL) >FEDERATED QUERY> BQ (for this need to use external data source in ADD DATA section
  EXTERNAL_QUERY(
  'us.connection_id',
  '''SELECT customer_id, MIN(order_date) AS first_order_date
  FROM orders
  GROUP BY customer_id''') this function after adding the connection queries from the cloud sql)

Cloud SQL federated queries are subject to the following limitations:

- **Performance**. A federated query is likely to not be as fast as querying only BigQuery storage. BigQuery needs to wait for the source database to execute the external query and temporarily move data from the external data source to BigQuery. Also, the source database might not be optimized for complex analytical queries.
- **Federated queries are read-only**. The external query that is executed in the source database must be read-only. Therefore, DML or DDL statements are not supported.
- **Unsupported data types**. If your external query contains a data type that is unsupported in BigQuery, the query fails immediately. You can cast the unsupported data type to a different supported data type.
- **Limited Cloud SQL instances.** Federated querying is only supported by the Cloud SQL V2 instance with public IP (versus private IP).
- **Project**. You must create the connection resource in the same project as the Cloud SQL instance.

- CLOUD STORAGE(csv,parquet) >>>>BQ (
  CREATE OR REPLACE EXTERNAL TABLE mydataset.sales
  OPTIONS (
  format = 'CSV',
  uris = ['gs://mybucket/sales.csv']);

```
CREATE OR REPLACE EXTERNAL TABLE mydataset.sales (
  Region STRING,
  Quarter STRING,
  Total_Sales INT64
) OPTIONS (format = 'CSV', uris = ['gs://mybucket/sales.csv'],skip_leading_rows = 1);
```

- Using bigquery we can query data from gcs and cloud sql

- CLOUD SQL IS RECORD BASED MEANS IF WE SELECTR ONLY ONE COLUMN IT FETCHES ENTIRE RECORD BUT IN BQ IT IS COLUMN BASED

- USE RDMS FOR TRANSACTION BASED DATA

DATA WAREHOUSE >> BI PIPELINE
DATA WAREHOUSE >> ENGINEERING PIPELINE >> DATA WAREHOUSE
DATA WAREHOUSE >> ML MODELIING

Cloud Audit Logs in monitoring says who executed and which query executed

Data Catalog Service is a data discovery from all sources like bq,gcs,pubsub etc.. And integrates with DLP datalosspreevetion(for creditcard,numbers etc..)

DATACATALOG IS NOW PART OF DATAPLEX SERVICE

GCS -> Retention policy(expire after some time),versioning,Lifecycle management (coldline to nearline etc..)

CLOUD KMS -> in gcs the encrypiton keys is managed in kms for objects or our own customer managed,customer supplied keys

Data Lock -> we can lock the objects to be not able to accessed or other like bucket lock object lock specially for auditing

CLOUD SQL:
Horizontal scaling(adding additional machines/node) => read
Vertical scaling (scaling/upgrading existing machine)=>read,write
For horizontal scaling for both r w use spanner

A BigQuery slot is a combination of CPU, memory and networking resources.
by default, each account has a quota limit of 2000 to BigQuery slots for on-demand querying

ROW LEVEL ACESSES IN BQ:

```
CREATE ROW ACCESS POLICY
<policy_name>
ON <dataset.table>
GRANT TO
("group:sales@example.com",
"user:ram@example.com")
FILTER USING (Region="APAC");
```

WE CAN ALSO USE
CREATE VIEW <_._> as SELECT *FROM <table>
(CANT EXPORT A VIEW)
(IN SAME LOCATION -TABLE,VIEW OF IT)

1 TB IS FREE EVERY MONTH FOR QUERYING

Batch load supported file formats:
CSV,NEWLINE_DELIMITEDJSON,AVRO,DATASTORE
_BACKUP,PARQUERT,ORC

FROM AVRO BQ CAN AUTOMATICALLY INFER SCHEMAS
BUT I CSV AND JSON MANUAL IS RECOMMENDED

QUERY A SNAPSHOT DATA WHEN ACCIDENTLY LOST DATA

```
CREATE OR REPLACE TABLE <table>
AS SELECT *FROM <table> FOR SYSTEM_TIME AS OF
TIMESTAMP_SUB(CURRENT_TIMESTAMP(),INTERVAL 24 HOUR)
```

BASH:

```
NOW=$(date +%s)
SNAPSHOT=$(echo "($NOW -120)*1000" | bc)
Bq --location=EU cp \
<_.table>@$SNAPSHOT \
<_.restored_table>
```

CAN CREATE TEMP FUNCTIONS (UDF)IN BQ USING STANDARD SQL,JS,SCRIPTING

METADATA

```
select dataset_id,table_id,size_bytes/pow(10,9) as gb,
TIMESTAMP_MILLIS(creation_time),TIMESTAMP_MILLIS(last_modified_time
),
row_count,type-- (TYPE 1 ==TABLE)(2==VIEW)
 from bigquery-public-data.austin_incidents.__TABLES__;

select * from bigquery-public-
data.austin_incidents.INFORMATION_SCHEMA.COLUMNS

SELECT schema_name FROM studious-
lore-344410.INFORMATION_SCHEMA.SCHEMATA # THIS WILL SHOW ONLY
THE DATASETS CREATED USING CREATE SCHEMA
```

Examples such as ride pickupdrop company we don't need to create multiple tables
Or single bigtable containing all the orderid's repeated in each row instead we can use the nested and repeated type

WE CAN ACCESS THE NESTED COLUMN VALUES USING UNNEST

```
SELECT t_ouputs.output.satoshis
FROM 'publicdata.<table>'  as b
,b.transactions as t
, UNNEST(t.outputs) as t_outputs
```

RECORD REPEATED(INCLUDING LISTS)
RECORD NULLABLE(STRUCTS)(INCLUDING ONLY SINGLE VALUE OF THE SAME RECORD)

PARTITIONING TABLES:

Ingestion time:

```
Bq query --destination_table mydataset.table
--time_partitioning_type=DAY
```

Column of type DATETIME,TIMESTAMP,DATE:

```
Bq mk --table --schema a:STRING,tm:TIMESTAMP
```

--time_partitioning_field tm
Integere-typed column:
Bq mk --table --schema "customerid:integer,value:integer"
--range_partitioning=customerid,0,100,10
Mydataset.table

CLUSTERING TABLES:

```
CREATE TABLE <datset.table>
(
C1 NUMERIC,
…
)
PARTITION BY DATE(eventdatefield)
CLUSTER BY userIdfield
OPTIONS(
Partition_expiration_days=3,
Description='cluster")
AS SELECT *FROM <ds.tb>
```

In streaming tables the sorting in clustering tables fail sometime
But bq automatically reclusteers your data
WE SHOULD USE CLUSTERING WHEN WE USE PARTITIONED DATE,DATETIME,
TIMESTAMP,USE AGGREGATION OFTEN
BUT IF WE WANT TO USE WITHOUT PARTITIONING CRAETE A FAKE DATE COLUMN

BUILDING BATCH DATA PIPELINES

Saturday, July 30, 2022 4:15 PM

IN ELT:

For data quality issues do one of the following:
Setup field data type constraints,Specify fields as nullable or required,
Check for null values,put case for range values

For cleaning data use
PARSE_DATE(),SUBSTR(),REPLACE(),IFNULL(),NULLIF(),COALESCE()

Use COUNT(DISTINCT FIELD) for removing dup's

Can use hash,MD5 checksum to verify file integrity (Hash functions are (MDS(),SHA256(),SHA1(),SHA512(),
FARM_FINGERPRINT())

The Automatic process of detecting data drops and requesting data items to fill in the gaps is called backfilling.(some data transfer service use this)

IN ETL:

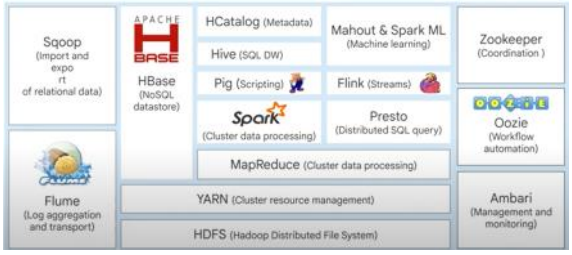
Architecture recommended is
- EXTRACT DATA FROM PUBSUB,GCS,SPANNER,SQL etc..
- TRANSFORM USING DATAFLOW
- Write to BigQuery

Dataflow to Bigtable -> for latency, throughput
Data Proc ->Reusing Spark pipelines (Hadoop ecosystem)
Data Fusion ->Need for visual pipeline building (no code) (also we can use dataprep for doing pipelines)

We can attach labels in Bigquery and in all services to be able to search form data catalog(metadata as a service) and track lineage

SPARK IN DATA PROC:

Hadoop ecosystem:
Use HDFS for distibuted file system and distributed processing happened

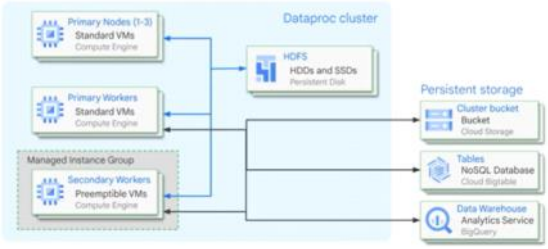


Dataproc: Build -in Hadoop/spark , Managed hardware,config , version management , low cost , fast , resizable clusters , integrated with bq,gcs etc.. ,

In DataProc by default Spark,Hive,HDFS,Pig is used others are options available

Spark (default)	Hive (default)	HDFS (default)
Pig (default)	Zeppelin	Zookeeper
Kafka	Hue	Tez
Presto	Anaconda	Cloud SQL Proxy
Jupyter	Apache Flink	Datalab
IPython	Oozie	Sqoop
Much more...		

We can use initialization actions to install additional components to the cluster
- gcloud dataproc clusters create clustername \
  --initialization-actions gs://bucket/hbase/hbase.sh \
  --num-masters 3 --num-workers 2
  https://github.com/GoogleCloudPlatform/dataproc-initialization-actions



Using data proc:

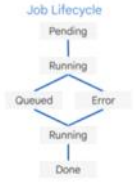
1) Setup - Create a cluster using console,gcloud,terraform,rest
2) Configure - (has options to have the particular image version,either jupyter,anaconda,docker and other extra are to be installed in the cluster etcc..)

Cluster options	Region and Zone Global or Regional endpoint Dataproc version (default is latest)	Single node (1:0) Standard (1:n) High Availability (3:n)
	Optional components	Cluster properties User Labels
Primary node options	vCPU cores Primary disk and disk type	Local SSDs
Worker nodes	Minimum number of worker nodes	VM options Initialization actions VM metadata
Preemptible nodes	Maximum number (default is 0)	

3) Optimize -

Preemptible VMs	Lower cost.
Custom machine types	Efficient allocation of resources for consistent workloads.
Minimum CPU platform	Consistent distribution of workload -minimum vCPU performance.
Custom images	Faster time to reach an operational state.
Persistent SSD boot disk	Faster boot time.
Attached GPUs	Faster processing for some workloads.
Dataproc version	Specify to prevent changes, or default to the latest.

4) Utilize -

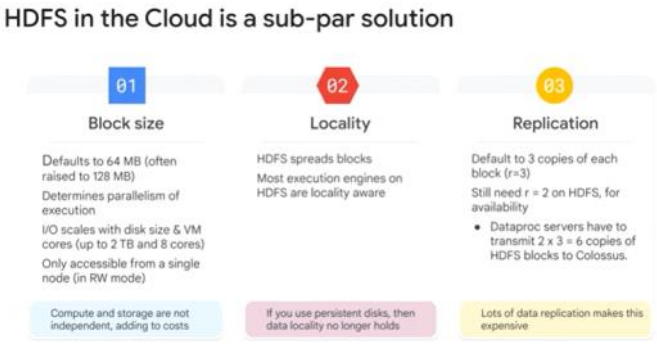


Submit a job

- Console
- gcloud command
- REST API
- Orchestration services:
  - Dataproc Workflow Templates
  - Cloud Composer

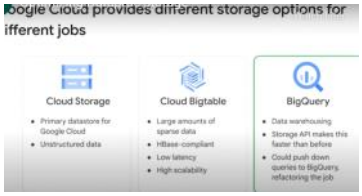
5) Monitor

Job driver output	Logs	Cloud Monitoring	Cluster Details graphs
• Gathered automatically so no need to connect to the cluster / web interface for job status	• Default to INFO • Can be set to DEBUG when job is submitted from command line	• HDFS • YARN • Job metrics • Operational metrics	• CPU utilization • Disk bytes • Disk operations • Network bytes • Network packets



With the petabit bandwidth in google dtaa center we can process the dtaa where it is without copying it

Also in the on prem hadoop clusters we can use gcs connector
If not willing to migrate fully



Local HDFS is a good option if:

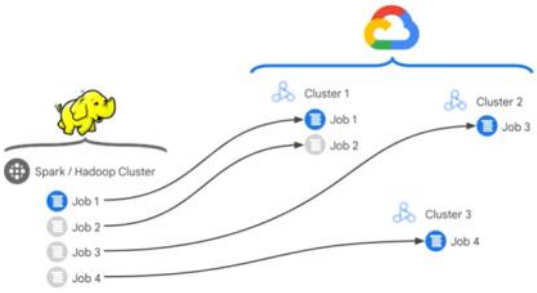
- Your jobs require a lot of metadata operations.
- You modify the HDFS data frequently or you rename directories.
- You heavily use the append operation on HDFS files.
- You have workloads that involve heavy I/O - `spark.read().write.partitionBy(...).parquet("gs://")`
- You have I/O workloads that are especially sensitive to latency.

Decrease the total size of the local HDFS by decreasing the size of primary persistent disks for the primary and workers.

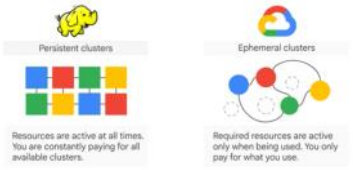
Increase the total size of the local HDFS by increasing the size of primary persistent disk for workers.

Cluster can be automatically deleted by schedule

Split clusters and jobs



With ephemeral clusters, you only pay for what you use



Dataproc Workflow Template

Dataproc workflow templates

```
# the things we need pip-installed on the cluster
STARTUP_SCRIPT=gs://$(BUCKET)/sparktobq/startup_script.sh
echo 'pip install --upgrade --quiet google-compute-engine google-cloud-storage matplotlib' >
/tmp/startup_script.sh
gsutil cp /tmp/startup_script.sh $STARTUP_SCRIPT

# create new cluster for job
gcloud dataproc workflow-templates set-managed-cluster $TEMPLATE \
    --master-machine-type $MACHINE_TYPE \
    --worker-machine-type $MACHINE_TYPE \
    --initialization-actions $STARTUP_SCRIPT \
    --num-workers 2 \
    --image-version 1.4 \
    --cluster-name $CLUSTER

# steps in job
gcloud dataproc workflow-templates add-job \
  pyspark gs://$BUCKET/spark_analysis.py \
  --step-id create-report \
  --workflow-template $TEMPLATE \
  -- --bucket=$BUCKET

# submit workflow template
gcloud dataproc workflow-templates instantiate $TEMPLATE
```

1) gcloud dataproc workflow-templates create template-id --region
2) gcloud dataproc workflow-templates set-managed-cluster template-id\
   --region=region\
   --master-machine-type=machine-type\
   --worker-machine-type=machine-type\
   --num-workers=number\
   --cluster-name=cluster-name
3) --start-after=foo in gcloud dataproc workflow-templates add-job pyspark gs://_--step-id=foo.... Means execute this job after the job with step-id foo like this we can difine multiple steps
4) Instantiate the template and it executes the jobs in the sequence we specified

Set the log level

You can set the driver log level using the following gcloud command:
`gcloud dataproc jobs submit hadoop --driver-log-levels`

To set the log level for the rest of the application from the Spark context. For example:
`spark.sparkContext.setLogLevel("DEBUG")`

Usually driver outputs
Will not be available so specify these two
<-

DATAPROC FROM SPARK:
In on prem spark jobs it would have referenced hdfc commands like hdfs:///filename there we need to copy the files to gcs and specify the hdfs to gs://bucket/filename

In jupyter workspace we can use below code in each cell to write contents to a python file created
%%writefile spark_analysis.py
python script contents

%%writefile -a spark_analysis.py # this for all the upcoming cells

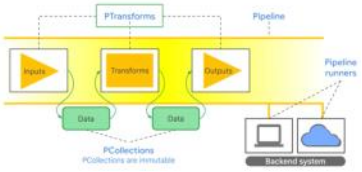
---

Spark dataproc jobs is as simple as giving a python file containing the spark code that does all required Logics to the dataproc cluster as a job to run the python file

MODES:
Single cluster mode: 1 primary worker node
Standard cluster mode : 1 primary :n workers

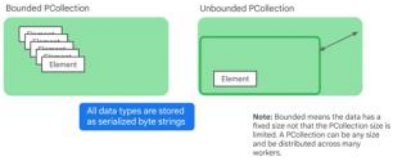
DATAFLOW:

Apache BEAM = Batch + strEAM



	Dataflow	Dataproc
Recommended for:	New data processing pipelines, unified batch and streaming	Existing Hadoop/Spark applications, machine learning/data science ecosystem, large-batch jobs, preemptible VMs
Fully-managed:	Yes	No
Auto-scaling:	Yes, transform-by-transform (adaptive)	Yes, based on cluster utilization (reactive)
Expertise:	Apache Beam	Hadoop, Hive, Pig, Apache Big Data ecosystem, Spark, Flink, Presto, Druid

Each PCollection element can be distributed for parallel processing



A Pcollection represents both batch and streaming data In apache beam dataflow
Each Pcollection element can be distributed for parallel processing
All dtypes are stored as serialized byte strings, only serialized after ptransform

After creating the beam python file that does all the transformations
We should specify the runner that is dataflow on which the beam code runs

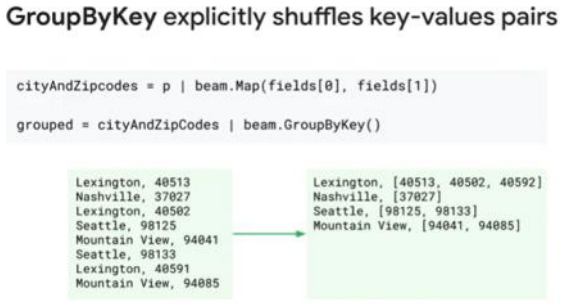
We can also run the dataflow job using cli .

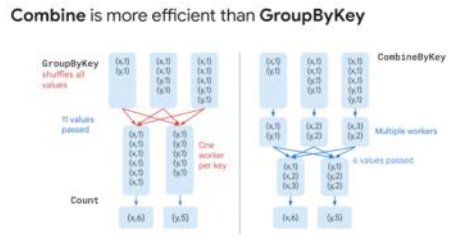
In the with beam.Pipeline(options=options) as p:
we specify the runner as dataflow in the options.

FlatMap() does the generator stuff,
ParDo does the parallel processing

Example Code:

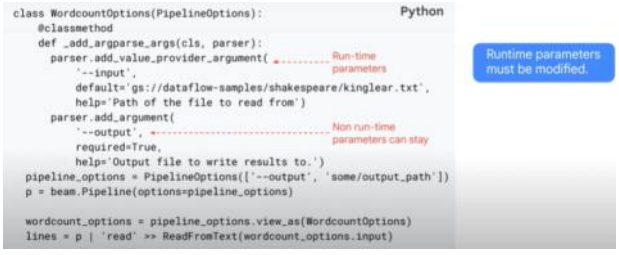
```
import apache_beam as beam

def my_grep(line, term):
  if line.startswith(term):
    yield line

PROJECT='qwiklabs-gcp-01-de1465438d5e'
BUCKET='qwiklabs-gcp-01-de1465438d5e'

def run():
  argv = [
    '--project={0}'.format(PROJECT),
    '--job_name=examplejob2',
    '--save_main_session',
    '--staging_location=gs://{0}/staging/'.format(BUCKET),
    '--temp_location=gs://{0}/staging/'.format(BUCKET),
    '--region=us-central1',
    '--runner=DataflowRunner'
  ]

  p = beam.Pipeline(argv=argv)
  input = 'gs://{0}/javahelp/*.java'.format(BUCKET)
  output_prefix = 'gs://{0}/javahelp/output'.format(BUCKET)
  searchTerm = 'import'

  # find all lines that contain the searchTerm
  (p
    | 'GetJava' >> beam.io.ReadFromText(input)
    | 'Grep' >> beam.FlatMap(lambda line: my_grep(line, searchTerm) )
    | 'write' >> beam.io.WriteToText(output_prefix)
  )

  p.run()

if __name__ == '__main__':
  run()
```





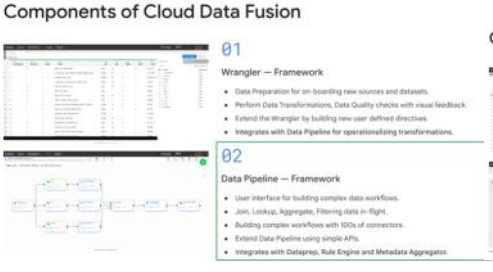
((ip1,ip2,ip3)|beam.Flatten()) peforms a union operation and flattens the three inputs into a Single pcollection
Scores|beam.Partition(partitionfn,10) # splits one into many pcollections

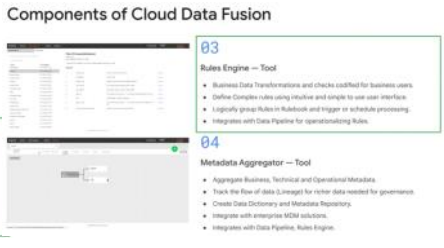
SIDE INPUT: (also we can use windowing to make the transform occur In the window specified and run every mentioned secs)

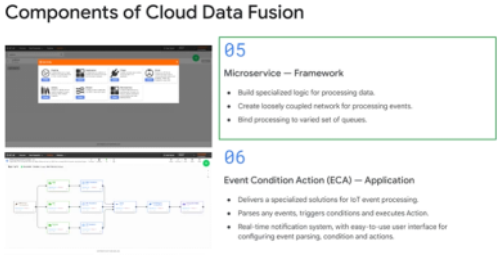


We can create our own dataflow template using the ValueProvider<>
Available in docs

CLOUD DATA FUSION:
==========
Cloud Data Fusion provides a graphical user interface and APIs that increase time efficiency and reduce complexities.







UI ELEMENTS:

- Control Center(like home page containing the apps,datasets,pipelines)
- Pipelines(developer studio,preview,export,schedule,connector,navigation)
- Wrangler(connections,transforms,data quality,Insights,Functions)

Cloud Data Fusion translates your visually built pipeline into an Apache Spark or

UI ELEMENTS:

- Control Center(like home page containing the apps,datasets,pipelines)
- Pipelines(developer studio,preview,export,schedule,connector,navigation)
- Wrangler(connections,transforms,data quality,Insights,Functions)
- Metadata(Search,Tags,Field,data)
- Hub(Plugins,Prebuild pipelines)
- Entities (like an object (pipeline,application,plugin,driver,lib,directive)
- Admin(management of everytihng)

Cloud Data Fusion translates your visually built pipeline into an Apache Spark or MapReduce program that executes transformations on an ephemeral Cloud Dataproc cluster in parallel

CLOUD COMPOSER:

For the operators available for each gcp services we can go to https://airflow.apache.org/ and search for a service in the integrations section and find the operators

In Airflow the macros.ds_add(ds,-1) # tells us that to take the date one day before the dag's scheduled run date

Two scheduling options:

1) Periodic(calendar schedules etc..)
2) Event-driven(gcp cloud functions)

FOR DATA FLOW CUSTOM TEMPLATE:

==========

In python code we get the arguments using

```
parser.add_argument(
      '--output',
      required=True,
      help='Output file to write results to.')
```

Instead need to use value provider argument

```
parser.add_value_provider_argument(
      '--input',
      default='gs://dataflow-samples/shakespeare/kinglear.txt',
      help='Path of the file to read from')
```

- After parameterizing the template using value provider
  Need to export it as a template using gcloud

```
python -m examples.mymodule\
  --runner DataflowRunner \
  --project PROJECT_ID\
  --staging_location gs://BUCKET_NAME/staging \
  --temp_location gs://BUCKET_NAME/temp \
  --template_location
gs://BUCKET_NAME/templates/TEMPLATE_NAME\
  --region REGION
```

```
class WordcountOptions(PipelineOptions):
  @classmethod
  def _add_argparse_args(cls, parser):
    # Use add_value_provider_argument for arguments to be
templatable
    # Use add_argument as usual for non-templatable arguments
    parser.add_value_provider_argument(
      '--input',
      default='gs://dataflow-samples/shakespeare/kinglear.txt',
      help='Path of the file to read from')
    parser.add_argument(
      '--output',
      required=True,
      help='Output file to write results to.')
pipeline_options = PipelineOptions(['--output',
'some/output_path'])
p = beam.Pipeline(options=pipeline_options)

wordcount_options =
pipeline_options.view_as(WordcountOptions)
lines = p | 'read' >> ReadFromText(wordcount_options.input)
```

FOR DATAFLOW FLEX TEMPLATES WE RUN ON CUSTOM DOCKER CONTAINER

Building Resilient Streaming Analytics Systems

Monday, August 1, 2022 3:31 PM

Pub/sub Topic: the source data publishes messages to this
Pub/sub subscription: the published message in the topic above is pulled/sent to subscriber of the topic
We can also apply filter to subscription in order to filter the messages the topic publishes(through gcloud,console,pubsub api)

Push:
PubSub will push the message to the chosen endpoint
Pull:
Pulling the messages from the pubsub service

Publishing messages through rest api should be base64 encoded and should have less than 10 mb of data size.

Pubsub message:(eg)
{
"data":string,
"attributes":{
String:string,
....
},
"messageId":string,
"publishTime":string
}

PUBLISHING:

Gcloud pubsub topics create test_topic
Gcloud pubsub topics publish test_topic --message "hello"
-- In the code the messages are sent through b"" and also can senf attributes like author="how"

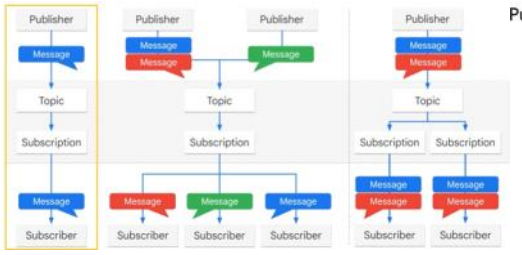
SUBSCRIPTIONS:

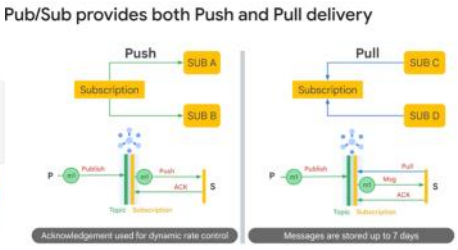
(SYNCHRONOUS)happen at the same time
Gcloud pubsub subcriptions create --topic test_topic test_sub
Gcloud pubsub subscriptions pull -auto-ack test_sub
-- In the code wither we specify a callback to retrive messages continously or specify the max messages to get and process
(sub_obj=subscriber.subscribe(callback)
Sub_obj.result()# ensures continous receive)
-- PUBSUB batches the messages we can also specify in the code to send maximum how many messages at a time

Exponential Backoff is the adding of progressive delays between retry attempts.

Pub/Sub stores your messages indefinitely until you request it

MANY PUBSUB PATTERNS:





- The unacknowledged messages will be in a snapshot

And it is deleted if it longer than 7 days we can use the snapshot
To get the unacknowledged messages

- We can also set the message retention duration to say the puubsub to retain the messages
  And give us the messages that are unacknowledged wither through snapshot or the time duration.
- If we seek into a specific time the messages after that time will be considered unacknowledged and they can be viewed by us and if we want to delete the messages we should seek in the future timing.

DATAFLOW:

In Streaming for eg the aggregation of messages happens by taking different time windows and aggregating the messages from each window and summing it up by dataflow automatically.We can also modify the timestamp of the incoming messages customly.

KINDS OF WINDOWS:

- Fixed (windowing divides data into time based finite chunks(hourly slice,daily slice)
- Sliding(eg: give me 30 minutes worth of data and compute every 5 minutes)
- Session(eg: user comes and visites 4-5 pages and leaves (session window))

Fixed-time windows

```
from apache_beam import window                                    Python
fixed_windowed_items = (
    items | 'window' >> beam.WindowInto(window.FixedWindows(60)))
```

WINDOW STARTING EVERY 60 SECS

Sliding time windows

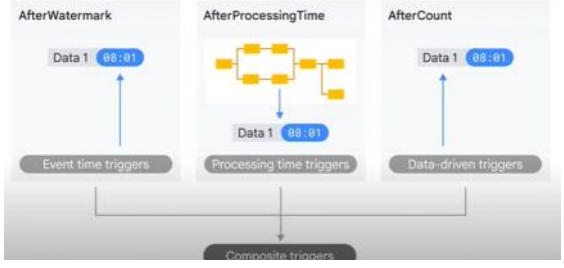
```
from apache_beam import window                                    Python
sliding_windowed_items = (
    items | 'window' >> beam.WindowInto(window.SlidingWindows(30, 5)))
```

(WINDOW TIME,HOW OFTEN RUNS)

Session windows

```
from apache_beam import window                                    Python
session_windowed_items = (
    items | 'window' >> beam.WindowInto(window.Sessions(10 * 60)))
```

600 SECONDS WINDOW SESSIONS



Session windows

```python
from apache_beam import window
session_windowed_items = (
    items | 'window' >> beam.WindowInto(window.Sessions(10 * 60)))
```

600 SECONDS WINDOW SESSIONS

Dataflow does the watermarking by calculating the lag time between pubsub essage and the processing time for flushing the window effectively

Trigger is for how to control and when the results in that window should be processed

```
pcollection | WindowInto(
    SlidingWindows(60, 5),                          # Sliding window of 60 seconds, every 5 seconds
    trigger=AfterWatermark(                         # Relative to the watermark, trigger:
      early=AfterProcessingTime(delay=30),          # -- fires 30 seconds after pipeline commences
      late=AfterCount(1))                           # -- and for every late record (< allowedLateness)
    accumulation_mode=AccumulationMode.ACCUMULATING)   # the pane should have all the records
    allowed_lateness=Duration(seconds=2*24*60*60))     # 2 days
```

<- this example is telling that we can create a sliding window of 60 seconds every 5 seconds
Were the trigger is based on after watermark period and for early arrivals process after 30 seconds
And for late arrivals process after accumulated 1 record(all records)

```
pcollection | WindowInto(
    FixedWindows(60),                               # Fixed window of 60 seconds
    trigger=Repeatedly(                             # Set up a composite trigger that triggers ...
        AfterAny(                                   # whenever either of these happens:
            AfterCount(100),                        # -- 100 elements accumulate
            AfterProcessingTime(1 * 60))),          # -- every 60 seconds (ignore watermark)
    accumulation_mode=AccumulationMode.DISCARDING)  # the trigger should be with only new records
```

<- this is telling to trigger whenever In the 60 second window 100 elements are acumulated
Every 60 seconds and the trigger should be new records not all records as above.

THE SOURCE COMMAND READS AND EXECUTES COMMAND IN THE FILES SPECIFIED IN THE UNIX SHELL ENV

STREAMING DATA IN BIGQUERY VIA STREAMING INSERTS :

Data enters a streaming buffer to insert rows
Interactive sql queries over pb's in seconds
DON'T POPULATE INSERTID TO GET STREAMING INGESTS QUOTAS FOR US REGIONS
In python using bigquery_client.insert_rows(table_id,rows) does the streaming stuffs

In data studio we can use custom query to build dashboards

CLOUD BIGTABLE:

LOW LATENCY WITH NOSQL IN MILLISECONDS

- In this only the row key is the only index available in big table databases
- Bigtable rebalances the data based on the various data access patterns

BIGTABLE PERFORMANCE TUNING:

- Change the schema for better performance
- Test with more data in order to bigtable to learn to improve its read/write
- Client should be in the same region for maximizing performance and also hdd-ssd will Also help for speed
- More nodes => more performance

Gcloud bigtable clusters creeate cluster-name
--instance=INSTANCE_ID
--zone=ZONE
--num-nodes=3
--storage-type=SSD

ANALYTIC WINDOW FUNCTIONS:

. Standard aggregations
. Navigation functions
. Ranking and numbering functions

Example: LEAD function in bigquery returns the value of a row n rows ahead of the current row
Select start_date,end_dtae,LEAD(start_date,1) as rental
From <table>
Op: start_date=>2022-04-05
rental => 2022-05-05
Navigation functions:

- LEAD
- NTH_VALUE
- LAG
- FIRST_VALUE
- LAST_VALUE

THE RANK FUNCTION RANK() OVER(PARTITION BY COL ORDER BY COL DESC) AS SOMETHING RETURNS
THE COL IN DESCENCING ORDER
SOME OTHER RANKING FUCNTIONS:

==========

- CUME_DIST
- DENSE_RANK
- ROW_NUMBER
- PERCENT_RANK

GIS(GEOGRAPHIC INFORMATION SYSTEMS)

-==========-

ST_Dwithin(geography1,geography2,distance)

- To see whether the given locations are at particular distance mentioned
- Eg: ST_Dwithin(zipcode_geom,STGeogPoint(logitude,latitude),1000)

ST_GeogPoint(lat,long)

- Provides well known text for the location(POINT(10.333 -120.33))
- It can be tested in bigquery geo vix provided for the values
- If the data is json can use ST_GeogFromGeoJSON()

ST_Makeline(location1,location2),ST_MakePolygon([loc1,loc2,loc3])

- Provide the locations and represen a region

ST_Intersects(),ST_Contains(),ST_CoveredBy()

- Gives us the result whether the two given points intersect

Can use windowing functions like timestamp_diff(_,lag(time) , _) etc..

PUT THE LARGEST TABLE ON THE LEFT ON JOINING TWO TABLES

USE APPROX_COUNT_DISTINCT INSTEAD OF COUNT(DISTINCT) WHICH IS JS UDF

Use row_number for sorting numbers writing
ROW_NUMBER() OVER(ORDER BY end_date) AS rental_number

- Use cache queries whenever can
- Instead of joining everything in a single query create intrmediate tables
- Selec tonly required columns or use except to except some columns
- Effective joins

Flex slots in bigquery charged 4 cents per slot per hour and it is flexible(bq reservation slots)
Monthly commitments(flex slots) cannot be cancelled for 30 calendar days
Annual commitements cannot be cancellled fpr one calendar year
(Flat rate and on demand pricing can be used together)

Eg: for 50 queries => 2000 slots can be used
When used hierarchial reservation of slots it is good for multiple projects

```
PARTITION BY date
 OPTIONS (
   partition_expiration_days=60,
   description="weather stations with precipitation, partitioned by day"
 ) AS
```

Partition_expiry_days is the living of the table

Smart Analytics, Machine Learning, and AI

Tuesday, August 2, 2022 12:56 PM

Content is categorized into 700+ possible categories in natural language api(used for classifying unstructured text)

NATURAL LANGUAGE API:

- Create an api key from api and services ->credentials page
- After that create the json file that gets passed in the curl command
- JSON:

```
{
    "document":{
      "type":"PLAIN_TEXT",
      "content":"A Smoky Lobster Salad With a Tapa Twist. This spin on the Spanish pulpo a la gallega skips the octopus, but keeps the sea salt, olive oil, pimentón and boiled potatoes."
    }
}
```

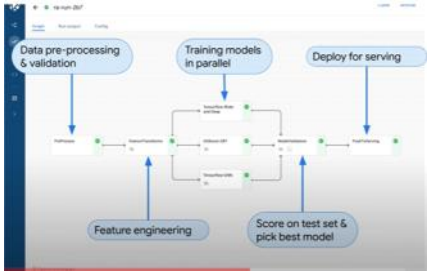
- Curl command
  curl "https://language.googleapis.com/v1/documents:classifyText?key=${API_KEY}" \
  -s -X POST -H "Content-Type: application/json" --data-binary @request.json
  Export the api key in API_KEY
- The same curl command post request can be done in python
  Eg:

```
from google.cloud import language_v1
nl_client = language_v1.LanguageServiceClient()
response = nl_client.classify_text(document=language_v1.types.Document(content=article,type_='PLAIN_TEXT'))
```

In notebooks magic commands are used to execute sql query in notebook
Cell or other possible implementations can be made like running bigquery queries
%%bigquery # %%bigquery df - to save it to dataframe
Sql query

You can easily change hardware including adding and removing GPUs

KUBEFLOW:



We can also upload and execute pipelines via UI
(package and sharing pipelines as zip files (via AI HUB)https://aihub.cloud.google.com/u/0/)
AI Hub stores various asset types

Assets are stored as either public or restricted assets

- Kubeflow pipelines and components
- Jupyter notebooks
- TensorFlow modules
- Trained models
- Services
- VM images

more pipeline.json is the command where the output is shown as we enter ENTER like 10% is shown and
20% is shown etc.....

VERTEX AI (your model,your data)
AUTO ML (our model,your data)
PREBUILT API (OUR MODEL,OUR DATA)

In bigquery ML we can also specify the tensorflow model path to build in bigquery
Model_type="tensorflow"
Model_path="gs://path"

Recommendation engine (matrix factorization)

```
create or replace model models.suggested_products_1or2_example
options(model_type='matrix_factorization',
        user_col='user_id', item_col='product_id', rating_col='rating',
        l2_reg=10)
AS

with purchases AS (
 select product_id, user_id from
 operations.orders_with_lines, unnest(order_lines)
),

total_purchases as (
select product_id, user_id, count(*) as numtimes
from purchases
group by product_id, user_id
)

select
product_id, user_id,
IF(numtimes < 2, 1, 2) AS rating
FROM total_purchases
```

Clustering

```
CREATE OR REPLACE MODEL
demos_eu.london_station_clusters
OPTIONS(model_type='kmeans', num_clusters=4,
standardize_features = true) AS


WITH hs AS …,
stationstats AS …

SELECT * except(station_name, isweekday)
from stationstats
```

1. 4 clusters (hardcoded)
2. Standardize features since different dynamic ranges
3. Remove the cluster "id" fields (keep just the attributes)

Find cluster attributes

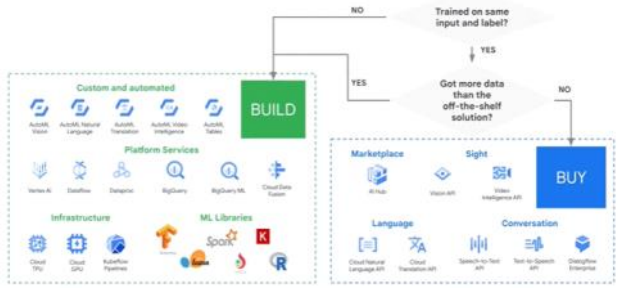
```
WITH T AS (
SELECT
centroid_id,
ARRAY_AGG(STRUCT(numerical_feature AS name, ROUND(feature_value,1) AS value) ORDER BY
centroid_id) AS cluster
FROM ML.CENTROIDS(MODEL demos_eu.london_station_clusters)
GROUP BY centroid_id
)
SELECT
CONCAT('Cluster#', CAST(centroid_id AS STRING)) AS centroid,
(SELECT value from unnest(cluster) WHERE name = 'duration') AS duration,
(SELECT value from unnest(cluster) WHERE name = 'num_trips') AS num_trips,
(SELECT value from unnest(cluster) WHERE name = 'bikes_count') AS bikes_count,
(SELECT value from unnest(cluster) WHERE name = 'distance_from_city_center') AS
distance_from_city_center
FROM T
ORDER BY centroid_id ASC
```

- **Label** = `alias a column as 'label' or specify column in OPTIONS using input_label_cols`
- **Feature** = passed through to the model as part of your SQL SELECT statement
  `SELECT * FROM ML.FEATURE_INFO(MODEL `mydataset.mymodel`)`
- **Model** = an object created in BigQuery that resides in your BigQuery dataset
- **Model Types** = Linear Regression, Logistic Regression
  ```
  CREATE OR REPLACE MODEL <dataset>.<name>
  OPTIONS(model_type='<type>') AS
  <training dataset>
  ```
- **Training Progress** = `SELECT * FROM ML.TRAINING_INFO(MODEL `mydataset.mymodel`)`
- **Inspect Weights** = `SELECT * FROM ML.WEIGHTS(MODEL `mydataset.mymodel`, (<query>))`
- **Evaluation** = `SELECT * FROM ML.EVALUATE(MODEL `mydataset.mymodel`)`
- **Prediction** = `SELECT * FROM ML.PREDICT(MODEL `mydataset.mymodel`, (<query>))`

AUTOML:
=======
- Natural Language
- Tables
- Vision

THE CSV FILE FOR DATASET PREP IT SHOULD BE UTF-8 ENCODED
THE IMAGE FOR VISION SHOULD BE BASE64 ENCODED(MAX 30MB)
GIVE VARIETY OF IMAGES TO MAKE THE MODEL WORK GOOD
NO UNICODE CHARACTERS IS SUPPORTED WHEN WORIING WITH TXT
IN MODEL BUILDING

As a data engineer should you build or buy a solution?



TO RESERVE THE MODEL TRAINED WE NEED TO TRAIN THE MODEL AGAIN
ELSE IT WILL BE DELETED AFTER A CERTAIN PERIOD

IN AUTOML ENSURE CONSISTENT LABELLING THAN INCREASING THE LABELS

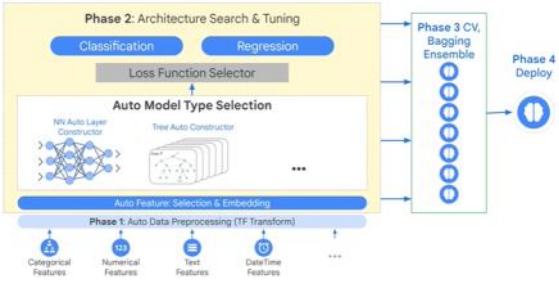
```
SELECT
 *
FROM
 ML.RECOMMEND(MODEL `cloud-training-prod-bucket.movies.movie_recommender`)
LIMIT
 100000
```

for recommending the movie

Bucketizing the data using bigquery ML

```
CREATE OR REPLACE MODEL
 bike_model.model_bucketized TRANSFORM(* EXCEPT(start_
date),
 IF
  (EXTRACT(dayofweek
    FROM
     start_date) BETWEEN 2 AND 6,
   'weekday',
   'weekend') AS dayofweek,
  ML.BUCKETIZE(EXTRACT(HOUR
    FROM
     start_date),
    [5, 10, 17]) AS hourofday )
OPTIONS
 (input_label_cols=['duration'],
  model_type='linear_reg') AS
SELECT
 duration,
 start_station_name,
 start_date
FROM
 `bigquery-public-data`.london_bicycles.cycle_hire
```

AutoML Table is for structured data



CREATE AND MANAGE CLOUD RESOURCES

Friday, August 5, 2022 8:16 PM

gcloud compute project-info describe --project $(gcloud config get-value project)
- Get the value of the project and descrube it

gcloud compute instances list --filter="name=('gcelab2')" # filtering using the attributes available when listing the compute instances("name=('') AND other=other") alse can be used

gcloud compute firewall-rules list #List the Firewall rules in the project

gcloud logging read "resource.type=gce_instance AND labels.instance_name='gcelab2'"--limit 5

Gcloud -h or gcloud --help

gcloud config list # list the configurations

gcloud components list #list the components available in gcloud sdk

WINDOWS SERVER:
- gcloud compute instances get-serial-port-output instance-1

to see whether the windows vm is ready to accept connection
- gcloud compute reset-windows-password [instance] --zone us-east1-b --user [username]

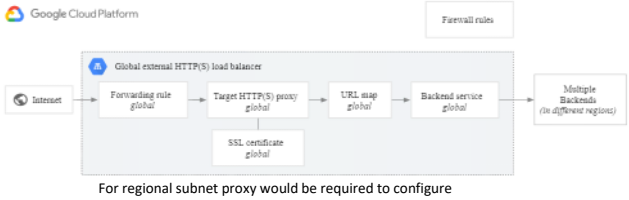
to setup password(user and password will be displayed
- Click donwload rdp or got to rdp and type the vm external ip address and user ,pass and go inside

USE
Spark View extension for non windows users

DEPLOYING APP TO CLUSTER GKE:
- gcloud container clusters get-credentials lab-cluster
- kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
- kubectl expose deployment hello-server --type=LoadBalancer # creates a gce loadbalancer for your app
  --port 8080 # port that the app container runs on
- kubectl get service # for listing the deployments
- Fot o http:EXTERNAL_IP_ADDRESS:port to go to the app

- kubectl delete deployment hello-server
- kubectl delete service hello-server
- gcloud container clusters delete lab-cluster

LOAD BALANCING THE WEB SERVERS:
- Create three web servers in GCE with target tag as same to allow http traffic
  Eg:
  gcloud compute instances create www1 \
  --zone=us-central1-f \
  --tags=network-lb-tag \
  --machine-type=e2-medium \
  --image-family=debian-11 \
  --image-project=debian-cloud \
  --metadata=startup-script='#!/bin/bash
  apt-get update
  apt-get install apache2 -y
  service apache2 restart
  echo "
  <h3>Web Server: www1</h3>" | tee /var/www/html/index.html'
- Create a firewall rule with that target tag
  gcloud compute firewall-rules create www-firewall-network-lb --target-tags network-lb-tag --allow tcp:80
- Create a static address for the load balancer
  gcloud compute addresses create network-lb-ip-1 \
  --region us-central1
- gcloud compute http-health-checks create basic-check
- gcloud compute target-pools create www-pool \ # region should be same as the vm instances
  --region us-central1 --http-health-check basic-check
- gcloud compute target-pools add-instances www-pool \
  --instances www1,www2,www3
- gcloud compute forwarding-rules create www-rule \
  --region us-central1 \
  --ports 80 \
  --address network-lb-ip-1 \
  --target-pool www-pool
- gcloud compute forwarding-rules describe www-rule --region us-central1 # curling the ip address of this forwarding rule will direct traffic to three instnces randomly



For regional subnet proxy would be required to configure

FOR HTTP LOAD BALANCING USING MIG:
- gcloud compute instance-templates create lb-backend-template \
  --region= \
  --network=default \
  --subnet=default \
  --tags=allow-health-check \
  --machine-type=e2-medium \
  --image-family=debian-11 \
  --image-project=debian-cloud \
  --metadata=startup-script='#!/bin/bash
  apt-get update
  apt-get install apache2 -y
  a2ensite default-ssl
  a2enmod ssl
  vm_hostname="$(curl -H "Metadata-Flavor:Google" \
  http://169.254.169.254/computeMetadata/v1/instance/name)"
  echo "Page served from: $vm_hostname" | \
  tee /var/www/html/index.html
  systemctl restart apache2'

  <- this is creating a instance template having a starting script,target network tags
- gcloud compute instance-groups managed create lb-backend-group \
  --template=lb-backend-template --size=2 --zone=

  < - this will create a mig with 2 instances
- gcloud compute firewall-rules create fw-allow-health-check \
  --network=default \
  --action=allow \
  --direction=ingress \
  --source-ranges=130.211.0.0/22,35.191.0.0/16 \
  --target-tags=allow-health-check \
  --rules=tcp:80

  <- create a firewall rule for the tag defined in te4mplate to allow tcp 80
- gcloud compute addresses create lb-ipv4-1 \
  --ip-version=IPV4 \
  --global

  <- create a ip address for the load balancer frontedn
- gcloud compute health-checks create http http-basic-check \
  --port 80

  <- health check for restarting vm's when unhealthy
- gcloud compute backend-services create web-backend-service \
  --protocol=HTTP \
  --port-name=http \
  --health-checks=http-basic-check \
  --global

  <- creating a backedn service and attaching the mig into this
- gcloud compute backend-services add-backend web-backend-service \
  --instance-group=lb-backend-group \
  --instance-group-zone= \
  --global
- gcloud compute url-maps create web-map-http \
  --default-service web-backend-service

  <- to map the incoming urls according to their paths
- gcloud compute target-http-proxies create http-lb-proxy \
  --url-map web-map-http
- gcloud compute forwarding-rules create http-content-rule \
  --address=lb-ipv4-1\
  --global \
  --target-http-proxy=http-lb-proxy \
  --ports=80

  <- this is the front edn of the load balancwr to use with static ip's

- URL map is a Google Cloud configuration resource used to route requests to backend services or backend buckets. For example, with an external HTTP(S) load balancer, you can use a single URL map to route requests to different destinations based on the rules configured in the URL map:
  - Requests for https://example.com/video go to one backend service.
  - Requests for https://example.com/audio go to a different backend service.
  - Requests for https://example.com/images go to a Cloud Storage backend bucket.
  - Requests for any other host and path combination go to a default backend service.

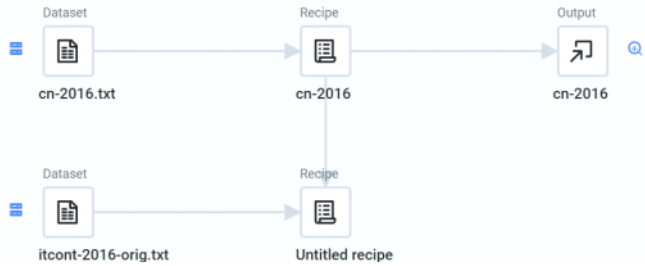
CREATING SERVICE ACCOUNT SUSING GCLOUD:

```
SERVICE_ACCOUNT_ID=vertex-custom-training-sa
- gcloud iam service-accounts create $SERVICE_ACCOUNT_ID  \
    --description="A custom service account for Vertex custom training with Tensorboard" \
    --display-name="Vertex AI Custom Training"
  PROJECT_ID=$(gcloud config get-value core/project)
- gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member=serviceAccount:$SERVICE_ACCOUNT_ID@$PROJECT_ID.iam.gserviceaccount.com \
    --role="roles/storage.admin"
```

Foundational Data, ML, and AI Tasks

Saturday, August 6, 2022 5:07 PM

DATAPREP:
-========-
Flow we create
- Datasets as input
- We click each dataset after importing and give recipies(steps to perform on the dataset)
- We do this for all the datsets and after that the result will be output(bq,gcs,etcc.)
- Output will be run on two platforms(dataflow,Trifacta Photon)
- We can change the file name at runtime or same filename or table name etcc.



DATAFLOW:
==========
- Using templates(pubsub to bq)
- Using python
  python -m apache_beam.examples.wordcount --project $DEVSHELL_PROJECT_ID \
  --runner DataflowRunner \
  --staging_location $BUCKET/staging \
  --temp_location $BUCKET/temp \
  --output $BUCKET/results/output \
  --region us-west1
- When running with batch text files to gcs should provider udf whicih returns line by line formatted Json file for the line's data for ready to insert to bq
  gs://cloud-training/gsp323/lab.js

DATAPROC:
==========
- Create a cluster on which we run the job
- JOB: create a job with like this in that cluster

Region	us-west4
Cluster	example-cluster
Job type	Spark
Main class or jar	org.apache.spark.examples.SparkPi
Jar files	file:///usr/lib/spark/examples/jars/spark-examples.jar
Arguments	1000 (This sets the number of tasks.)

- Using gcloud

1) gcloud dataproc clusters create example-cluster --worker-boot-disk-size 500
2) gcloud dataproc jobs submit spark --cluster example-cluster \
   --class org.apache.spark.examples.SparkPi \
   --jars file:///usr/lib/spark/examples/jars/spark-examples.jar -- 1000
3) gcloud dataproc clusters update example-cluster --num-workers 4

CLOUD NATURAL LANGUAGE API:
==========
- Create service account and make request using env variables for auth
- gcloud ml language analyze-entities --content="Michelangelo Caravaggio, Italian painter, is known for 'The Calling of Saint Matthew'." > result.json

Result:

```
{
  "entities": [
    {
      "name": "Michelangelo Caravaggio",
      "type": "PERSON",
      "metadata": {
        "wikipedia_url": "http://en.wikipedia.org/wiki/Caravaggio",
        "mid": "/m/020bg"
      },
      "salience": 0.83047235,
      "mentions": [
        {
          "text": {
            "content": "Michelangelo Caravaggio",
            "beginOffset": 0
          },
          "type": "PROPER"
        },
        {
          "text": {
            "content": "painter",
            "beginOffset": 33
          },
          "type": "COMMON"
        }
      ]
    },
      ],
  "language": "en"
}
```


CLOUD SPEECH API:
==========
- Create an api key in the api&services>credentials->create credentials
- export API_KEY=apikeycreated
- Create a request.json file and give this
  ```
  {
    "config": {
        "encoding":"FLAC",
        "languageCode": "en-US"
    },
    "audio": {
        "uri":"gs://cloud-samples-tests/speech/brooklyn.flac"
    }
  }
  ```
- curl -s -X POST -H "Content-Type: application/json" --data-binary @request.json \
  "https://speech.googleapis.com/v1/speech:recognize?key=${API_KEY}" # also op via >result.json
- Result:
  ```
  {
    "results": [
      {
        "alternatives": [
          {
            "transcript": "how old is the Brooklyn Bridge",
            "confidence": 0.9828748
          }
        ],
        "resultEndTime": "1.770s",
        "languageCode": "en-us"
      }
    ],
    "totalBilledTime": "15s"
  }
  ```

VIDEO INTELLIGENCE API:
==========
- gcloud iam service-accounts create quickstart
- gcloud iam service-accounts keys create key.json --iam-account quickstart@qwiklabs-gcp-01-d2262afdb762.iam.gserviceaccount.com
- gcloud auth activate-service-account --key-file key.json
- gcloud auth print-access-token # this will give token for your service account
- cat > request.json <<EOF
  ```
  {
    "inputUri":"gs://spls/gsp154/video/train.mp4",
    "features": [
      "LABEL_DETECTION"
    ]
  }
  ```
- curl -s -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer '$(gcloud auth print-access-token)'' \
  'https://videointelligence.googleapis.com/v1/videos:annotate' \
  -d @request.json
  ```
  {
   "name": "projects/1076404650911/locations/asia-east1/operations/6901329292829361619"
  }
  ```
  (this will give us operation id that we will use to curl in the next step)
- curl -s -H 'Content-Type: application/json' \
  -H 'Authorization: Bearer '$(gcloud auth print-access-token)'' \

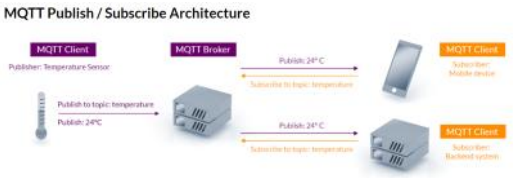
'https://videointelligence.googleapis.com/v1/projects/PROJECTS/locations/LOCATIONS/operations/OPERATION_NAME'

Replace the corresponding values obtained in the previous step

- This will provide us the timestamp level annotations as the response

ENGINEER DATA ON GCP

Saturday, August 6, 2022 8:08 PM



IOT CORE:
========

https://eclipse.org/paho/clients/python/docs/

- Create a pubsub topic
- Create a bq table
- Create a gcs bucket
- Create a data flow streaming pipeline from topic to bq

You must create a registry for the iot devices. The registry is a point of control for devices

MQTT (MQ Telemetry Transport)

- gcloud iot registries create iotlab-registry \
  --project=$PROJECT_ID \
  --region=$MY_REGION \
  --event-notification-config=topic=projects/$PROJECT_ID/topics/$TOPIC_ID
  <- creates a registry in IOT core for all the iot devices we need to use
- git clone http://github.com/GoogleCloudPlatform/training-data-analyst
- cd $HOME/training-data-analyst/quests/iotlab/
- openssl req -x509 -newkey rsa:2048 -keyout rsa_private.pem \
  -nodes -out rsa_cert.pem -subj "/CN=unused"
  <- This openssl command creates an RSA cryptographic keypair and writes it to a file called rsa_private.pem and public key file rsa_cert.pem
- CREATING A DEVICE IN THE REGISTRY
  gcloud iot devices create temp-sensor-buenos-aires \
  --project=$PROJECT_ID \
  --region=$MY_REGION \
  --registry=iotlab-registry \
  --public-key path=rsa_cert.pem,type=rs256
  <- we will use the public key generated to create a device in the registry
- enter these commands to download the CA root certificates from pki.google.com
  cd $HOME/training-data-analyst/quests/iotlab/
  curl -o roots.pem -s -m 10 --retry 0 "https://pki.goog/roots.pem"
- RUN THE SIMULATED DEVICE IN THE BACKGROUND
  python cloudiot_mqtt_example_json.py \
  --project_id=$PROJECT_ID \
  --cloud_region=$MY_REGION \
  --registry_id=iotlab-registry \
  --device_id=temp-sensor-buenos-aires \
  --private_key_file=rsa_private.pem \
  --message_type=event \
  --algorithm=RS256 > buenos-aires-log.txt 2>&1 &
  <- the .py file creates the client using the project id,region,registryid,deviceid and authenticates using jwt without username only using password (password=create_jwt(args.project_id, args.private_key_file, args.algorithm))
  And register message callbacks
  client.on_connect = on_connect
  client.on_publish = on_publish
  client.on_disconnect = on_disconnect

  Also it uses roots.pem certificate to Enable SSL/TLS support and connects to mqtt bridge
  client.connect(mqtt.googleapis.com,8883)
  And using mqtt_topic = '/devices/{}/{}'.format(args.device_id, sub_topic)
  And client.publish(mqtt_topic,payload) to send the data
- python cloudiot_mqtt_example_json.py \
  --project_id=$PROJECT_ID \
  --cloud_region=$MY_REGION \
  --registry_id=iotlab-registry \
  --device_id=temp-sensor-istanbul \
  --private_key_file=rsa_private.pem \
  --message_type=event \
  --algorithm=RS256

DATAFLOW PYTHON EXAMPLES LAB:
==========

gsutil -m cp -R gs://spls/gsp290/dataflow-python-examples .

COMPOSER:
==========

gcloud composer environments run ENVIRONMENT_NAME \
--location LOCATION variables -- \
set KEY VALUE

- this is for creating the variables in composer env
- gsutil -m cp -r gs://spls/gsp283/python-docs-samples .
  python-docs-samples/composer/workflows/

DATAPREP:
==========

flow_Ecommerce_A...

VERTEX AI:
========

lab_exercise

```
create or replace table taxirides.taxi_training_data_641 as
select pickup_datetime,pickup_latitude,pickup_longitude,
dropoff_latitude,dropoff_longitude,passenger_count,tolls_amount+fare_amount as fare_amount_580 from
taxirides.historical_taxi_rides_raw TABLESAMPLE SYSTEM (50 PERCENT)
where (pickup_longitude BETWEEN -180 AND 180 )
AND  (dropoff_longitude between -180 and 180)
and (pickup_latitude between -90 and 90)
and (dropoff_latitude between -90 and 90)
and( trip_distance >4)
and (fare_amount >2.5)
and passenger_count>4
```

```
CREATE OR REPLACE MODEL
taxirides.fare_model_299
TRANSFORM(
   EXTRACT (DAY FROM pickup_datetime) as day,
EXTRACT (MONTH FROM pickup_datetime) as month,passenger_count,fare_amount_
580,ST_Distance(ST_GeogPoint(pickup_longitude, pickup_latitude), ST_GeogPoint(dropoff_longitude, dropoff_latitude)) AS euclidean
)
OPTIONS(
  model_type='linear_reg',
  input_label_cols=['fare_amount_580']
)
AS SELECT * FROM
taxirides.taxi_training_data_641
```

```
SELECT * FROM
```

```
ML.EVALUATE(MODEL taxirides.fare_model_299);
select EXTRACT (DAY FROM pickup_datetime) as day,
EXTRACT (MONTH FROM pickup_datetime) as month from `taxirides.taxi_training_data_641`
```

ROOT MEAN SQUARE ERROR(RMSE) SHOULD BE LOWEST POSSIBLE AFTER CREATING MODEL(LESS THAN 10) TO BE ABLE TO PERFORM WELL

MORE DATA,GOOD USABLE FEATURES=GOOD MODEL

INCREASE THE DATA VARIETY INSTEAD OF SAME KIND OF DATA

```
CREATE TABLE taxirides.2015_fare_amount_predictions as
SELECT * FROM
ML.PREDICT(MODEL taxirides.fare_model_299,
(select pickup_datetime,passengers as passenger_Count,pickuplon as pickup_longitude, pickuplat as pi
ckup_latitude,
dropofflon as dropoff_longitude, dropofflat as dropoff_latitude
from taxirides.report_prediction_data))
```

WORKED OUT SETUFFS IN LAB:

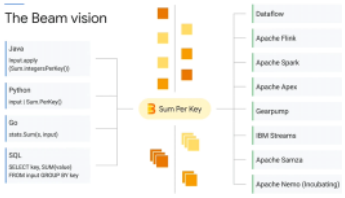
==========

```
create or replace table taxirides.taxi_training_data_752
as
select pickup_datetime,pickup_latitude,pickup_longitude,
dropoff_latitude,dropoff_longitude,passenger_count,tolls_amount+ fare_amount as fare_amount_393 from
taxirides.historical_taxi_rides_raw
where --(pickup_longitude BETWEEN -180 AND 180 )
-- AND
--(dropoff_longitude between -180 and 180)
(pickup_latitude between -90 and 90)
and (dropoff_latitude between -90 and 90)
-- and( trip_distance >1)
-- and (fare_amount >2)
-- and passenger_count>1 limit 1000000
```

```
CREATE or REPLACE MODEL taxirides.fare_model_280
OPTIONS
  (model_type='linear_reg', labels=['fare_amount_393']) AS
WITH
  daynames AS
    (SELECT ['Sun', 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat'] AS daysofweek),
  taxitrips AS (
  SELECT
    fare_amount_393,
    daysofweek[ORDINAL(EXTRACT(DAYOFWEEK FROM pickup_datetime))] AS dayofweek,
    EXTRACT(HOUR FROM pickup_datetime) AS hourofday,
    ST_Distance(ST_GeogPoint(pickup_longitude, pickup_latitude),
    ST_GeogPoint(dropoff_longitude, dropoff_latitude)) AS euclidean,
    passenger_count AS passengers
  FROM
    `taxirides.taxi_training_data_847`, daynames
    -- WHERE
    -- MOD(ABS(FARM_FINGERPRINT(CAST(pickup_datetime AS STRING))),1000000) = params.TRAIN
  )
  SELECT *
  FROM taxitrips
```

DATAFLOW FOUNDATIONS

Sunday, August 7, 2022 6:35 PM



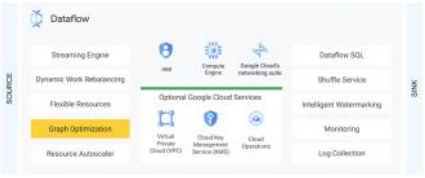
To be able to run in an custom container enviroment
Need to pass an argument

- Build docker image with python sdk
- Push the image to gcr and run the following
- Python script.py --input=,--output=,project,region,temp_location,--runner=DataflowRunner --worker_harness_container_image=gcr.io/project/theimagepushed

Cross language transforms:

- The kafka is available in java
- But it can be imported for use using apachebeam.io.kafka
- Internally it connects with java sdk and transforms
- Then at runtime after giving it to dataflow service two worker vm's are created one for java and another for python simultaneoulsy

The Google Cloud runner: Dataflow



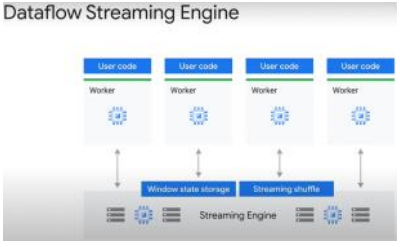
Dataflow Shuffle service: Batch

(Only for batch)
Used while doing groupbykey,cogroupbykey
Combinebykey etc..
(by partitioning and sorting the data)

- To activate this mode use this while giving job to dataflow using cli

--experiments=shuffle_mode=service (us-central1)

Dataflow Streaming Engine



FLEXRS(FLEXIBLE RESOURCE SCHEDULING:

- Uses advanced scheduling techniques using Dataflow shuffle service,preemptible+regular VM'S
- Used for daily,weekly and for those that does not require
- Immediate start because scheduling delay(within 6 hours) is there
- To enable
- --flexrs_goal=COST_OPTIMIZED or SPEED_OPTIMIZED

- Dataflow streaming engine and shiffling services moves operations from worker VMs into a Cloud Dataflow backend service
- For streaming engine enable --experiments=enable_streaming_engine --region={ us-central1 | europe-west1 }

IAM for dataflow:

- when user submits the code it is submitted to dataflow
And after that also the code is in gcs/put by dataflow
And fataflow creates gce vm's for processing for all these IAM is
required

Credentials playing role in dataflow job:

---

1) User roles
2) Dataflow service account
3) Controller service account

USER ROLES: dataflow Viewer(only view)
dataflow developer(view,update,cancel jobs)
Dataflow admin(create,manage)

DATAFLOW SERVICE ACCOUNT: used for worker creation,monitoring (has dataflow service agent role)

CONTROLLER SERVICE ACCOUNT: used by the workers to access resources needed by the pipeline(compute engine service account)

Quotas:

- CPU (cores)
- IP's In-use IP address limit in gce api
- Persistent disks(hdd or ssd) limit
- For ssd pd-ssd for hdd pd-standard

Persistent Disks - Batch Pipeline

- VM to PD ratio is 1:1 for Batch
- Size if Shuffle on VM: 250 GB
- Size if Shuffle Service: 25 GB
- Flag to override default:
  Python: --disk_size_gb
  Java: --diskSizeGb

Persistent Disks - Streaming Pipeline

- Fixed number of PDs
- Default size if shuffle on VM: 400 GB
- Default size if Streaming Engine: 30 GB
- Flag to override default:
  Python: --disk_size_gb
  Java: --diskSizeGb

Flag to override default:

- Python: --service_account_email
- Java: --serviceAccount

(should at least have dataflow worker role)

Persistent Disks

Python: Use the --worker_disk_type flag

```
$ python3 -m apache_beam.examples.wordcount \
  --input gs://dataflow-samples/shakespeare/kinglear.txt \
  --output gs://$BUCKET/results/outputs --runner DataflowRunner \
  --project $PROJECT --temp_location gs://$BUCKET/tmp/  --region $REGION \
  --worker_disk_type compute.googleapis.com/projects/$PROJECT/zones/$ZONE/diskTypes/pd-ssd
```

the number of disks allocated equals the maximum number of workers in streaming pipelines. When shuffle is done on the VM, the default PD size is 400 GB. Doing 100 + (0.4 TB * 100 workers) gives you 140 TB.
>

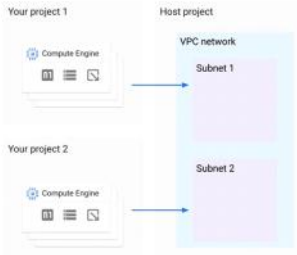
SECURITY:

- Data locality: --region=$REGION, --workerZone=$ZONE,--worker_region=$REGION when submitting jobs
  Specify regional endpoint for security compliance
  (mainly for government that does not want the data to go outside
  The country) and network latency
- Shared VPC'S
- Private ips's
- CMEK

Shared VPC

Hosts and services

- Dataflow jobs can run in either VPC or Shared VPC
- Works for both default and custom networks
- Number of VMs is constrained by subnet IP block size
- Dataflow service account needs Compute Network User role in host project



Private ip's:

- Disabling the external ip's and this secures our project from Accessing the outside internet instead internal ip's

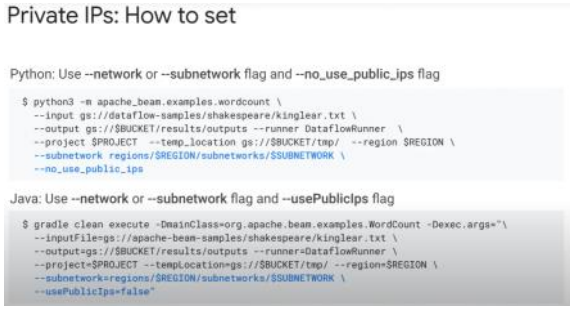
- SPECIFIED USING --network default / --network custom-network

Private IPs: How to set

Python: Use --network or --subnetwork flag and --no_use_public_ips flag

Use --network or --subnetwork flag

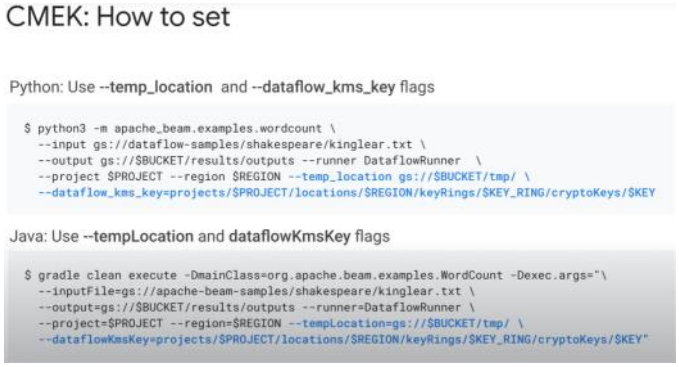
```
$ python3 -m apache_beam.examples.wordcount \
  --input gs://dataflow-samples/shakespeare/kinglear.txt \
  --output gs://$BUCKET/results/outputs --runner DataflowRunner  \
  --project $PROJECT  --temp_location gs://$BUCKET/tmp/  --region $REGION \
```




```
$ python3 -m apache_beam.examples.wordcount \
  --input gs://dataflow-samples/shakespeare/kinglear.txt \
  --output gs://$BUCKET/results/outputs --runner DataflowRunner  \
  --project $PROJECT  --temp_location gs://$BUCKET/tmp/  --region $REGION \
  --network default
```

```
$ gradle clean execute -DmainClass=org.apache.beam.examples.WordCount -Dexec.args="\
  --inputFile=gs://apache-beam-samples/shakespeare/kinglear.txt \
  --output=gs://$BUCKET/results/outputs --runner=DataflowRunner \
  --project=$PROJECT --tempLocation=gs://$BUCKET/tmp/ --region=$REGION \
  --subnetwork=https://www.googleapis.com/compute/v1/projects/$HOST_PROJECT_ID/regions/$REG
  ION/subnetworks/$SUBNETWORK
```

CMEK(CUSTOMER MANAGED ENCRYPTION KEY):



Use --region to specify a supported regional endpoint, and use --worker_region to specify the region where worker processing must occur.

THE REGION OF THE DATAFLOW JOB AND THE KMS KEYRINGS SHOULD BE THE SAME NO GLOBAL OR MULTI REGIONAL IS SUPPORTED AND GCS

LABS STUFFS:

==========

GETTING THE VALUE USION GCLOUD

```
PROJECT=`gcloud config list --format 'value(core.project)'`
USER_EMAIL=`gcloud config list account --format "value(core.account)"`
REGION=us-central1
```

GETTING THE PERMISSOINS OF OUR ACCOUNT

```
gcloud projects get-iam-policy $PROJECT  \
--format='table(bindings.role)' \
--flatten="bindings[].members" \
--filter="bindings.members:$USER_EMAIL"
```

ADDING ROLE TO US

```
gcloud projects add-iam-policy-binding $PROJECT --
member=user:$USER_EMAIL --
role=roles/dataflow.admin
```

RUN USING PRIVATE IPS

you first try to launch a Dataflow job with the **--disable-public-ips** directive. It will fail in the first attempt because the network does not have Private Google Access (PGA) turned on. You configure PGA and re-run the command to launch the job

- To enable them
  gcloud projects add-iam-policy-binding $PROJECT --member=user:$USER_EMAIL --role=roles/compute.networkAdmin
  gcloud compute networks subnets update default \
  --region=$REGION \
  --enable-private-ip-google-access

```
gcloud dataflow jobs run job2 \
--gcs-location gs://dataflow-templates-us-central1/latest/Word_Count \
--region $REGION \
--staging-location gs://$PROJECT/tmp \
--parameters inputFile=gs://dataflow-samples/shakespeare/kinglear.txt,output=gs://$PROJECT/results/outputs --disable-public-ips
```

Build and Optimize Data Warehouses in bq

Monday, August 8, 2022 11:39 AM

bq show bigquery-public-data:samples.shakespeare -- gives the details or metadata of the table
bq help query -- gives information about query command
bq query --use_legacy_sql=false 'SQL' -- queries the table -- --use_legacy_sql=false makes standard SQL the default query syntax.
bq ls 'projectid/dataset' -- list the datasets/tables
bq mk babynames -- creates dataset
bq load babynames.names2010 yob2010.txt name:string,gender:string,count:integer -- loads the txt file to table
bq rm -r babynames -- delete the d

```
SELECT * FROM `ecommerce.sales_by_sku_2017*`
WHERE _TABLE_SUFFIX = '0802'
```

- Selects the table having prefix 0802

```
DATE_DIFF(CURRENT_DATE(), date, DAY) partition_date difference from todays
```

```
SELECT
  v2ProductName,
  COUNT(DISTINCT productSKU) AS SKU_count,
  STRING_AGG(DISTINCT productSKU LIMIT 5) AS SKU
FROM `data-to-insights.ecommerce.all_sessions_raw`
  WHERE productSKU IS NOT NULL
  GROUP BY v2ProductName
  HAVING SKU_count > 1
  ORDER BY SKU_count DESC
```

- Here we concatenate all the productSKU Grouped by productname with a , seperator

!USE DISTINCT ID'S BEFORE JOINING

CROSS JOIN MULTIPLES THE DATSET INTO MULTIPLES OF THE RIGHT DATASET IF MORE RECORD ARE THERE TO JOIN

```
ARRAY_AGG(DISTINCT v2ProductName) AS push_all_names_into_array
without distinct it will produce duplicate values if present
Instead of having all the data in each row making it as a single row array
ARRAY_LENGTH(ARRAY_AGG(pageTitle)) AS num_pages_viewed # length of the array
ARRAY_AGG(FIELD ORDER BY <field>)
ARRAY_AGG(FIELD LIMIT 5)

TO QUERY ARRAY FIELDS USE UNNEST()
```

Instead of splitting the data into two tables for normalization we can convert array for the example persons having fruits type where each record have person name repeated

```
SELECT
  visitId,
  totals.*,
  device.*
FROM `bigquery-public-data.google_analytics_sample.ga_sessions_20170801`
WHERE visitId = 1501570398
LIMIT 10
```

- HERE WE USE * FOR RETURNING ALL THE STRUCT FIELDS
  THIS IS LIKE JOINING EXTERNAL TABLE .struct(RECORD repeated)
- #standardSQL
  SELECT STRUCT("Rudisha" as name, 23.4 as split) as runner
  THIS IS FOR REATING STRUCTS runner.name,runner.split
- We can also join on the structs like this

```
SELECT race, participants.name
FROM racing.race_results
CROSS JOIN
race_results.participants
```

-

Updating schema :
bq update mydataset.mytable /tmp/myschema.json

DATACATALOG:
==========
DATAcATALOG>TAG Templates>Entry Groups>sql server entry groups

Cloud sql

```
gsutil cp gs://spls/gsp814/cloudsql-sqlserver-tooling.zip .
unzip cloudsql-sqlserver-tooling.zip
cd cloudsql-sqlserver-tooling
bash init-db.sh

gcloud iam service-accounts create sqlserver2dc-credentials \
--display-name "Service Account for SQLServer to Data Catalog connector" \
--project $PROJECT_ID
gcloud iam service-accounts keys create "sqlserver2dc-credentials.json" \
--iam-account "sqlserver2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com"
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member "serviceAccount:sqlserver2dc-credentials@$PROJECT_ID.iam.gserviceaccount.com" \
--quiet \
--project $PROJECT_ID \
--role "roles/datacatalog.admin"
cd infrastructure/terraform/
public_ip_address=$(terraform output -raw public_ip_address)
username=$(terraform output -raw username)
password=$(terraform output -raw password)
database=$(terraform output -raw db_name)
cd ~/cloudsql-sqlserver-tooling
docker run --rm --tty -v \
"$PWD":/data mesmacosta/sqlserver2datacatalog:stable \
--datacatalog-project-id=$PROJECT_ID \
--datacatalog-location-id=us-central1 \
--sqlserver-host=$public_ip_address \
--sqlserver-user=$username \
--sqlserver-pass=$password \
--sqlserver-database=$database
./cleanup-db.sh
docker run --rm --tty -v \
"$PWD":/data mesmacosta/sqlserver-datacatalog-cleaner:stable \
--datacatalog-project-ids=$PROJECT_ID \
--rdbms-type=sqlserver \
--table-container-type=schema
./delete-db.sh
```

```
gsutil cp gs://spls/gsp814/cloudsql-postgresql-tooling.zip .
unzip cloudsql-postgresql-tooling.zip

cd ~/cloudsql-postgresql-tooling
docker run --rm --tty -v \
"$PWD":/data
mesmacosta/postgresql2datacatalog:stable \
--datacatalog-project-id=$PROJECT_ID \
--datacatalog-location-id=us-central1 \
--postgresql-host=$public_ip_address \
--postgresql-user=$username \
--postgresql-pass=$password \
--postgresql-database=$database
```

```
gsutil cp gs://spls/gsp814/cloudsql-mysql-tooling.zip .
unzip cloudsql-mysql-tooling.zip

cd ~/cloudsql-mysql-tooling

docker run --rm --tty -v \
"$PWD":/data mesmacosta/mysql2datacatalog:stable \
--datacatalog-project-id=$PROJECT_ID \
--datacatalog-location-id=us-central1 \
--mysql-host=$public_ip_address \
--mysql-user=$username \
--mysql-pass=$password \
--mysql-database=$database
```

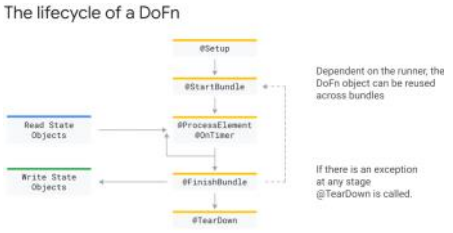
DATAFLOW DEVELOP PIPELINES

Tuesday, August 9, 2022 6:41 PM

	Input	Output	Side inputs and side outputs
ParDo	1	0, 1 or many	✓
Filter	1	0 or 1	✗
MapElements	1	1	✗
FlatMapElements	1	0, 1 or Many	✗
WithKeys	value	(f(value), value)	✗
Keys	(key, value)	key	✗
Values	(key, value)	value	✗

The dataflow pipeline receives bundles of work and each bundle is passed to DoFn() like 1 process per one worker

```
class MyDoFn(beam.DoFn):
    def setup(self):
        pass
    def start_bundle(self):
        pass
    def process(self, element):
        pass
    def finish_bundle(self):
        pass
    def teardown(self):
        pass
```



CREATING VENV

```
sudo apt-get install -y python3-venv
python3 -m venv df-env
source df-env/bin/activate
```

bq show --schema --format=prettyjson logs.logs - gives us the schema in json format

What is CoGroupByKey used for?
To join data in different PCollections that share a common key.

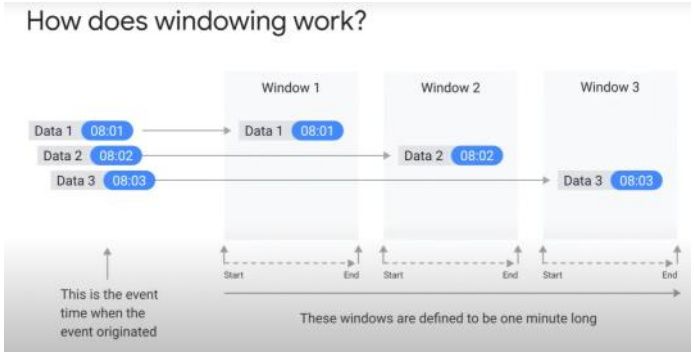
JAVA:

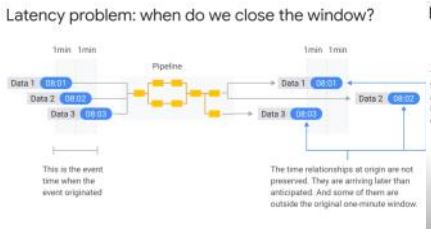
- BUILD

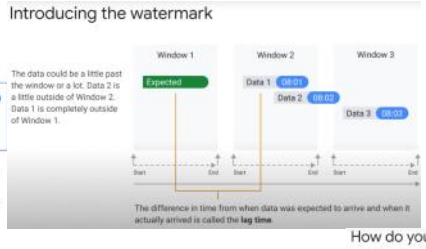
```
exportMAIN_CLASS_NAME=com.mypackage.pipeline.MyPipeline
mvn compile exec:java \
-Dexec.mainClass=${MAIN_CLASS_NAME}
```

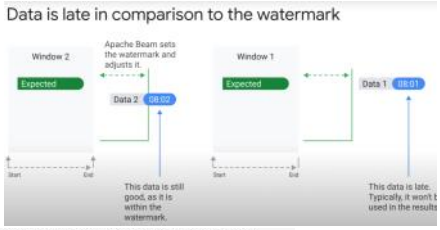
- Execute

```
mvn compile exec:java \
-Dexec.mainClass=${MAIN_CLASS_NAME}\
-Dexec.cleanupDaemonThreads=false\
-Dexec.args=" \
--project=${PROJECT_ID}\
--region=${REGION}\
--stagingLocation=${PIPELINE_FOLDER}/staging \
--tempLocation=${PIPELINE_FOLDER}/temp \
--runner=${RUNNER}"
```

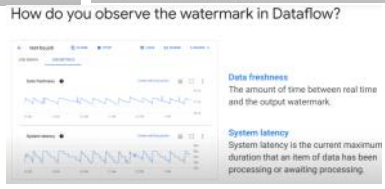


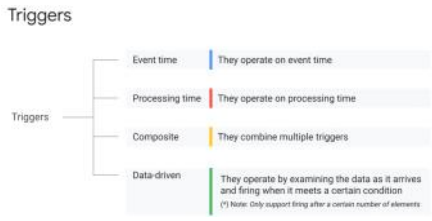






The dataflow/apache beam adjusts the window according to the watermark calculated(event time - processing time) if even though data is not in the in the window boundary then triggers are used else dropped.





CODE EXAMPLES: 1. BY USER ID(I.E FROM THE DATA)

```
beam.GroupBy('user_id')
        .aggregate_field("item_id", CountCombineFn(), "num_purchases") # groups item_id and counts it and renames the column as num_purchases
        .aggregate_field("cost_cents", sum, "total_spend_cents")
        .aggregate_field("cost_cents", max, "largest_purchases")
        .with_output_types(PerUserAggregation) # this is a class that has the schema =>
```

```
class PerUserAggregation(typing.NamedTuple):
    user_id : str
    num_purchases : int
    total_spend_cents : int
    largest_purchases : int
beam.coders.registry.register_coder(PerUserAggregation, beam.coders.RowCoder)
```

2.**by TIME:**

- All transforms | beam.Map(add_timestamp) # this will add event timestamp |
  | "WindowByMinute" >>
  beam.WindowInto(beam.window.FixedWindows(60))
  | "CountPerMinute" >>
  beam.CombineGlobally(CountCombineFn()).without_defaults( ) # this will return only the int value because we did not specify any schema| | "AddWindowTimestamp" >>
  beam.ParDo(GetTimestampFn())

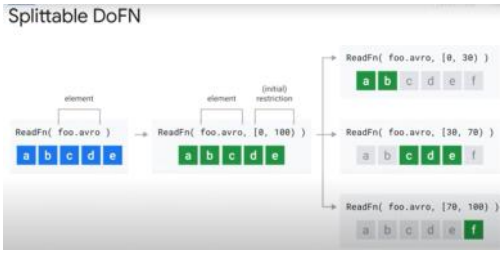
```
class GetTimestampFn(beam.DoFn):
    def process(self, element, window=beam.DoFn.WindowParam):
        window_start = window.start.to_utc_datetime().strftime("%Y-%m-%dT%H:%M:%S")
        output = {'page_views': element, 'timestamp': window_start}
        yield output

def add_timestamp(element):
    ts = datetime.strptime(element.timestamp[:-8], "%Y-%m-%dT%H:%M:%S").timestamp()
    return beam.window.TimestampedValue(element, ts)
```

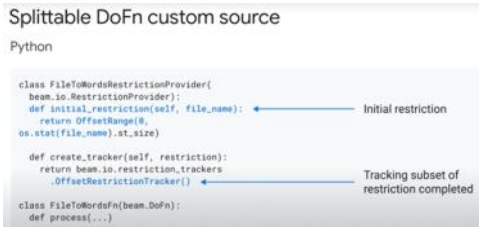
you can recover the order of the messages with a window using event time.
A message is late If its timestamp is before the watermark.
we can set as many triggers as we want

Data sinks are the ptransform that write data into destination
UPDATED LIST OF APACHE BEAM IO CONNNECTORS CAN BE SEEN IN THIS WEBSITE
s.apache.org/beam-io / https://beam.apache.org/documentation/io/built-in/

IO:
==

- Text IO
  ReadFromText(filename),p | 'create' >> Create([]) | ReadAllFromText()
  WriteToText(path,coder=JsonCoder())
-
```
(my_pcollection
| beam.io.fileio.WriteToFiles(
      path='/my/file/path',
      destination=lambda record: 'avro'
        if record['type'] == 'A' else 'csv',
      sink=lambda dest: AvroSink()
        if dest == 'avro' else CsvSink(),
        file_naming=beam.io.fileio
        .destination_prefix_naming()))
```
- File IO
  readfiels = fileio.MatchFiles('gs://tt*.txt') | fileio.ReadMatches() | beam.ReShuffle()
  P | readfiels | beam.Map(lambda x:x.metadata.path)
- BigqueryIO
  Beam.io.ReadFromBigquery(query='',use_standard_sql-true)
  Beam.ioWriteToBigquery(table,schema)
- Kafka IO (Java based but can be used in python)
  ReadFromKafka(consumer_Config={'bootstrap.servers':bootstrap.servers},topic=topic)
- BigTable IO
  We can get from bigtable by filtering rows or filtering based on the key range,wait till wrtie operation and finishes
- AVRO IO
  Beam.io.ReadFromAvro('pathtoavrofiles*')
  you can build your own custom sources and sinks using the various components such as a Splittable DoFn or customer Ptransforms https://beam.apache.org/documentation/io/developing-io-overview/





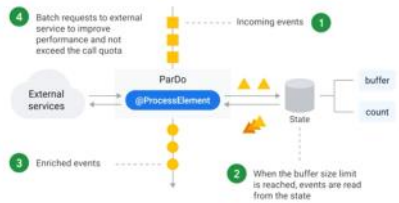
SCHEMAS:
========
For beam to convert elements into objects:
1) Bytes,
2) Proto Object(POJO in java)
3) Custom serialization

For beam to infer schemas use avros from source

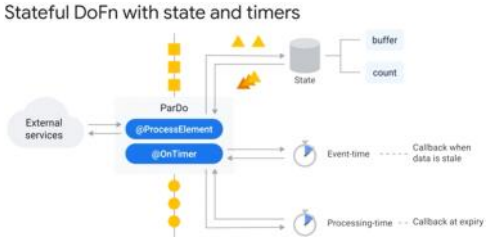
```
Transactions
bank:                STRING
transactionId :      STRING
purchaseAmountCnts : LONG
```

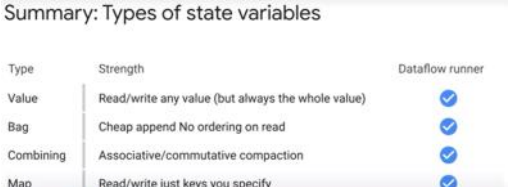
Avro objects,single list of json objects,byte string objects,
Protobuf objects Can be encoded as a schema in a pcollection

STATE API:
==========



TIMER API:
==========





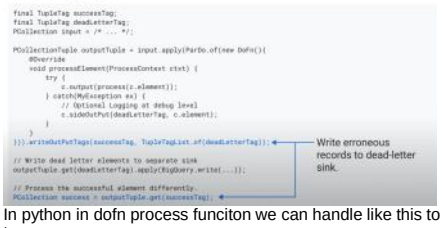
Value	Read/write any value (but always the whole value)
Bag	Cheap append No ordering on read
Combining	Associative/commutative compaction
Map	Read/write just keys you specify
Set	Membership checking

What can you do with state and timers?

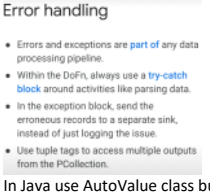
- Domain-specific triggering ("output when five people who live in Seattle have checked in")
- Slowly changing dimensions ("update FX rates for currency ABC")
- Stream joins ("join-matrix" / "join-biclique")
- Fine-grained aggregation ("add odd elements to accumulator A and event elements to accumulator B")
- Per-key workflows (like user sign up flow w/ reminders & expiration)

BEST PRACTICES:
==========

- Schemas make our way to process data for array json etc all
  In NamedTuple specify the tyoes and in table_shema speicfy the target types
- Handling unprocessable data



- In python in dofn process funciton we can handle like this to procss
  try:
  row = json.loads(element.decode('utf-8'))
  yield beam.pvalue.TaggedOutput('parsed_row', CommonLog(**row))
  except:
  yield beam.pvalue.TaggedOutput('unparsed_row', element.decode('utf-8'))
- In the calling pipeline we can do this
  Rows=| 'ParseJson' >> beam.ParDo(ConvertToCommonLogFn()).with_outputs('parsed_row', 'unparsed_row')
  .with_output_types(CommonLog)
  Rows.unparsed_rpw | write togcs
- But need to process them through window to fire them using trigger
- Handling the wrong records into sink and successful records into a separate sink
- Error handling



- In Java use AutoValue class builder to generate POJO's or if you want to be in concert with apache beam Add @DefaultSchema(AutoValueSchema.class)
- Use dofn lifecycle for various purposes(mainly with external systems)
- --allow_unsafe_triggers used for some cases

For registering POJO's we need to annotate the pojo class with
@DefaultSchema(AutoValueSchema.class)
Class _{
Getter setters
}
And in the pipeline using Convert class parse into this schema
To convert json into beam rows
Json.apply('fff',JsonToRow.withSchema(expectedschema)
If using convert
Json.apply('ddd',Convert.To(Schemaclass.class)

git clone https://github.com/GoogleCloudPlatform/training-data-analyst/

If your pipelines interact with external systems,
It is important to provision those external systems appropriately (i.e., to handle peak volumes).
Not provisioning external systems appropriately may impact the performance of your pipeline due to back pressure.>

It is recommended to utilize startBundle and finishBundle functions of the DoFn object for micro-batching. Based on runner implementation a DoFn object may be used to process more than one bundle, so it is important to appropriately reset the state of the variables used within DoFn.

APACHE BEAM HAS INTERACTIVE RUNNER FOR NOTEBOOKS SEEING INTERMEDIATE RESULTS

Set interactivity options before we run the cell

Ib=>interactivebeam

```
# Materializes the resulting PCollection in a table
ib.show(windowed_word_counts, include_window_info=True)

# Load the output in a Pandas DataFrame
ib.collect(windowed_word_counts, include_window_info=True)
```

Windowed_word_counts is the output of a pipeline or a first output or intermediate result
Visualize_data=true attribuete gives us the visual diagram

Going from development to production

```
# Import the production Dataflow runner
from apache_beam.runners import DataflowRunner

# Set up Apache Beam pipeline options
options = pipeline_options.PipelineOptions()

# Run the pipeline
runner = DataflowRunner()
runner.run_pipeline(p, options=options)
```

Beam SQL dialects

Apache Calcite
- Provides compatibility with other OSS SQL dialects (e.g. Flink SQL)
  - Copy-paste queries may require changes to table names, array indexing
- More mature implementation
  - Supports Java UDFs

ZetaSQL
- Provides BigQuery compatibility
  - Copy-paste queries may require changes to table names

Dataflow SQL

- It's a Beam ZetaSQL SqlTransform in a Dataflow Flex Template!
- Write Dataflow SQL queries in the BigQuery UI or gcloud CLI.
- Uses ZetaSQL, the same dialect as BigQuery Standard SQL.
- Optional engine for long running batch jobs.

Streaming pipelines made easy

- Target of Dataflow SQL!
- An example use case would:
  - Select from PubSub
  - Join with batch data
  - Aggregate over Window
  - Publish to BigQuery or PubSub
- Open framework with more connectors coming like Kafka and Bigtable.

```
String sql1 = "select MY_FUNC(c1), c2 from PCOLLECTION";
PCollection<Row> outputTableA = inputTableA.apply(
   SqlTransform
       .query(sql1)
       .addUdf("MY_FUNC", MY_FUNC.class, "FUNC");
```

```
$ gcloud dataflow sql query 'SELECT SUM(foo) AS baz, end_of_window
        FROM my_topic WHERE something_is_true(bizzle)
        GROUP BY TUMBLING(timestamp, 1 HOUR)'
```

The same SqlTransform in python can be written as SqlTransform(query,, dialect='zetasql') in the pipeline

```
query = """
      SELECT user_id,
COUNT(*) AS page_views, SUM(num_bytes) as total_bytes,
MAX(num_bytes) AS max_bytes, MIN(num_bytes) as min_bytes
FROM PCOLLECTION
GROUP BY user_id
      """
```

SQL FOR WINDOWS:
==========

- Use tumbling for fixed windows

```
SELECT productId,
TUMBLE_START("INTERVAL 10 SECOND") as period_start,COUNT(transactionId)
AS num_purchases
FROM pubsub.topic.`instant-insights`.`retaildemo-online-purchases-json`
AS pr
GROUP BY productId,
        TUMBLE(pr.event_timestamp, "INTERVAL 10 SECOND")
```

- Use hopping for sliding windows

```
SELECT productId,
      HOP_START("INTERVAL 10 SECOND",
                "INTERVAL 30 SECOND") as period_start,
      HOP_END("INTERVAL 10 SECOND",
              "INTERVAL 30 SECOND") as period_end,
      COUNT(transactionId) AS num_purchases
FROM pubsub.topic.`instant-insights`.`retaildemo-online-purchases-json`
AS pr
GROUP BY productId,
```

Beam dataframes:
==========

- Can use DataFrameTransform(functoin) to pass the input as a df to the ficntion and return after processing
- To_dataframe(pcollection) to convert into a df
- To_pcollection(resultofdf) to convert to pcollection
- Pandas operations like head and tail that considers ordering is not supported since Pcollection are unordered,cant us transpose

```
        INTERVAL 30 SECOND) as period_start,
      HOP_END("INTERVAL 10 SECOND",
          "INTERVAL 30 SECOND") as period_end,
      COUNT(transactionId) AS num_purchases
FROM pubsub.topic.`instant-insights`.`retaildemo-online-purchases-json`
AS pr
GROUP BY productId,
        HOP(pr.event_timestamp,
            "INTERVAL 10 SECOND",
            "INTERVAL 30 SECOND")
```

- Session windows

```
SELECT userId,
      SESSION_START("INTERVAL 10 MINUTE") AS interval_start,
      SESSION_END("INTERVAL 10 MINUTE") AS interval_end ,
      COUNT(transactionId) AS num_transactions
FROM pubsub.topic.`instant-insights`.`retaildemo-online-purchases-json`
AS pr
GROUP BY userId,
        SESSION(pr.event_timestamp, "INTERVAL 10 MINUTE")
```

- Pandas operations like head and tail that considers ordering is not supported since Pcollection are unordered,cant us transpose

It is possible by using different filters from the same source Dataframe and aggregating individually in pandas df not in beam df

Beam SQL client,dataflow template can use calcite sql

bq head -n 10 $PROJECT_ID:logs.minute_traffic to see the top 10 values(rows)

CUSTOM PTRANSFORM:

==========

- Ptransform class

```
class ParseAndGetEventTimestamp(beam.PTransform):
    def expand(self, pcoll):
        return (
            pcoll
            | 'ParseJson' >> beam.Map(parse_json)
            | 'GetEventTimestamp' >> beam.ParDo(GetEventTimestampFn())
            )
```

- Call that like this

```
| 'ParseAndGetEventTimestamp' >> ParseAndGetEventTimestamp().with_output_types(CommonLog)
```

Pipeline_opts = parser.parse_known_args()

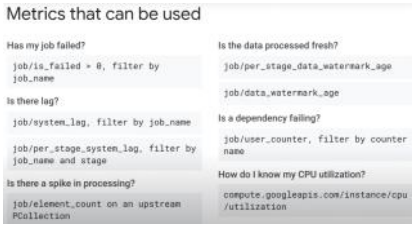
```
options = PipelineOptions(pipeline_opts, save_main_session=True, streaming=True) # says it as streaming
```

DATAFLOW OPERATIONS

Thursday, August 11, 2022 2:06 PM

In the jobs list page we can filter by their attribute like name ,status,started time etc..

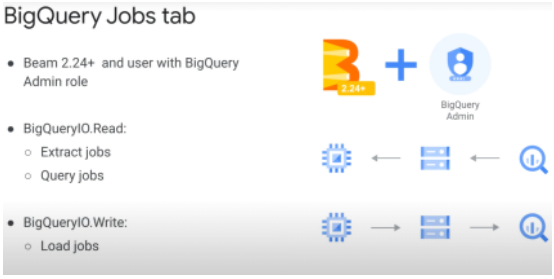
The pipeline options gave during execution of job can be found in the job info seciton of the particular jopb



We can set alerts for the metrics we can see in dataflow if the threshold is reached

The data freshness graph if the pubsub topic has messages befpre 16 hours there will be upward slope line at the start.

Same will be for latency

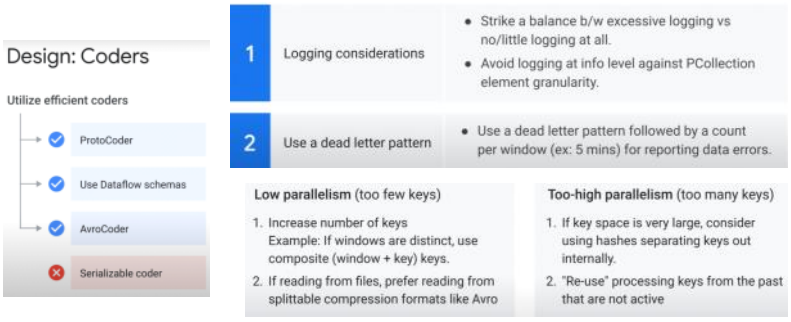


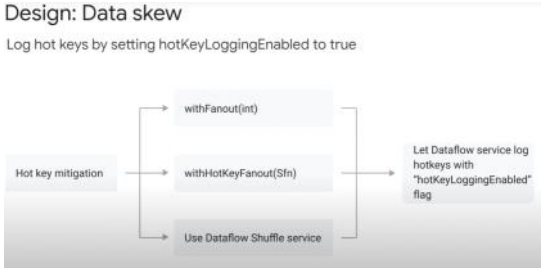
In the logs panel in dataflow UI there will be job logs, worker logs, diagnostics(for any errors),bigquery jobs (if any)

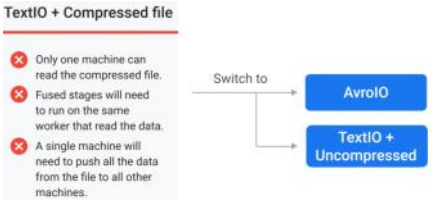
Opening the error in diagnositics will go to error reporting where we can acknowledge the error and also can link that error to an issue tracker page in a organization

We can see bigquery logs for load jobs and query jobs not for streaming inserts and extracts gcloud alpha bq jobs describe BIGQUERY_JOB_ID

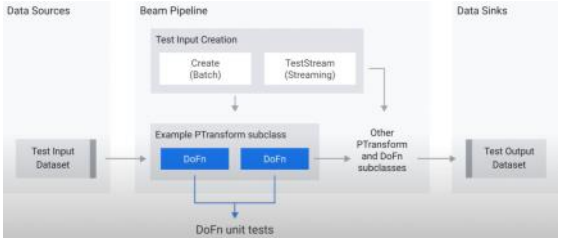
Worker log messages are limited to 15,000 messages every 30 seconds, per worker. If this limit is reached, a single worker log message is added saying that logging is throttled:No more messages are logged until the 30 second interval is over. This limit is shared by log messages generated by the Apache Beam SDK and user code.

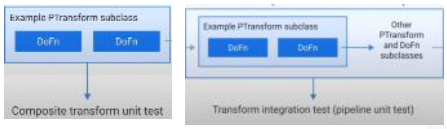


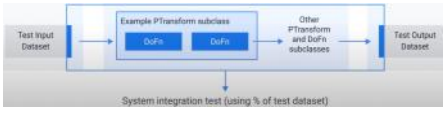


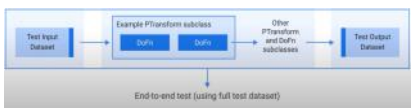


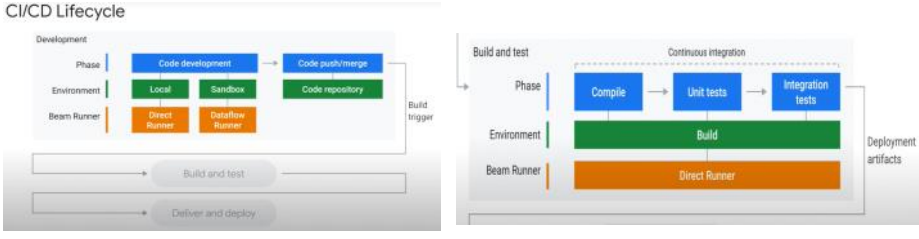
There are some known cases where fusion should be avoided. A pipeline involving massive fanout operations is one such scenario.





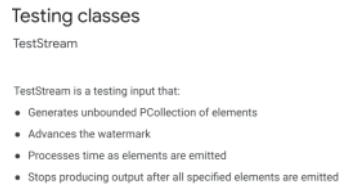


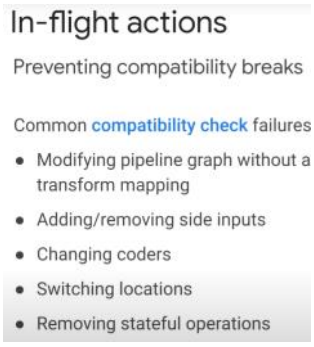




For unit testing java uses junit and in the beam TesPipeline can be used and Passert can be used for asserting errors

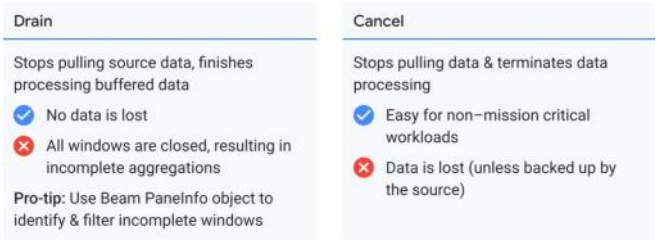
```
@Rule
public final transient TestPipeline p =
  TestPipeline.create();

@Test
@Category(NeedsRunner.class)
public void myPipelineTest() throws Exception {
  final PCollection<String> pcol = p.apply(...)
  PAssert.that(pcol).containsInAnyOrder(...);
  p.run();
}
```

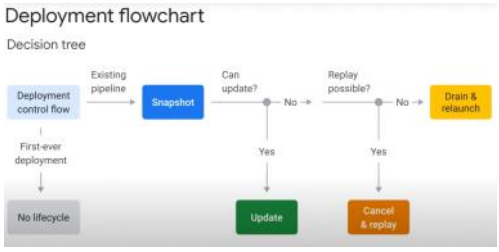




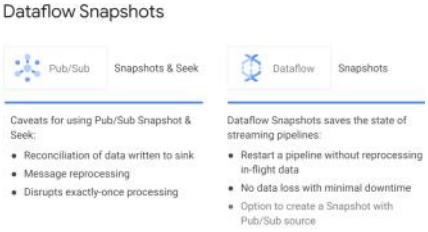
- With pAssert we can see whether the window is the Last pane or it is in a time window still
- For integration testing we can clone the production Pipelines and for the source we can create our own inmemory data or some sample data and for ouput can use Passert to test.
- Use beam 26 and nhiger
- For deployment we can either launch jobs from the Development env or launch it as a template in UI,cli,REST API call
- And after deployment use snapshots(for testing.rolling back) to preserve the state in streaming and update the job and re lauch the same job with updates
- For creating a job using snapshot need to pass extra arguments --enablestreamingengine,--createfrom snapshot=IDOFSNAPSHOT
- For creating a job update need to pass extra arguments like --update,--job_name NAME,--transform_name_mapping={'oldtransform1':'newtransform1','oldtransfrom2':'newtransform2'...}
- These are the transform names that we pass to update the names

For termination of a dataflow job we can drain(streaming) to stop receiving data and complete processing the buffered data and cancel (batch,streaming) for cancelling computing and data ingestion anonymous subclasses in your ParDos is an anti-pattern because Anonymous subclasses are harder to test than concrete subclasses.



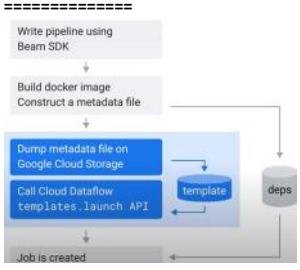


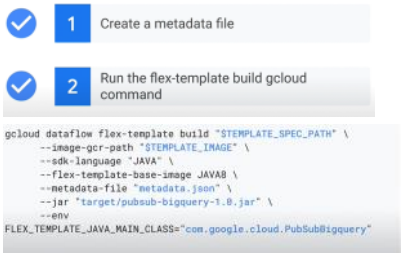
- when executing jobs don't specify region ,worker_region only specify region



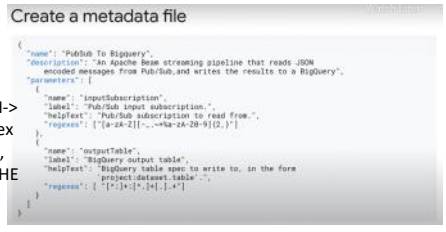
- For high availablity (if you wont consider the data loss choose to read the pubsub sub form global and process in regional and sink in multiregional
- For you who not cost is a matter then red from two different sub's and process intwo different zone or regions, write to two diferet sinks in different regions

FLEX TEMPLATES:
==========





TEMPLATE_SPEC_PATH-> path to which in GCS flex templats to be written , TEMPLATE_IMAGE IS THE BASE IMAGE THAT HAS THE PIPELINE CODE COPIED



Can launch through console,gcloud,restapi,cloud scheduler
Launching through gcloud

```
gcloud dataflow flex-template run "job-name-`date +%Y%m%d-%H%M%S`"
\
    --template-file-gcs-location "$TEMPLATE_PATH" \
    --parameters inputSubscription="$SUBSCRIPTION" \
    --parameters outputTable="$PROJECT:$DATASET.$TABLE" \
    --region "$REGION"
```

Gcloud auth print-access token can be used to generate bearer token

```
gcloud scheduler jobs create http scheduler-job --schedule="*/30 * * * *"
--uri="https://dataflow.googleapis.com/v1b3/projects/$PROJECT/locations/${REGI
ON}/flexTemplates:launch" --http-method=POST \
 --headers Content-Type=application/json \
 --oauth-service-account-email=email@project.iam.gserviceaccount.com \
 --message-body='{
    "launch_parameter": {
      "jobName": "job-name"
      "parameters": {
        "inputSubscription": "'$SUBSCRIPTION'",
        "outputTable": "'$PROJECT:$DATASET.$TABLE'"
      },
      "containerSpecGcsPath": "'$TEMPLATE_PATH'"
    }
  }'
```

gcloud config set builds/use_kaniko True to have the builder
export TEMPLATE_IMAGE="gcr.io/$PROJECT_ID/my-pipeline:latest"
gcloud builds submit --tag $TEMPLATE_IMAGE .

CI/CD:
=====
CONFIGURE A SOURCE REPO FOR ALL THE SOURCE CODE
CREATE A COMPOSER ENVIRONMENT OR DATAFLOW ENVIRONMENT FOR EXECUTION
CREATE BUCKETS HAVING THE LATEST CODE IN PLACE
AFTER THAT GO TO CLOUD BUILD SETUP A TRIGGER ON THAT SOURCE REPO
AND TELL IT TO TRIGGER THE YAML FILE BUILDING WHENEVER SOMETHING IS

AND TELL IT TO TRIGGER THE YAML FILE BUILDING WHENEVER SOMETHING IS COMMITTED AND THE YAML FILE SHOULD BE IN THE SAME REPO

DURING PRACTICAL TESTS NOTES

Thursday, August 18, 2022 8:54 PM

Cloud IAP Tunneling:
==========
When an vm instance has not external IP address we can tunnel to that vm with the help of other vm having external ip address.for that we need to have the iap ssh,tcp tunneling > add principal > IAP Secure Tunnel User to the external ip's not having vm's and we can ssh from the vm having external ip and it willl work and also using iap-tunnel command to tunnel securely

Cloud Armor we can use for denying or allowing user's or any ip address(source) to access a target like http load balancer(target)

When we click allow http connection checkbox in vm instance creation it will create a tag with http-server and tcp:80 for us

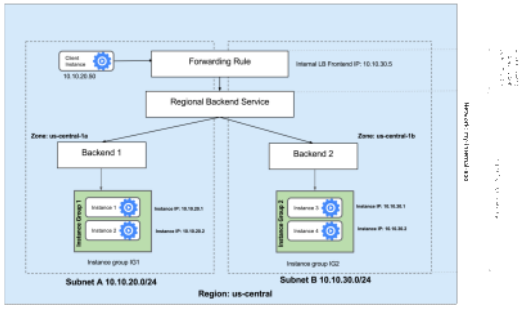
Internal Load balancer is a regional service

RDP=TCP:3389
SSH=TCP:22
HTTP:TCP:80

RPS(request per second configuration in load balancing allows the load balancer to keep the instance for 50 RPS

For Internal load balancing, the health check probes to your load balanced instances come from addresses in the ranges 130.211.0.0/22 and 35.191.0.0/16
For HTTP load balancing, the health check probes to your load balanced instances come from addresses in the ranges 130.211.0.0/22 and 35.191.0.0/16

Internal Load balancing



Internal load abalancer load balances the traffic between tow MIG's congured by us that is internally load balancing between vm's
Eg: when I access the load balancer ip(from the vm or using the vm) then it will forward the tcp connect to one of the MIG's VM instead of forarding http traffic to MIG's in case of https load baalncing

To learn hadoop:
- **Mapreduce**
- **Hadoop hdfs**
- **Pig,spark**
- **Kafka**

To learn dataproc how can replace
To learn storage migration and using it as usual
Ai,ml concepts kuberflow,ai platform

Cloud sql only is having development ,production option while creating instance
Promotion means that the destination Cloud SQL instance is disconnected from the source, and is promoted from a replica instance to a primary instance.(when running cdc the sql instance is sa read replica only but after promoiting it will be available for standalone read/write)
Can connect thorugh vpc peering ,allowist,tunneling etc)

Dnsutils to install dig cli to get the ip for a website/instance,hostname

TO INSTALL AWS CLI IN CLOUD SELL
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

AWS IS HAVING BLOCKCHAIN FRAMEWORK SERVICE USING HYPERLEDGER FABRIC OPEN SOURCE
ALSO SATELLITE DATA COMMUNICATION SERVICE (
)

AWS BRAKET(QUANTUM COMPUTING RESEARCH):
https://github.com/aws/amazon-braket-sdk-python
https://docs.aws.amazon.com/braket/latest/developerguide/what-is-braket.html
https://github.com/aws/amazon-braket-examples

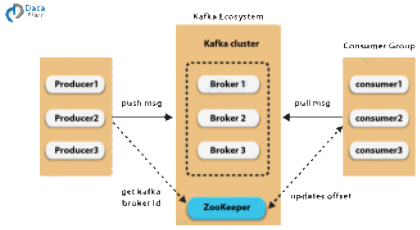
For Database migartion:
1) Create a conneciton profile in database migration service with hotsname password etcc..
2) Create a migration job with source as eg: RDS AWS, target as cloud sql(createing new destination)
3) Migartion type: CDC(continous),one time
4) Allowlist the cloud sql to the source group(secirity group)
5) Database migration service creates an external databas

After migrating the objects will be in definer class in sql we may need to change it(in src) to invoker class to avoid failures while querying objects.

SELECT DISTANCE, DEP_DELAY
FROM dsongcp.flights_tzcorr
WHERE RAND() < 0.001(example of sampling)

KAFKA:
=====
1) Zookeeper
2) Kafka Broker
3) Input topic for producing messages
4) Oupput topic for consuming messages

-- these all should be running in order to consujme messages concurrently/continiuldy wihtout disruption



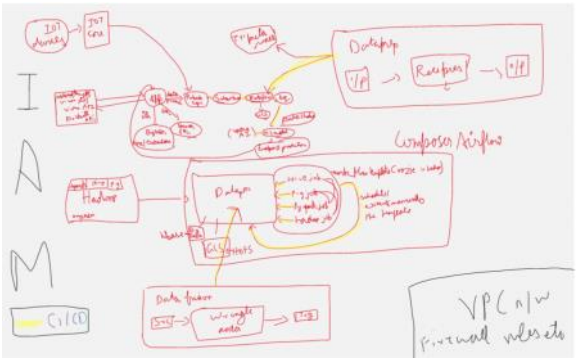
DATASTORE USED FOR KIND(TABLE),ENTITY(ROW),PROPERTIES(DTYPES)
Spanner creates the instance fast
Spanner gives a query tool to query the database, has replica expot,import options,backup etc..

Completes the Dockerfile by entering the statement, gunicorn ..., that executes when the container runs. Gunicorn (Green Unicorn) is an HTTP server that supports the Python Web Server Gateway Interface (WSGI) specification.
gunicorn>=19.7.1,grpcio>=1.33.1,grpcio-gcp>=0.2.2,grpcio-tools>=1.33.1,

WORKFLOW TEMPLATES CLUSTER SELECTOR:
==========
Goog-dataproc-cluster-name:cluster-name
While creating clusters if any labels specified like --labels key:value that we can specify here

```
from airflow.utils.task_group import TaskGroup

group1=TaskGroup("group1")
group1.add(d1.send_email_notification)

Or

With TaskGroup("group1") as group1:
     Task1=sfss
     Task=mjgdd
     Task1>>task2
```

ln -s ~/training-data-analyst/courses/developingapps/v1.2/python/kubernetesengine ~/kubernetesengine creates a working directory from the target directory which we cd 'd into



```
start = BashOperator(task_id='start', bash_command='exit 0')
paths = TaskGroup(group_id='paths')
path_a = TaskGroup(group_id='path_a', parent_group=paths)
task_process_a  = BashOperator(task_id='task_process_a', task_group=path_a, bash_command='exit 0')
task_store_a    = BashOperator(task_id='task_store_a', task_group=path_a, bash_command='exit 0')
task_process_a >> task_store_a
path_b = TaskGroup(group_id='path_b', parent_group=paths)
task_process_b  = BashOperator(task_id='task_process_b', task_group=path_b, bash_command='exit 0')
task_store_b    = BashOperator(task_id='task_store_b', task_group=path_b, bash_command='exit 0')
task_process_b >> task_store_b
path_a >> path_b
end = BashOperator(task_id='end', bash_command='exit 0')
start >> paths >> end
```

```
from airflow.decorators import task_group
@task_group(group_id=f'path_a')
def path():
task_process  = BashOperator(task_id=f'task_process_a',
bash_command='exit 0')
task_store  = BashOperator(task_id=f'task_store_a',
bash_command='exit 0')
task_process >> task_store
path()
```

cloudshell download $HOME/openapi2-functions.yaml to download the file specifried

The e2-standard-4 allows up to 4 network interfaces

_TABLE_SUFFIX in standard SQL for wildcard table queries
TABLE_DATE_RANGE([DATASET.TABLE*],TIMESTAMP(),TIMESTAMP()) in legacy SQL
TABLE_QUERY([DATASEY],'regexp(table_id,[0-5])')

Streaming data cant be avaolable for copy till 90 minutes

For **SAMPLING:**
==========
Use * FROM <datset.table> TABLESAMPLE SYSTEM(20 PERCENT)-SELECTS 20% OF THE DATABLOcKS FROM STORAGE

Use WHERE rand() < 0.1- randomly selects 10% rows

Sampling in views,subquerirs,in IN clause,row-levelrestricted table not supported

GENERATE_ARRAY(11, 33, 2) used to generate arrays
The maximum size of a variable in a session is 1 MB, and the maximum size of all variables in a session is 10 MB.
CREATE TEMP TABLE Flights(total INT64) AS SELECT * FROM UNNEST([10,23,3,14,55]) AS a;
SELECT * FROM Flights;
Sessions are auito trminated after 24 hours of inactivity or 7 days
Can termiinate manunally by closing the editor tab
CALL BQ.ABORT_SESSION(SESSION_ID); session_id==@@session_id(also found in the query results>job indormation)
Also view query history in query history button in the editor tab

Each base table can be referenced by up to 20 materialized views from the same dataset, up to 100 materialized views from the same project, and up to 500 materialized views from the whole organization.
CREATE MATERIALIZED VIEW project-id.my_dataset.my_mv_table
PARTITION BY RANGE_BUCKET(column_name, buckets)
OPTIONS (enable_refresh = false, refresh_interval_minutes = 60) AS
SELECT date, AVG(net_paid) AS avg_paid
FROM project-id.my_dataset.my_base_table
GROUP BY date
Scheduled queries are a convenient way to run arbitrarily complex calculations periodically. Each time the query runs, it is being run fully. The previous results are not used, and you pay the full price for the query. Scheduled queries are great when you don't need the freshest data and you have a high tolerance for data staleness.

For BI engine we can either leaver the defaults or specify our preferred tables for the analysis to happened in the background for query performance
INFORMATION_SCHEMA.BI_CAPACITIES,CHANGES contains metadata about the current state of BI Engine capacity.

Materialized views are suited for when you need to query the latest data while cutting down latency and cost by reusing the previously computed result. You can use materialized views as pseudo-indexes, accelerating queries to the base table without updating any existing workflows.
ALTER MATERIALIZED VIEW PROJECT.DATASET.MATERIALIZED_VIEW
SET OPTIONS (enable_refresh = true);
CALL BQ.REFRESH_MATERIALIZED_VIEW('PROJECT.DATASET.MATERIALIZED_VIEW');
USED WHEN WORKED WITH BEST QIERY ACCESS PATTERNS

ANALYTICS HUB:
==========
- Create a data exchange where we can add listings that can be viewed in explore analytics hub options in ADD DATA
- Can also set permissions to only view to specific users(subscribers)
- When connecting to listing it will create a linked dataset to link the dataset provided in the listing

CLONIGN TABLSES
CREATE TABLE
myproject.myDataset_backup.myTableClone
CLONE myproject.myDataset.myTable; CANT CLONE VIEWS,MATERIALIZED VIEWS,EXTENRAL TABLES

SNAPSHOT TABLE:
==========
CREATE SNAPSHOT TABLE myproject.library_backup.books
CLONE myproject.library.books
  OPTIONS (
    expiration_timestamp = TIMESTAMP '2022-04-27 12:30:00.00-08:00');
Or using UI to click button in table details
TIMETRAVEL:
FOR SYSTEM_TIME AS OF
  TIMESTAMP_SUB(CURRENT_TIMESTAMP(), INTERVAL 1 HOUR);
USE SCHEDULED QUERY TO TAKE A SNAPSHOT MONTHLY OR PERIODICALLY

BigQuery does not support foreign keys.For OLTP workloads, consider using either Cloud Spanner or Cloud SQL Dremel is a distributed system developed at Google for interactively querying large datasets. Dremel is the query engine used in Google's BigQuery service.

From an existing cluster also we can create pipeline jobs in data fusion
By creating a new profile for compute

BigQuery does not support foreign keys.For OLTP workloads, consider using either Cloud Spanner or Cloud SQL Dremel is a distributed system developed at Google for interactively querying large datasets.
Dremel is the query engine used in Google's BigQuery service.
After submitting a query it will hit the api request and after that execution tree is created to assign works to workers in parallelly to execute and the intermediate resutls are stored in shuffle tier and a query plan(INFORMATION_SCHEMA.JOBS* views or execution tab) is created in order to execute them and finally the results are sent to the user

From an existing cluster also we can create pipeline jobs in data fusion
By creating a new profile for compute
Also there are dataproc profile,aws profile,hadoop profile

Legacy SQL(dremel dialect) is used in bigquery dremel or some infrastricture it is using .
Window functions in bigquery/sql is used to aggregate data in a group withou using join
Need to speell out OVER after the column like(AVG(col1) OVER(PARTITION BY COLNAME ORDER BY COLNAME)) (DENSE_RANK()-ROW NUMBER WITHIN THE SUBSET,FIRST_VALUE,LAST_VALUE() IN THE SUBSET,LAG(),LEAD()- FOR GETTING THE PREVIOUS/NEXT NTH ROW GIVEN OFFSET)

Dremel does the query tree execution stuff and jupier manages the petabit network between borg(for computing),colossus,dremel

```
SELECT * FROM EXTERNAL_QUERY("studious-lore-344410.asia-south1.spanner-conn-
id", "SELECT * FROM tst_tb");
```
For spanner instarnce while updating(edit instance) we can increase the processing units(node units)(horizontal scaling)
We can create tables,views
We can import and export the data (sqldumps etc..)
We can see the data in the UI in spanner and query like bigquery using UI,seeing schema , adding secondary indexes
Querying database is like snapshotting one time(session)
Adding a secondary index to an existing table requires a schema update. Like other schema updates, Cloud Spanner supports adding an index while the database continues to serve traffic. Cloud Spanner populates the index with data (also known as a "backfill") under the hood

Cloud SQL: deletion protection is there to avoid deleting using button
Only by editing instance we can delete(can swithc b/w dev,prod)(vertically scalable)
Supports automatic backups,pointin time recovery,
We can specify a maintenance window for update to take place,
We can set flags,labels at the time of creation
Also we can use write for primary instance,reads from read replica for not heavy loading the primary instance(we can customize our instance after creation by editing itIn order to enable replication for a MySQL instance, automated backups and point-in-time recovery (binary logging) must be enabled first. Enabling replication will enable both of these settings, which affects cost.
export ADDRESS=$(curl -s http://ipecho.net/plain)/32 for getting the address of the currrent VM

Cloud BigTable:= App profile for routing the incoming requests
Single cluster(if request contacts cluster 1 and if that cluster is not running then the request fails)
Multi cluster routing(tries between different clusters if one fails)
We can manually update the cluster either maual allocation or autoscale(when editing instance we can add clusters) we cant change storage type for that we need to import by creating a new instacne by expoeting data from this instacne Field promotion involves moving fields from the column data into the row key to make writes non-contiguous(bigtable does not require acid prpperties)
cbt set test-sessions green1939#1638940844260 Interactions:red_hat=seen
r1 cf1:cq1=c1
If we set the value for same cell it records the timestamped wise cell values (this will be taken care in garbage collection) (modifying a row is called mutation)

Dataproc charges will be minute by minute

AI PLATFORM:
==========
in pre built training job only custom and basic scale tier apply
But in custom code training basic(1 worker),standras1(1m,4w,3paramservers),premium 1(1m,19w,11ps),basicgpu(one worker with nvidia gpu),basictpu(master cpu,cloud tpu),custom tiers are there

ML FUNDAMENTALS:
==========
REGRESSION:
Mean Absolute Error (MAE) = Average of All absolute errors
Mean Squared Error (MSE): Average of squared differences between prediction and actual observation.
While training the model we will adjust the weights(w) to reduce MAE,MSE
The error calculations plotted against w is called cost function(minimization fn)
When plotting cost function to 'w',there will be some slope that should come to minimum for that we use gradient descent.
In the gradient descent algorithm, we start with random model parameters and calculate the error for each learning iteration, keep updating the model parameters to move closer to the values that results in minimum cost.(repeated until minimum cost)
There are three ways of doing gradient descent:
Batch gradient descent: Uses all of the training instances to update the model parameters in each iteration.
Mini-batch Gradient Descent: Instead of using all examples, Mini-batch Gradient Descent divides the training set into smaller size called batch denoted by 'b'. Thus a mini-batch 'b' is used to update the model parameters in each iteration.
Stochastic Gradient Descent (SGD): updates the parameters using only a single training instance in each iteration. The training instance is usually selected randomly. Stochastic gradient descent is often preferred to optimize cost functions when there are hundreds of thousands of training instances or more, as it will converge more quickly than batch gradient descent

CLASSIFICATION:
ensure the cost function is convex (and therefore ensure convergence to the global minimum), the cost function is transformed using the logarithm of the sigmoid function
Over-fitting=increase the features one by one and test (don't add too many features)
- **Reduce the number of features**: Manually select which features to keep. Doing so, we may miss some important information, if we throw away some features.
- **Regularization**: Keep all the features, but reduce the magnitude of weights W. Regularization works well when we have a lot of slightly useful feature.
- **Early stopping**: When we are training a learning algorithm iteratively such as using gradient descent, we can measure how well each iteration of the model performs. Up to a certain number of iterations, each iteration improves the model. After that point, however, the model's ability to generalize can weaken as it begins to over-fit the training data.

Underfitting=don't decrease the features too much have sufficient features to avoid undefitting

FOR BOTH CLASSIFICATION ,REGRESSION:
==========
- L2 or Ridge regularization can be applied to both linear and logistic regression by adding a penalty term to the error function in order to discourage the coefficients or weights from reaching large values.The simplest such penalty term takes the form of a sum of squares of all of the coefficients, leading to a modified linear regression error function where lambda is our regularization parameter and apply gradient descent function L2 penalty aims to minimize the squared magnitude of the weights.
- L1(Lasso) Regularization L1 penalty aims to minimize the absolute value of the weightsL2 shrinks all the coefficient by the same proportions but eliminates none, while L1 can shrink some coefficients to zero, thus performing feature selection.

Hyperpaarmeters example are the learning rate regularization lambda function Cross-validation allows us to tune hyper-parameters with only our training set. This allows us to keep the test set as a truly unseen data-set for selecting final model . That is done by splitting training data into 4 parts and using that repeatedly checking the validation etcc..

UNSUPERVISED LEARNING:
In this in the training data the desired outputs are not given so the example would be clustering

FILESTORE:
==========
After creating a instance we get the ipaddress:/mountname
And we can install mount in this filestore in vm or somewhere
IN VM:
sudo apt-get -y update &&
sudo apt-get -y install nfs-common
sudo mkdir /mnt/test
sudo mount YOUR_INSTANCE_URL:/vol1 /mnt/test
df /mnt/test to see the disk usage

Switching between datastore and firestroe native:
$ gcloud alpha firestore databases update --type=firestore-native
$ gcloud alpha firestore databases update --type=datastore-mode
Of by console if available