Book

# Table of Contents

# Chapter 1 : Selecting appropriate storage technologies

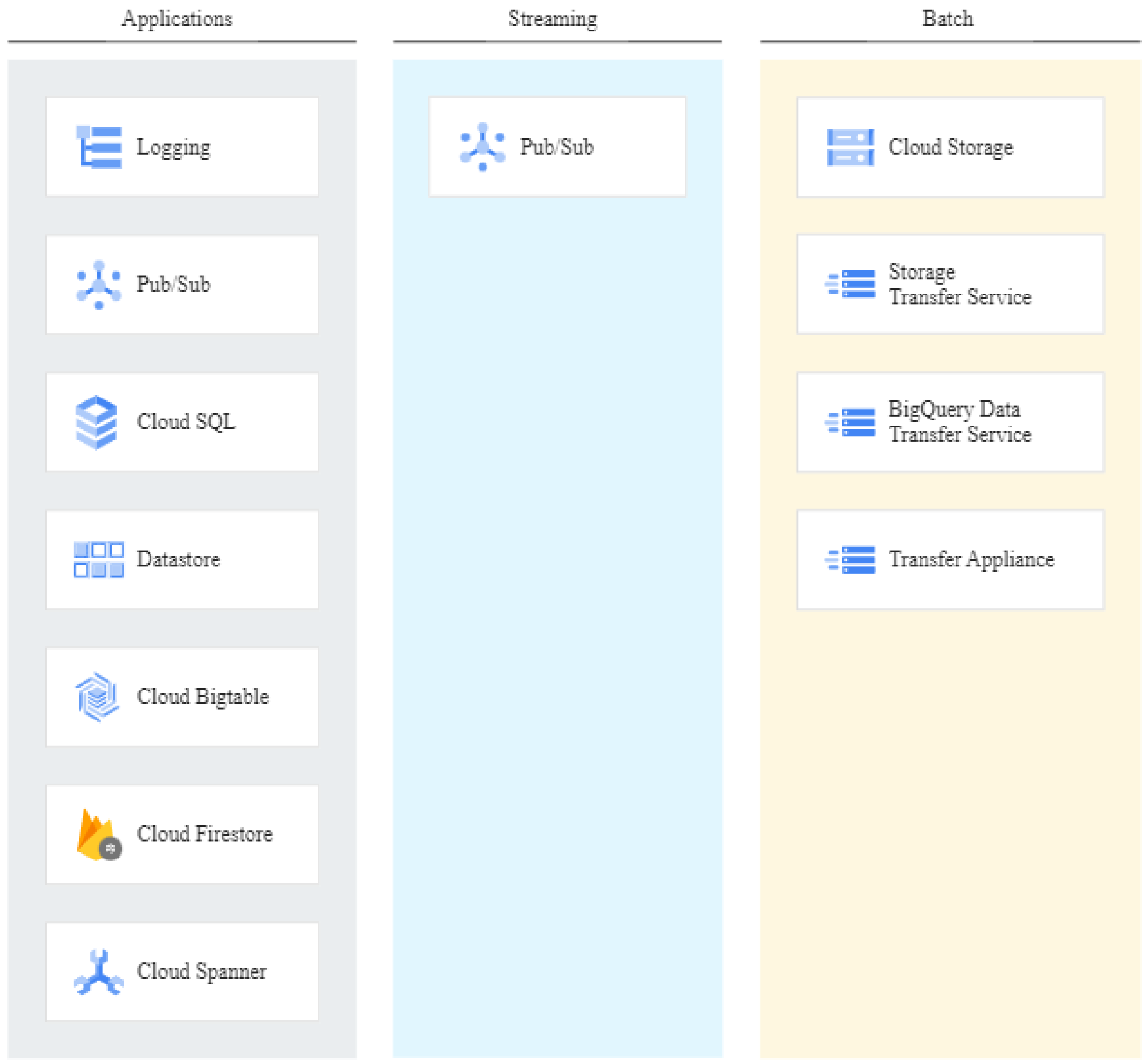
## Business aspects of choosing a storage system

The data lifecycle consists of : ingestion, store, process and analyze, explore and visualize.

1.1.1. Ingest : 3 ingestion modes

- Application data( websites, mobile apps, logs), can be ingested with any compute units such as compute engine, Kubernetes, app engine. Or can be written to stackdriver logging or a managed database : cloud sql or cloud datastore.
- Streaming data : IOT data, VM monitoring data. Pub/sub ingestion
- batch data : transaction data ( collected from applications stored in RDB and exported to ML pipelines), archiving data, migrating large volume of data. google cloud storage.

tools : App engine, compute engine, Kubernetes engine, cloud functions, cloud run, pub/sub, logging, storage transfer service, bigquery data transfer service, transfer appliance


Applications
Streaming
Batch
Logging
Pub/Sub
Cloud SQL
Datastore
Cloud Bigtable
Cloud Firestore
Cloud Spanner
Pub/Sub
Cloud Storage
Storage Transfer Service
BigQuery Data Transfer Service
Transfer Appliance

1.1.2.Store : consider access control requirements and how well storage system supports those requirements (nearline storage :accessed < once per 30 days; coldline storage accessed < once oer year)

- Cloud storage and bigQyery : long lived analytics data
- Relational database or NoSQL suited for frequently accessed data
- As data ages : possible to delete data or exported to cloud storage

Tools : Cloud spanner, cloud firestore, cloud storage for firestore, cloud storage, cloud SQL, datastore,cloud bigtable, bigquery

1.1.3.Process and transformation: data cleansing, normalization, and standardization

- Cloud dataflow is suited for transforming stream and batch data

1.1.4. Data analysis, explore and visualization

- Typical data analysis description methods,: mean, std, histograms, correlation, predictions, clusters
- Cloud dataflow, cloud dataproc, bigquery and cloud ML engine are useful for this
- Cloud datalab: based on jupyter notebooks is a widely used gcp tool for explore and viz. it uses python or SQL to explore/
- Google data studio is useful for tabular reports and charts: avoid contact with code
- Google data prep

## Technical aspects of data for selecting a storage system

1.2. Technical aspects of data for selecting a storage system : volume and velocity, variation in structure, data access patters, security requirements

1.2.1. Volume

- Cloud storage : up to 5TB, no limit on read/write
- Cloud Bigtable : 8TB per node when using HDD (hard disk driver), and up to 2.5TB per node with SSDs. Each bigtable instance can have up to 1000 tables/
- BigQuery : no limit on number of tables, up to 4000 partitions per table ( datasets).
- Persistent disks : attached to Compute engine, store up to 64TB
- Mysql , PostgreSQL, and SQL server can store up to 30TB.
- Cloud sql good choice for relational database serving requests in single region.

1.2.2.Velocity : important for IOT and ML

- Important that income data rate matched the velocity of data store writing

- Bigtable up to 10000 rows/second using 10 node cluster with ssd
- Pub/sub can be very useful when processing and ingesting at the same time
- Cloud pub/sub is scalable managed messaging service.
- With low velocity : large migrations that may wait days ➔ cloud storage

1.2.3. Structure variations

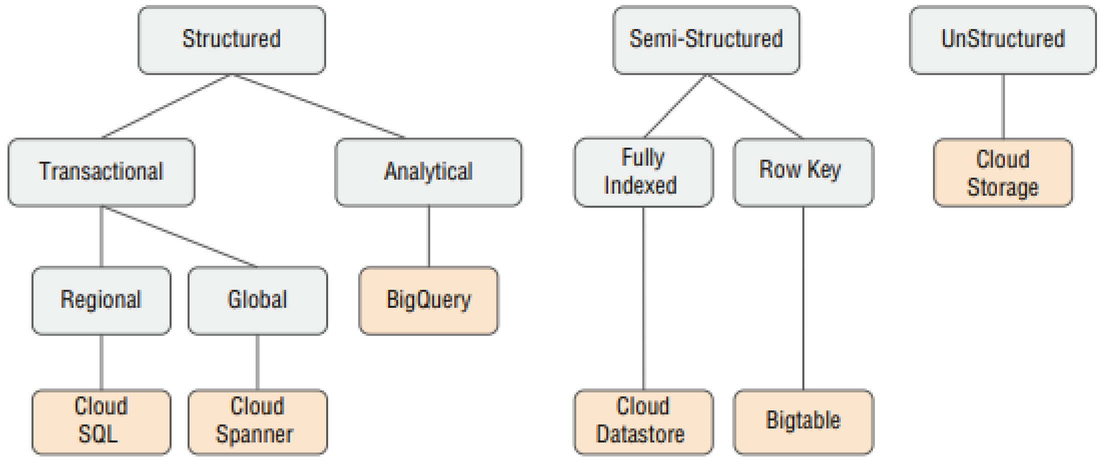
- Structured : relational data : row database (MySQL, Postgres) column database ( Bigquery)
- NoSQL : Document database[ MongoDB, CouchDB, orient DB (key value based)], Wide column databases [ Bigtable, Cassandra]
- Unstructured : images, videos ( usually stored in cloud storage)

1.2.4.Data access patterns and security: depends on how much data is retrieved/written in a read/write operation, how often is data read/written

- Telemetry: due to low latency : bigtable
- Cloud SQL deals with less amounts of data
- Cloud storage supports large volumes of data in bulk using transfer service and transfer appliance
- BigQuery: large volume but rescales the complexity using columnar storage
- Security : all data on google cloud is encrypted

## Types of data :

1. Structured data : fixed attributes that fits to rows and columns
    a. Transactional : row-oriented storage : Cloud SQL and Cloud Spanner
    b. Analytical: uses column-oriented storage : BigQuery
2. Semi structured data : with attributes but the set of attribute can differ from an instance to another.
    a. Fully indexed :documents: each instance has an id value that distinguish it from others datastore
    b. Row key access wide columns: values of rows become the access key, as it organizes data so that rows with similar row keys are close to each other. Designed to answer particular queries (time, sensnor ID )
3. Unstructured data : does not fit into a tabular structure, images, videos audios



## Relational database design

1. OLTP : Online transaction processing : follow rules of normalization, mainly3 rules

- 1FN : attributs élémentaires
- 2FN : $\forall\, a \notin identifiant : partie\ identifiant \nrightarrow a$
- 3FN : $\forall\, a, b \notin identifiant : a \nrightarrow b$

   a. First norm rule : each table has only atomic values ( like composed addresses needs to be decomposed )
   b. Second form rule : 1st form + separate tables for values that apply multiple rows and links using foreign key ( subsequencing an attribute from an element of primary key, for example id primary key is ID name and date, but attribute name can be defined only with ID name, then transform ID name and name to a different table )
   c. Third form rule: 2nd form + no relations between attributes that don't belong to primary key

A latency/ normalization compromise is usually performed.

2. OLAP : Online analytical processing( dimensional models) : used for data warehouse and data mart application ( PA cubes)

## No SQL database

d. Key value

- Uses associative arrays or dictionaries as the basic datatype. Keys are data used to look up values.
- Simple , but can get more complex if the key value is complex, for example if it's a JSON. If the JSON object is all a key, then no problem, but if the key value is a partition of JSON than it is preferable to consider document databases.
- Cloud Memorystore(does not support persistence)/Reddis (supports persistence : writing of data to durable storage, such as a solid-state disk (SSD) )

e. Document

- Can support multi cases with different document distribution connected with an element ( for example ID)
- Managed databases : use cloud datastore
- Own document database : use mongodb couchdb and orientDB

f. Wide columns

- Use cases : high volume of data, low latency writing, more write than read, no ad hoc queries, lookup by a single key
- Are often sparse( empty cells, null, NA)
- Bigtable on GCP , very useful for migration on-premises Hadoop Hbase
- It can also be managed with Cassandra which run in Compute engine and Kubernetes engine

g. Graph

- GCP has no Graph based database, but Bigtable can be used as a storage backend of HGraphDB or Janus Graph
- Data can be retrieved from a graph using SQL-like declarative statements which specifies the list of neighbors, or traversal language such as gremlin which specifies the moves ( more like a state/ transition merit)

# Chapter 2 : building and operationalizing storage systems

## Storage systems

| Service | Type | Operational tasks managed by GCP | Regions | Max storage | Instance type (max) | Connection | Availability | Improve performance | available tools | Import export | Moe infos |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cloud SQL | Managed relational database *Access (instance ID,pwd,region & zone, DB version) | Create database, performing backups, Updating OS, scale disks, configure failover, monitoring and network connection authorizing | regional | 30 TB 10000 tables | 64 vCPUs 416 GB RAM | Public/private IP Cloud SQL proxy | Possibility of high availability : creating a second instance in another zone and synchronously replicate data | *Read replicas : copy of primary instance=> read with the replica and let primary instance for writing. * same region. *Can support multiple replicas but cloud SQL does not balance between them. *it can be promoted to a standalone cloud SQL; it's irreversible primary instance cannot be restored from backup if read replicas exist *available failovers | MySQL Mysqldump | CSV SQL dump | *Exclude :Views, triggers, stored procedures, functions. * database objects have to recreated using scripts after importation *Can be compressed |
| | | | | | | Possible use of flags | | | PostgreSQL Pg_dump | | |
| | | | | | | | Possibility to shard data into smaller databases: recommended | | SQL server Bcp( bulk copy) | | |
| Could spanner | Managed relational database *Access (instance name, | *High availability,horizontally scalable * distributed nature creates challenges fro | Global | Depended on nodes: each node | Recommended to keep | | High availability with replicas and reduced latency ( data closer to applications) | *automatic replications *does not require failover instances | | Apache avro CSV Import using | *hotposts : read and write on same node. It happens when using sequential |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | ID,pwd,regional or multi regional, nbr of nodes,region) | writing, for that it uses a voting mechanism to determine writers | | stores up to 2TB<br><br>Each row limited to 4GB | below : 65% fro regional And 45% for multi regional instances | | | * multi replicas, one leader responsible for writing<br>*- read-write replicas ( full data& read op,vote writing op)<br>*- read only replicas (full data & read op)<br>*- witness replicas (vote writing op)<br><br>*regional uses 3 read-write replicas<br>* multi regional uses all types : 2 regions read-write, one of them is leader. Witness replica is in a third region<br>• regio | | dataflow connector | Primary key( timestamps, incrementing counters).<br>* solution: use hash values<br>* to avoid high latency when joining the tables, it uses interleaving data : save order row with order line item!! |
| BIG table | NoSQL, wide columns<br><br>*Access (instance name, ID,region, zone,nbre of nodes, Production or dev cluster) | Production cluster : high availability and at least 3 nodes<br><br>Dev : low cost option<br><br>*Linear scalability<br>* suitable for low latency writes | global | *Field promotion* involves moving fields from the column data into the row key to make writes non- | | | High availability with more clusters and nodes | For multi-regional, create replicated cluster in another region<br><br>App profile: specifies how to handle incoming requests, in this case automatic failover | *Hbase shell<br>* Hbase client<br>*cbt command line utility | Using cloud dataflow<br>*avro format<br>*sequenceFile<br>*CSV | Expensive |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | contiguo us. | | | | If routed to one cluster: manual failover | | | |

| | | | | | |
|---|---|---|---|---|---|
| Firestore | Managed document database (replacing datastore) | Two modes :<br>*Native mode<br>*Datastore mode<br>Real time update, web and mobile client libraries( native)<br><br>Suitable for data that does not require low latency writes | Data model : entities, entity groups, properties and keys<br>Entity group[<br>Entity{propertyA: value, propB: {prop1 :val1,prop2 : val2}}] | Indexing: two types of indexing<br>*Built-in indexes created by default for each property in an entity.<br>* Composite indexes index multiple values of an entity. They are defined in a config file called index.yaml<br><br>Querying: it uses SQL-like = GQL ( graph QL) | Import & export :<br>Entities can be, but not indexes. They are rebuilt during import<br>Possibility of exporting according to filters<br>Exported entities can be imported to bigquery<br><br>*cannot have a wildcard, cannot be appended to an existing table<br>*truncated to 64KB on export |
| BigQuery | Managed , Relational but interacts with NoSQL, Columnar based | petabyte scale, low cost, Interacting with datasets, importin/exporting ,streaming inserts, Monitoring and logging , opt tables and queries, data lifecycle assessment<br>Works with creating datasets(ID, location,expiration ) | Two supported SQL dialects : legacy / standard to be defined when bq command. Standard is preferred cuz it supports advanced SQL features<br><br>Memory allocation : by slots, default is 2000 slots shared across a project. Max processing is 100 GB.<br><br>Stackdriver monitoring : scanned bytes, query time, slots allocated/available, nbre tables in dataset, uploaded rows. To understand how queries and jobs perform. | *load & import :<br>Its data transfer service automates data from a SaaS offering : Ad Manager, Google Asq, play, youtube channel reports, or from amazon S3<br>Data flow can load directly into bigQuery<br><br>*Clustering, partitioning(when splitting according time-timestamp) and sharding (split according to other attriutes )tables<br><br>*streaming inserts:<br>- it provides a de-duplication of rows: ignore duplication of records with IDinsert . it inserts up to 10^5 rows/s and 100MBps. | Load & export :specify type of data source(cloud storage/local drive), data transfer service, data source (URI from cloud storage, bigtable…), file format(avro, csv,json,orc,parquet,possible datastore exports),destination (table,project,dataset), schema, partitioning<br><br>Bigquery expects data to be encoded UTF-8, if not it can convert it with a margin of error<br>Preferred format : avro ( a parallel reading of blocks) |

| | | | | | |
|---|---|---|---|---|---|
| | | | Stackdriver logging : logs entries: insert-update-patch-delete tables, insert jobs, execute querie. To understand who is performing action | Without de-duplication : 10^6 and 1GBps<br><br>*advantage of template tables, use wildcards internal tables. | Optimizing bigquery : avoid select *; use –dry-run to estimate cost, partition when possible, denormalize rather than join, set a max nbre of bytes for a query |
| Cloud memorys tore | Managed redis service, commonly used for caching Create using cloud console ( gcloud) | set ID, size, region and zone, redis version, instance tier basic or standard,memory capacity 1 to 300 GB | Basic tier instance : not highly available, reads/writes blocked during scaling memory,<br>Standard tier instance: include a failover replica in another zone, it can scale while read/write => replica is resized then synchronized with the primary, then primary fail over replica. | If memory used >80% ➔instance under pressure. To fix that: rescale, lower max memory limit,modify eviction policy ( for all keys not only TTL keys), set TTL or delete data. | Managed but demand a monitoring |
| Cloud Storage | Bucket based | *Buckets with unique names,<br>No personally ID names, follow DNS naming, globally unique ID, no sequencially, timestamps naming ( hotspot), possible subnames<br><br>*No filesystem=> no ability to navigate a path, possible use of cloud storage FUSE | Types :Regional, multi regional, nearline, coldline<br>Durability[1] : 99.999999999 %<br>An object can be durably stored but unavailable<br><br>Multi regional minigates risks, lower latency with replicas in multiples regions<br><br><br>Network tiers : GCP offers 2 : standard and premium<br>*Standard : routes between regions using public internet infrastructure | Use cases :mostly handling objects as atomic units<br>* data shared among multiple instances with no persistent attached storage ( log files)<br>*Backup and archival storage (persistent disk snashots…)<br>*as a staging area for uploaded data (app data which is held for a short time before processing | |

| | | | | | |
|---|---|---|---|---|---|
| | | open source project for maping object storage systems to filesystems<br>• No | *Premium : routes over google's global high speed network ( faster ).<br><br>Nearline : data access less than once a month<br>Coldline: data access less than once a year | | |

## Data retention and lifecycle management

Choice of storage system depends on data lifecycle and retention policies, but it does impact how the policies are implemented. data moves through several stages starting with creation, active use, infrequent access but kept online, archived, and deleted. Not all data goes through all stages.

Ex : Cloud storage policies determine when moving data from nearline to coldline storage after some time.

To consider when choosing storage system :

- Submillisecond access time needed : cache such as cloud memorystore
- Frequently accessed, updated, persistently stored : database SQL or NoSQL
- Less likely to be accessed the older it gets : time partionned tables Bigquery or bigtable
- Infrequently accessed and does not require querying: cloud storage ( can be expored to database)
- Not likely accessed but must be archived : coldline storage class in cloud storage
- Policies can be defined on cloud storage according to buckets : moved to another class( coldline), deleted after some time, updated…

Retention policies : object in bucket is not deleted until they reach the age specified : gov & industry regulations ( 1 year in Europe). **Once the policy is locked, it cannot be revoked.**

## Unmanaged databases

GCP offers all managed database in a mode of unmanaged/ self managed. For that, the user must configure certain parameters :

- Updating and patching OS
- Updating and patching database system
- Backing up and recovering

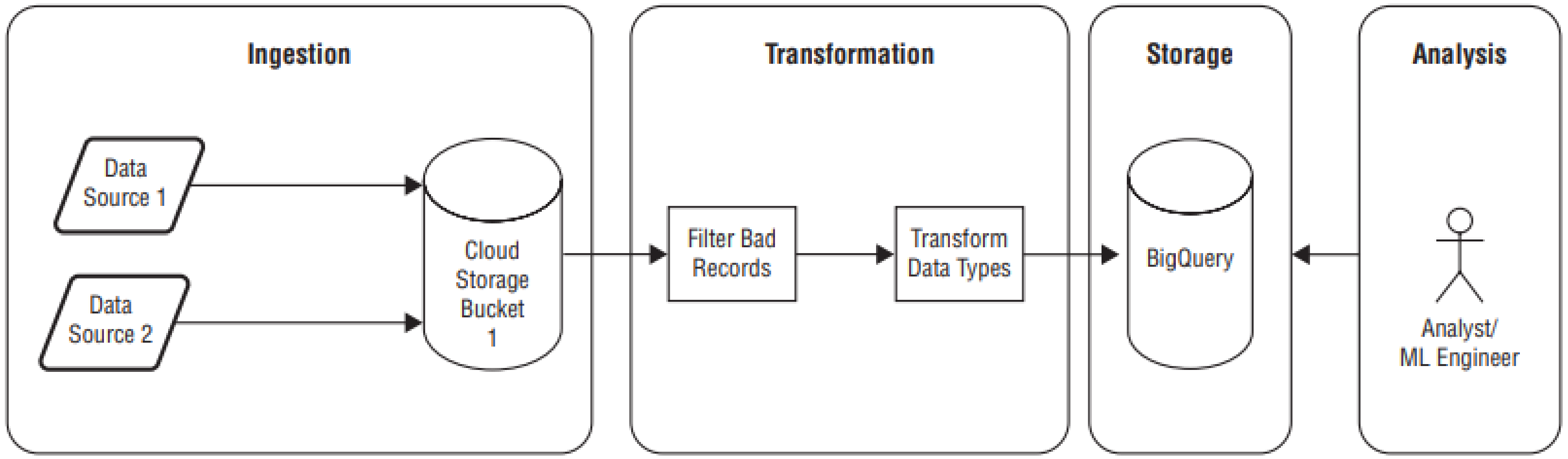
- Network access
- Managing disk space
- Monitor database performance and resource utilization ( stackdriver monitoring and stackdriver logging)
- High availability & manage failover
- Configure and manage read replicas

Importance of stackdriver :

- Monitoring : CPU, memory & I/O metrics
- Logging : info on who created, who accessed, ..etc

# Chapter 3 Designing Data Pipelines

## Data pipelines stages



1. Ingestion : getting data (stream or batch) to google cloud env. Ways : gsutil copying, transfer services, transfer appliance.?
2. Transformation : change data from source structure to stored structure : converting type, fill gaps, aggregation, filtering, augmenting, dropping\adding columns. Tools **: dataflow, dataproc** for transforming batch and streaming data. **Dataprep,datafusion**( more popular) for interactive review and prep for analysis.
3. Storage : can be stored in all chapter 2 systems. But cloud storage is widely used in pipelines for staging and storing data after ingestion, for a long or short term. BQ can treat cloud storage data as external tables. Cloud dataproc uses cloud storage as HDFS compatible storage.
4. Analysis : many forms : SQL querying, report gen, ML, Data science analysis.
   - BQ SQL querying, BQ ML to build ml models using SQL.
   - Data studio to build reports and exploring structured data
   - Datalab interactive workbook ( jupyter notebooks) for data viz, ML, data science
   - Dataproc allows running spark ML libraries while accessing bigtable using Hbase interface for large scale ML models

## Types of data pipelines

There are mainly 3 : Data warehousing pipelines ■ Stream processing pipelines ■ Machine learning pipeline

1. Data warehousing pipelines : for storing data from multiple data sources typically organized in a dimensional data model ( denormalized). Some common patterns are the following :
   a. Extraction, Transformation, and load ETL

   The extraction processes need to be coordinated between different data sources (mostly it's time bases). Example : an output is the join of two databases, if product ID is added in the two databases it requires a new pull request from both databases. If only one is explored, it will lead to errors. Data is transformed before stored in a DB. For that : custom scripts, ETL toolssing GUI or GCP tools :

   - Cloud Dataproc : can be written in spark or Hadoop supported languages : Java,scala,python,R, SQL. Transformations are written according to Hadoop's map reduce model or Spark's distributed tabular data structure. Hadoop provides Pig, a high-level language for data manipulation which compiles into map reduce programs that run on Hadoop. Suitable for migration using existing Hadoop/spark programs
   - Cloud dataflow: Apache Beam model : unifed batch and stream processing model in Python or java( Pipelines, Pcollection: distributed dataset, Ptransform : an operation such as groupby). For writing data: it uses connectors : Bigtable, cloud Spanner,& bigquery. Suitable for new ETL process. Serverless ( no cluster )

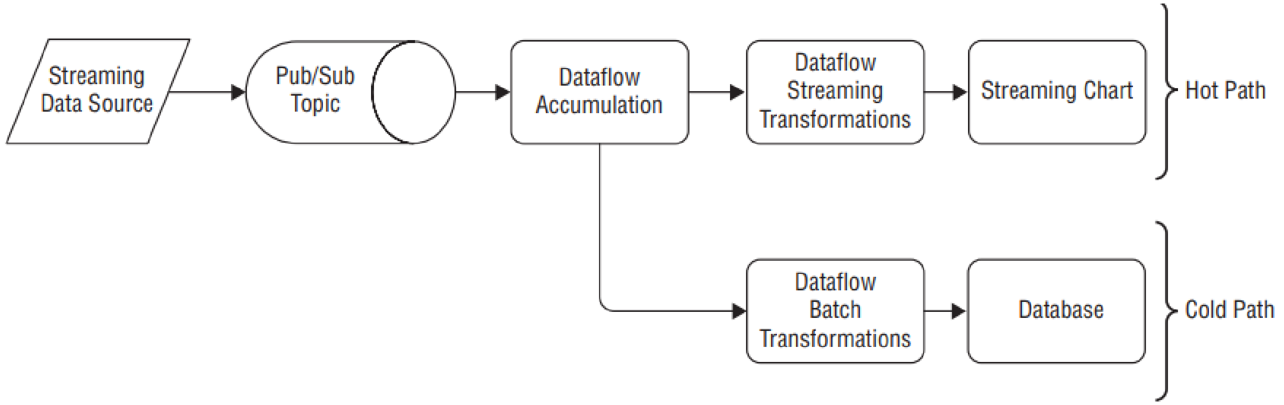
   b. Extraction, Load, and transformation ELT

   Data is loaded before transformation ➔ the DB contains the original data ➔ querying using SQL & transformation using SQL.

c. Extraction and load : No transform
d. Change data capture : it saves changes that the data encountered during the processing / transformation.

These are often batch processing pipelines, But they can have some characteristics of streaming pipelines, especially in the case of change data capture

2. Stream processing pipelines : for IOT, connected objects, monitoring of metrics. Streaming data is similar to batch data but it is not available at once. Thus obligation to pick a subset. ( latest 5 measures or average of last hour arrivings). Factors to consider :
   - Event time and processing time : as it is indicated, select one or consider the difference between event time ( time of happening) and time of processing( time of ingestion or transformation)
   - Sliding and trumbling windows : select the window of data that you wish to consider. trumbling with n steps means moves to i+n element
   - Late arriving data and watermarks : define policy to deal with late arriving data : maybe ignore the late variable on chart after T time but include it later on in the database. Maintain a buffer for late arrivals when working with batchs
   - Missing data
   - Hot path and cold path ingestion



3. ML pipelines : very similar to previous pipelines but have certain specifications, it include :
   - Data ingestion : same services : cloud storage for batch and pub/sub for streaming
   - Data preprocessing ( kinf transformation) : **dataprep** (viz & explore) & dataflow (apply process)
   - Feature engineering ( another transformation) : related to input data, FFT or average
   - Model training & evaluation
   - Deployment

## GCP pipeline components

| Component | Definition | Technology | Interaction with | Commands | Connected through | Extra |
|---|---|---|---|---|---|---|
| Cloud Pub/Sub | Real time mssaging service for push & pull subscriptions<br>Managed service<br>Auto scale & partition loads | Topic(needs only ID), subscriber (customers), publisher (producers)<br><br>Message queus MQ | End points : Cloud functions, App engine, cloud Run | Create topic :<br>gcloud pubsub topics create pde-topic-1<br>subscription :<br>gcloud pubsub subscriptions create --topic pde-topic-1 pde-subscripton-1<br>publish :<br>gcloud pubsub topics publish pde-topic-1 --message "data engineer exam"<br>pull :<br>gcloud pubsub subscriptions pull --auto-ack pde-subscripton-1 | Languages : C#, Go,Java,NodeJS, PHP,Python,Ruby.<br>REST API & gRPC APIs | *Message can stay to 7 days<br>Idempotent : de-duplication<br><br>*arrival not necessarily in order |
| Kafka ( apache) | Similar to pub/Sub<br>Open source | Interaction :<br>*Producer API<br>*Consumer API<br>*Streams API<br>*Connector API | Connector API allows communication with other applications such as pub/sub | | CloudPubSubConnector, is a bridge between the two messaging systems (pub/sub & kafka) using Kafka Connect | It can save data for a long period of time |
| Dataflow | Managed Stream & batch processing : collect,process and output | Written using apache beam API ( beam runner /apache flink)<br>No configuration or region | Pub/sub, BQ,ML engine, Bigtable, Kafka | Create a job using a template file<br>gcloud dataflow jobs run pde-job-1 \ --gcs-location gs://pde-exam-cert/templates/word-count-template | Languages : Python & Java | *Jobs: an executing pipeline<br>* template : separate dev from staging & execution |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | Concepts : Pipelines(series of computation) ■ PCollection (dataset) ■ Transforms (operation) ■ ParDo ( parallel processing) ■ Pipeline I/O ■ Aggregation ■ User-defined functions ■ Runner ( software executer ) ■ Triggers ( functions that triggers another function sending or aggregation) | | | | |
| Cloud dataproc | Managed, Hadoop & spark service<br>Based on clusters<br>Main role : migration from Hadoop clusters to GCP<br><br>*supports manual and auto scaling of clusters<br>* to launch a cluster : specify name, | *Configured with : Hadoop, spark, pig,Hive.<br><br>*Supports ephermal clusters instead of long running Hadoop cluster<br>* conf cluster: nodes : | Hadoop, HDFS, Cloud storage, BQ | Run script by file location in C storage bucket<br>Create cluster :<br>gcloud dataproc clusters create pde-cluster-1<br><br>scaling clusters:<br>gcloud dataproc clusters update pde-cluster-1 --num-workers 10 --num-preemptible-workers 20 | Pyspark files for jobs<br><br>Hadoop, Hbase for datasets, | *Possible usage of HDFS instead of cloud stoage,<br>In this case, it's better to save data in cloud storage too<br><br>*Scaling clusters change only number of worker nodes |

| | | | | | | |
|---|---|---|---|---|---|---|
| | region, zone, mode, nbre of masters & workers nodes)<br><br>* Modes : two :<br>-Standard :run on one master node& N workers (suitable for dev, N can be 0)<br>-High availability mode : 3 masters & N workers | -master nodes: resp for distributing and managing workload dist, runs on YARN& worker nodes<br><br>*worker nodes can include preemetible machines | | *Submit jobs:<br>gcloud dataproc jobs submit pyspark --cluster pde-cluster-1 --region us-west-1 gs://pde-exam-cert/dataproc-scripts/analysis.py | | *autoscaling by policy in YAML file including (maxInstances, scaleUpFactor, scaleDownFactor, cooldownPeriod)<br><br>*it is a good practice to keep clusters in the same region as the Cloud Storage buckets that will be used for storing data. You can expect to see better I/O performance when you configure nodes with larger persistent disks and that use SSDs over HDDs |
| Cloud composer | Managed implementing apache airflow : scheduling and managing workflows automatic | | Built-in integration with BQ,Dataflow, dataproc,datastore, C storage,Pub/sub, AI plateform | | Workflows are defined in Python & are directed acyclic graphs | Before running workflow with Composer, an env must be created on Kubernetes Engine ( specify nbre of node, location, machine type, disk size,…) & create a C storage bucket |

## Migrating Hadoop & spark to GCP

Plan :

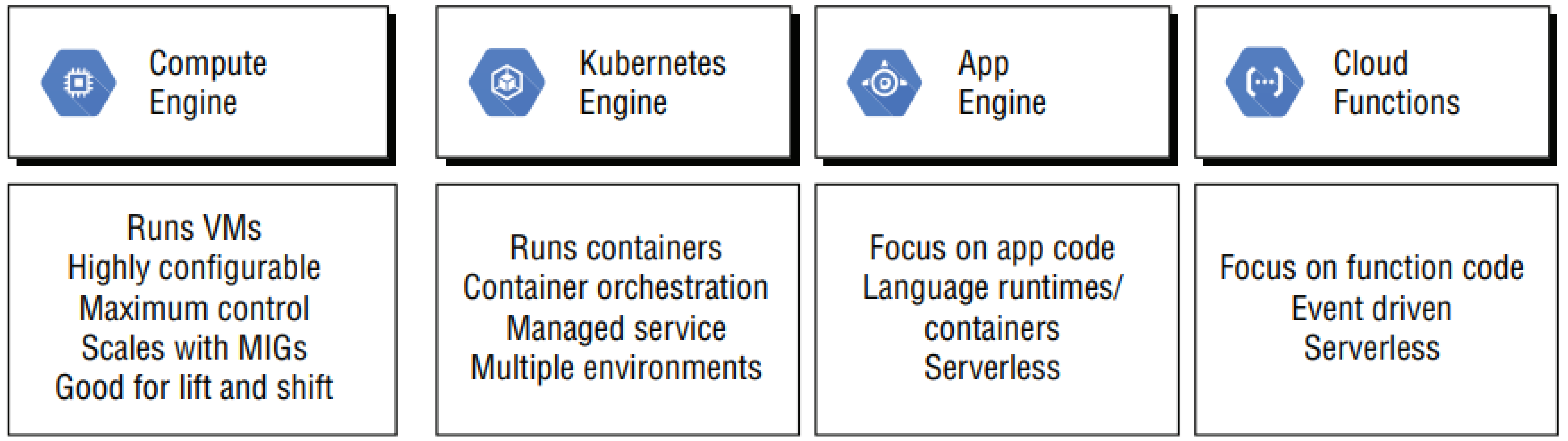
1. migrating data : some data to cloud storage to run ephermal clusters
2. migrate jobs
3. migrate Hbase to Bigtable: step by step, from Hbase to Bigtable passing by cloud storage.
   From Hbase sequence file to C storage, then to Bigtable using Dataflow.
   Less than 20 TB use distcp ( Hadoop copy command),>20 Tb use Transfer appliance.

# Chapter 4 Designing a Data Processing Solution

## Designing infrastructure

1. Choosing infrastructure : appropriate to the use case
   a. Compute engine: managed service, Infrastructure as a service IaaS. Greatest amount of control over the provided VM instances, choice of OS system, type of machine ( vCPUs, RAM, GPUs, TPUs..etc), full access to OS: installing and configuring extra software, security features ( shielded VM & accelerators GPUs), region, zone.
   VMs can be grouped in clusters for higher availability & scalability. A managed instance group is a set of VMs identically configured that are managed as a single unit, configuration demand min & mx number of instances ( for scaling up or down)
   b. Kubernetes Engine:
   managed Kubernetes service for deploying containerized applications on a cluster of servers on managed instance groups. Advantage : tune the allocation of cluster resources to each container ( not necessarily equal distribution), useful when having a set of microservices and some run more or less than others. Kubernetes can run on other cloud providers & in on premises data centers.
   Constraint : applications must be containerized to run on Kubernetes, for that, a best option is a lift & shift migration to compute engine is the fastest way to get the cloud, once there, applications can be containerized. It uses pods.
   c. App Engine :
   managed service , Platform as a service PaaS : minimize the need to support the infrastructure.
      i. App engine Standard: run apps in a language specific serverless env: GO,Java, PHP,NodeJS, python. Available memory& CPU : from 256MB and 600MHz to 2048MB and 4.8 GHz.
      ii. App engine Flexible : Runs on Docker containers=> customize the runtine env : run a docker image, install addition libraries & packages, then deploy container instances based on that image.+ health checks. Suitable for a scale based on workload.
   d. Cloud Functions : Serverless managed service. Run code in response to events. Example : writing a message to pub/sub topic. Responds to events in HTTP(Get post put depete options), Firebase & stackdriver logging. Languages: Javascript, Python3 and Go.

**While App Engine supports many different services within a single application, Cloud Functions support individualized service s.**



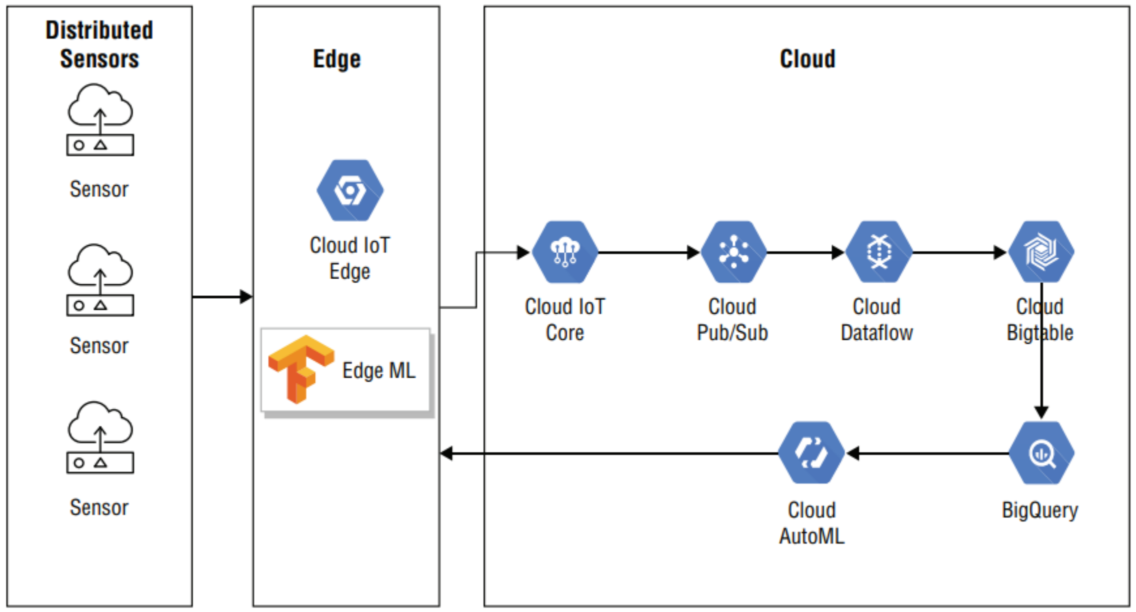
| Compute Engine | Kubernetes Engine | App Engine | Cloud Functions |
|---|---|---|---|
| Runs VMs<br>Highly configurable<br>Maximum control<br>Scales with MIGs<br>Good for lift and shift | Runs containers<br>Container orchestration<br>Managed service<br>Multiple environments | Focus on app code<br>Language runtimes/ containers<br>Serverless | Focus on function code<br>Event driven<br>Serverless |

2. Availability, reliability and scalability of infrastructure : definition in appendix
    a. In compute resources :
        i. Compute engine : you are also responsible for configuring managed instance groups, load balancers, and autoscalers ( with defined policy). This is done with managed instance groups (MIGs) defined using a template. All members of a MIG are identical. The global load balancers are HTTP(S) Load Balancing, SSL Proxy, and TCP Proxy. The regional load balancers are Network TCP/UDP, Internal TCP/UDP, and Internal HTTP(S).
        ii. Kubernetes Engine : designed for high availability reliability and scalability: pods fail ➔ replaced. Failed nodes ➔ reprovised with autorepaire. You can also specify if access to clusters is zonal or regional. Replications of across zones.
        iii. App engine + Cloud functions : highly available, scalable, and reliable.
            a. App engine : send to existent instance or create a new one ( depends on config). Scaling policies.
            b. Cloud Function : one request at a time, if an over workload => additional function instances created.

    b. In storage resources
        i. Memorystore (redis ) : high availability, not scalability
        ii. Persistent disks : for Compute & kubernetes engines. built-in redundancy for high availability and reliability.

    c. In network resources : standard tier ( public internet network), premium tier ( google's global network). Use of VPN : 99.9% availability. HA VPN ( high availability ) : 99.99 % .interconnect & partner interconnect
3. Hybrid cloud : Runs on both cloud and on premises machines
4. Edge computing :

Suitable for IOT applications that demand ML. Rather than sending data to the machine learning model running in the cloud, the model could be run near the monitored machine using an Edge TPU, which is a tensor processing unit (TPU) designed for inference at the edge. It needs continuous Integration/Deployment CI/CD. Consider using containers.



# Chapter 6 Designing for Security and Compliance ( small chapter)

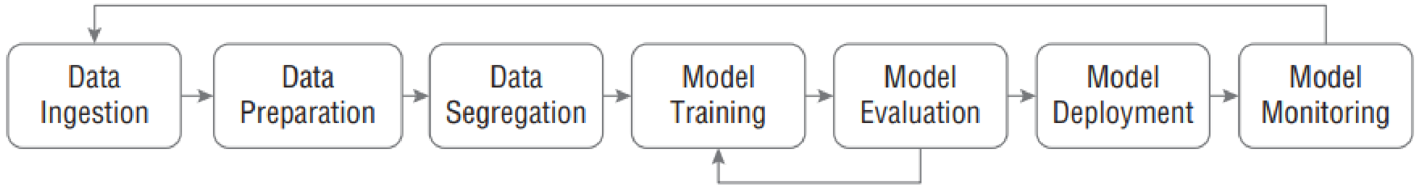
## Identify and access management IAM

are collections of permissions that can be assigned to identities. Permissions are granted to roles, and then individuals can be assigned multiple roles.

1. Primitive roles : Viewer(read only), Owner( read + modify the state), Editor (read, modify+manage permissions & billing) not recommended (only in fewer people with high privilege ). **Apply on project level. On datasets level** : Reader, Writer( create, change delete tables), Owner ( create update delete datasets)
2. Predefined roles : apply on service (app engine, BQ) : roles/service.entity.operation example : roles/appengine.versions.create
3. Custom roles : assign one or more permissions to a role then assign role to users, group or service account. For example a sub permission of a predefined role.

# Chapter 9 Deploying Machine Learning Pipelines ( relatively small chapter)

## Structured ML pipelines



1- Data ingestion : stream or batch
2- Data preparation : by order : data exploration, data transformation, and features engineering
3- Data segregation : splitting data to train validation and test
4- Model training : fitting
5- Model evaluation : with
    a. Individual metrics : accuracy, precision, recal, F1 score
    b. K fold cross validation
    c. Confusion matrix
    d. Bias and variance : MSE
6- Model depployement
7- Model monitoring :
    a. Performance monitoring : how well the model is using resources
    b. Stackdriver monitoring : resource utilization
    c. Stackdriver logging : capture information on events while model running

d. Monitoring accuracy

## GCP Options for Deploying Machine Learning Pipeline

1. Cloud AutoML : designed for applying ML without having to learn the details. Equipped with with GUI to train and evaluate models. Products :
   a. AutoML vision : vision classification & vision edge- image classficiation, Vision object detection, vision edge- object detection
   b. AutoML Video intelligence : classification and Object tracking.
   c. AutoML Natural Language : analyse docs and classify them, identify text, sentiment or attitudes from text.
   d. autoML Translation : custom translation models
   e. auto ML Tables : for structured data, ( include pre-processing : normalize, fill gaps & features engineering). Suitable when wanting to optimize model without lots of experiments with different algos and feature engineering. However, it takes long to return model since it tests a variety of models : linear regression, deep NN, gradient-boosted decision trees, AdaNet …It allows Tables to determine the best algo.
2. BigQuery ML : allow ML models using SQL and data in BQ. ML methods available through SQL + not having to move data. It can be accessed through : BQ web user interface, bq command line tool, BQ REST API, external tools such as jupyter NB. It supports several ML algos : Linear Regression, Binary logistic regression, Multiple logistic regression, Kmeans clustering, Tensorflow model importing to import TF models. Suitable for short time model generation.
3. KubeFlow : open source adapted to Kbernetes platform. A multicloud framework for running ML pipelines. It supports : training TF models, TF serving ( used to deploy trained model ), jupyterHub installation ( platform managing multiple instances of 1 user Jyper notebook server), Kubeflow pipelines ( define ML workflow )
4. Spark ML : included with spark a library called MLib. Suitable when using cloud Dataproc or self managed spark cluster. It supports : ML algos( SVM, Linear regression, Decision tree, random forests, gradient boosted trees, naïve bayes , K means, Latent Dirichlet allocation, PCA and Signle value decomposition, Frequent pattern) , features engineering , data transf, dimentionality reduction, ML pipelines for exceuting multistep workloads, other libraries and data management tools.

## Appendix A

Semi structured data had document and wide columns structures

- Document oriented are NoSQL such as json
- Wide column oriented type of columnar database that supports a column family stored together on disk, not just a single column

- [1]**Data durability**: the ability to keep the stored data consistent, intact without the influence of bit rot, drive failures, or any form of corruption. 99.999999999% (11 nines) durability means that if you store 10 million objects, then you expect to lose an object of your data every 10,000 years.
- **Failover :** In the event of an instance or zone failure, the standby instance becomes the new primary instance. Users are then rerouted to the new primary instance.
- [2]**Availability** is the ability to access an object when you want it
- **Multi regional** : multiple regions with different zones
- **Persistence:** Persistent disks. Persistent disks are durable network storage devices that your instances can access like physical disks in a desktop or a server.
- **Add hoc** : customized queries according to a value or a variable

- **Hotspot in spanner** : read and write on same node. It happens when using sequential Primary key( times timestamps, incrementing counters). Solution is to avoid high latency when joining tables, it uses interleaving data : save order row with order line item!
- **Wildcard table in BQ** : wildcard table enables you to query multiple tables using concise SQL statements. Example : table_1,table_2 table_31 can be all be called using 'table_*'
- **HDFS** : Hadoop distributed file system
- **Hadoop**: This is an open source, big data platform that runs distributed processing jobs using the map reduce model. Hadoop writes the results of intermediate operations to disk.
- **Spark**: This is another open source, big data platform that runs distributed applications, but in memory instead of writing the res ults of map reduce operations to disk.
- **Pig**: This is a compiler that produces map reduce programs from a high-level language for expressing operations on data.
- **Hive**: This is a data warehouse service built on Hadoop.
- **Ephemeral clusters**( dataproc) : a large cluster can be created to run a task and then destroyed once the task is over in order to save costs. (=/= on premises clusters)
- **Data on premises** : refers to local hardware, meaning data is stored on local servers, computers or other devices.
- **A Preemptible VM** (PVM) is a Google Compute Engine VM instance that can be purchased for a steep discount as long as the customer accepts that the instance will terminate after 24 hours.
- **YARN** : Yet Another Resource Negotiator, uses data about the workload and resource availability on each worker node to determine where to run jobs.
- Kubernetes is a container orchestration system
- **Lift-and-shift migration:** you move workloads from a source environment to a target environment with minor or no modifications or refactoring. in which the system architecture is not changed.
- **Serverless** allows developers to build and run applications without having to manage servers
- **Availability:** the ability of a user to access a resource at a specific time. Availability is usually measured as the percentage of time that a system is operational. Availability is a function of **reliability,** which is defined as the probability that a system will meet service-level objectives for some duration of time, measured as **the mean time between failures**.
- **Scalability** is the ability of a system to handle increases in workload by adding resources to the system as needed.
- ***Pods*** are the smallest deployable units of computing that you can create and manage in Kubernetes.
- **Roles** : are collections of permissions that can be assigned to identities. Permissions are granted to roles, and then individuals can be assigned multiple roles.
- Precision = True Positive / (True Positive + False Positive)

- Recall = True Positive / (True Positive + False Negative)
- F1score = 2 * ((precision * recall) / (precision + recall))

## Appendix B : important concepts not detailed previously ( from other courses or examtopics)

- **Dataprep** by is an intelligent data service for visually exploring, cleaning, and preparing structured and unstructured data for analysis, reporting, and machine learning. Because Dataprep is serverless and works at any scale, there is no infrastructure to deploy or manage. Your next ideal data transformation is suggested and predicted with each UI input, so **you don't have to write code**.
- If data is already present in a data lake, no need for ingestion since it is already ingested, best option is to move to exploring or transforming depending on the ML pipelines using dataprep
- **Google Cloud transfer** options include:
  - **Transfer Appliance** for moving offline data, large data sets, or data from a source with limited bandwidth. Only from on premises to cloud
  - **BQ Data Transfer Servic**e to move data from SaaS applications to BigQuery. supports loading data from the various data sources like Cloud Storage, Google Ads, Campaign Manager, Youtube channel reports and few more, between two BigQuery datasets..
  - **Transfer service** for on-premises data to cloud or within multicloud, or from other cloud suppliers
- **gsutil** : cp command allows to copy data between local file system or on-prem and the cloud, within the cloud, and between cloud providers.
- **Cloud Data Fusion**: fully-managed data engineering product. helps users in ETL/ELT data pipelines. it leverages a convenient user interface for building data pipelines in a drag and drop manner.
- **Exponential backoff** : Truncated exponential backoff is a standard error-handling strategy for network applications. In this approach, a client periodically retries a failed request with increasing delays between requests. Clients should use truncated exponential backoff for all requests to Cloud IoT Core that return HTTP `5xx` and `429` response codes, as well as for disconnections from the MQTT server.
- **Gcloud daflow jon drain** : Once Drain is triggered, the pipeline will stop accepting new inputs. The input watermark will be advanced to infinity. Elements already in the pipeline will continue to be processed. Drained jobs can safely be cancelled.
- **Federated queries :** is a way to send a query statement from BQ to an external database (or a file) and get the result back as a temporary table. Federated queries use the BigQuery Connection API to establish a connection with the external database. For example : Cloud SQL, spanner, CSV file in CS.

- **gsutil or Storage Transfer Service**: Transfer scenario Recommendation
  Transferring from another cloud storage provider Use Storage Transfer Service.
  Transferring less than 1 TB from on-premises Use gsutil.
  Transferring more than 1 TB from on-premises Use Transfer service for on-premises data.
  Transferring less than 1 TB from another Cloud Storage region Use gsutil.
  Transferring more than 1 TB from another Cloud Storage region Use Storage Transfer Service.
- **HDFS with Hive** - Hive allows users to read, write, and manage petabytes of data using SQL. Hive is built on top of Apache Hadoop, which is an open-source framework used to efficiently store and process large datasets. As a result, Hive is closely integrated with Hadoop, and is designed to work quickly on petabytes of data. HIVE IS NOT A DATABSE.
- **HBase** is a distributed non-relational database management system, written in Java, with structured storage for large tables.
- **ODBC:** ( used with BQ to connect to databases) uses standard SQL querying (not legacy) and connection is only allowed with a *service account*.
- **BQ data transder service** : initially supports Google application sources like Google Ads, Campaign Manager, Google Ad Manager and YouTube but it **does not support destination anything other than bq data set**
- **OpenCensus** is a free, open source project whose libraries: Provide vendor-neutral support for the collection of metric and trace data across multiple languages. And Can export the collected data to various backend applications**, including Cloud Monitoring, by using exporters.**
- **Cloud Data Loss Prevention** :Fully managed service designed to help you discover, classify, and protect your most sensitive data. You can Deploys de-identification in migrations, data workloads, and real-time data collection and processing using DLP
- Strong consistency is **one of the consistency models used in the domain of concurrent programming** (e.g., in distributed shared memory, distributed transactions). The protocol is said to support strong consistency if: All accesses are seen by all parallel processes (or nodes, processors, etc.)
- eventual consistency is only a liveness guarantee (updates will be observed eventually),
- A synthetic feature is **a combination of the econometric measures using arithmetic operations (addition, subtraction, multiplication, division)**.
  a. Des examens sur udemy : https://www.udemy.com/course/google-cloud-professional-data-engineer-gcp-exams/