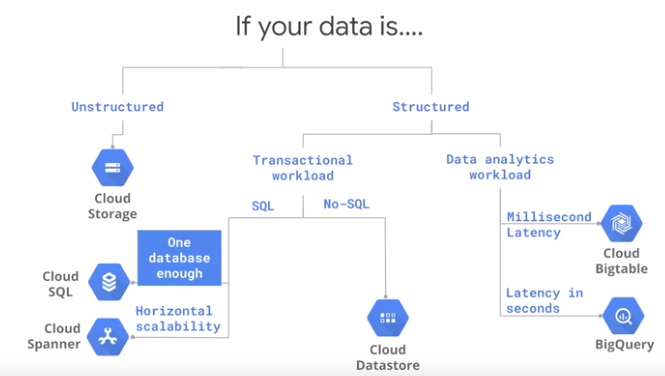
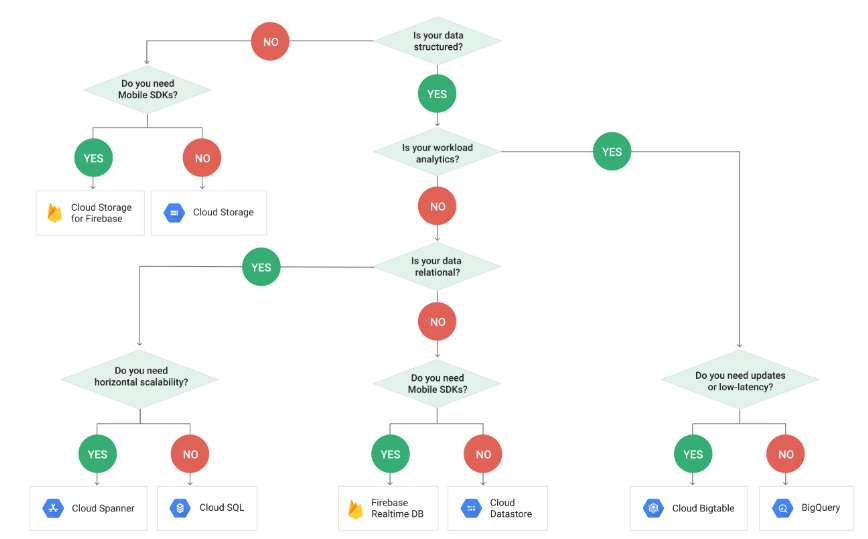
**GCP Notes**



**Storage**

****



**Cloud Storage**

Object storage system. Designed for persisting **unstructured data** (files, video, backup files.. )

A bucket is a group of objects that share access control at bucket level. Individual object can have their own controls as well.

If **renaming a bucket** -> need to **copy and delete bucket**

Avoid sequential bucket names to avoid hot spot

It is not filesystem but naming convention.

Type of storage:

* **Regional storage** -> store multiple copies of an object in multiple zones
* **Multi regional storage** -> replicas in multiple regions. Avoid regional outage risk / improve latency if users are in different regions (exp : 5 ms latency vs 30 ms latency)
* **Nearline storage** -> access less than **once a month**
* **Coldline storage ->** access **less** than **once a year**

Latency:

* Standard -> public internet
* Premium -> Google high speed network

**Cloud SQL**

Fully managed relational database that support MySQL

Regional support **up to 30 TB**, more => cloud spanner

Will perform only daily backup (can be specified)

High availability option => create second instance in second zone (in case of failure)

Database = **max 10 000 tables**

**Read replicas improve reads (in same region).** Maintenance can occur on read replicas

**Fail-over replicas** are used for high availability

Cloud SQL proxy provide secure access, without having to configure SSL

**Cloud Spanner**

Google’s relational, **horizontally scaled**, global database.

**Server based**

Highly available: does not require failover instance (automatic replication with voting mechanism)

Price: regional + single node 0.9$ per hour / multi regional 9$ per hour

Recommend keeping **CPU utilization in an instance below 65% regional / 45% multi regional** -> else open new instance

**Each node up to 2TB**

Data import/export cloud storage bucket from AVRO / CSV files -> process with cloud dataflow connectors.

Not the best option for IOT

Cloud spanner support both **primary and secondary index**

**Interleave tables** for related data:

* Parent-child relationship
* More efficient for joins than store it separately
* Support **up to 7 interleaves tables**

*exp : CREATE TABLE table1(………) PRIMARY KEY(orderId)*

*CREATE TABLE table2(………) PRIMARY KEY(nameId) INTERLEAVE IN PARENT table1 ON DELETE CASCADE*

Avoid hot spot:

* **hash is not recommended** (hash can be use with bigtable)
* Use **UUID that generate random identifiers**

Cloud spanner breaks data into chunks known as **splits:**

* Up to **4 GB per split**
* Range of **rows in a top-level table**
* Rows that are **interleaved** are kept together -> parent-child (interleave) = **max 4 GB**

Secondary indexes are useful to query with a WHERE close

It is possible to examine **execution plan** (query). You also can force index if you want to modify execution plan. (*SELECT \* FROM table1 @{FORCE\_INDEX = indx})*

You can use parametrized queries -> *WHERE … = @parameter*

**STORING clause** is used to create indexes that can answer **(very frequent) queries** just using the index

**Mongo DB** (document) like data can be stored **in STRUCT object**

**Bigtable**

**Server-based**

Wide-column petabyte-scale, fully managed **NoSQL** database service for large **analytical and operational workload** used for high volume databases that requires **low millisecond latency**.

Used for IOT, time series, finance..

**Scalable** => 3-nodes cluster (30 000 rows/sec) -> 6-nodes (60 000 rows/sec)

Tables are denormalized

Avoid sequential data to avoid hotspot

* **Field promotion.** Move fields from the column data into the row key to make writes non-contiguous. (**recommend**)
* **Salting.** Add an additional calculated element to the row key to artificially make writes non-contiguous.
* **Hashing**

**Don't exploit atomicity of single rows, Rows can be big, but not infinitely big**

Import/export : use cloud dataflow process

Bigtable instance with line SDK or REST API:

* The instance can be either in production or in development (**can upgrade to prod cannot downgrade to dev** after creation)
* Storage type SSD pr HDD (**cannot** be changed after creation)
* Specific region or zone (**cannot** be changed after creation)
* Number of node /cluster (**can** be changed after creation)
* Name (**can** be changed after creation)



Use **HDD** if you store at least **10 TB and latency is not important**.

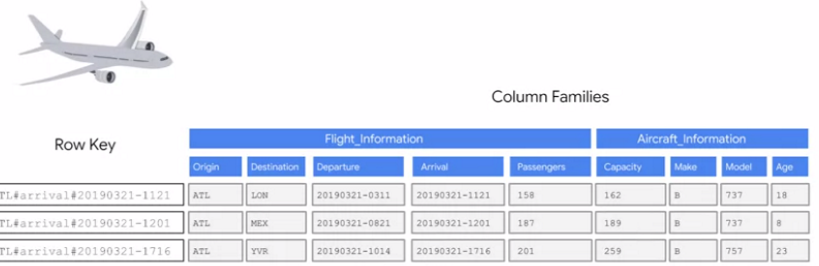
Bigtable support **up to 4 replicated clusters** (all clusters must be in their own zone / in different region increase latency but prevent failure in a region)

Consistency:

* If it is needed all user get **exactly same** read -> specify **strong consistency** (in app profile) => **all traffic route to the same cluster** (other clusters are used only for failover).
* If you can **tolerate difference** between instance for a short period of time -> use **eventual consistency** -> lower latency

Three dimensions:

* **Rows** -> **indexed by a row-key (atomic level)**
* **Columns** -> can be grouped in column family
* **Cells** -> by default the latest timestamp



Tables are sparse

Google recommend storing no more than **10 MB in a single cell** / **100 MB in a single row**

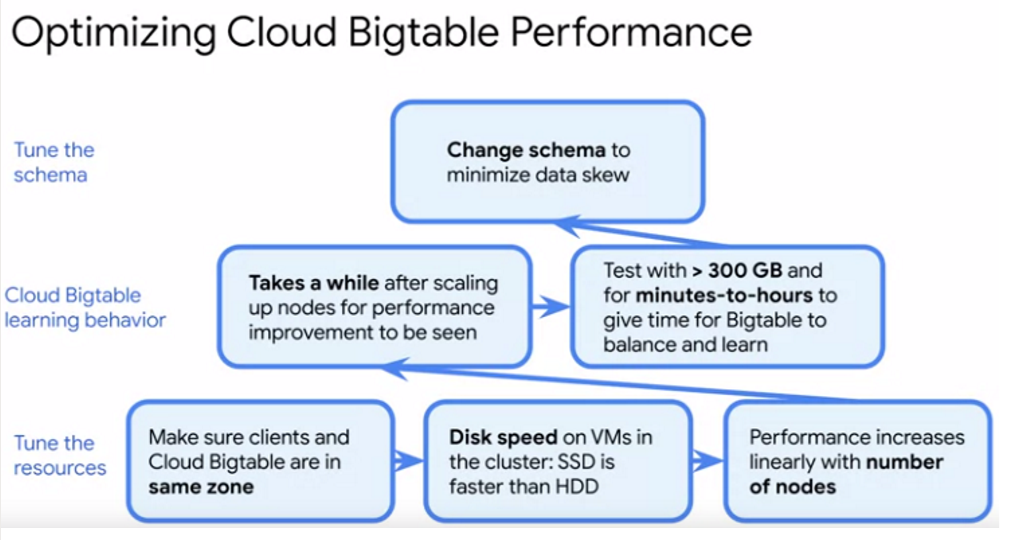
**Data size have to be more 1TB for performance** (fewer data => lower performance)

Databased are deployed on a set of instances (VM)

**Metadata and logs** are stored in **Google’s Colossus file system in SSTables** (sorted string tables) called tablets

Data from **key visualizer heatmap** -> dark = low activity / **bright = heavy activity**

It is better having few columns and many rows (if you have 2 sensors with different telemetry -> make 2 tables)



You can **analysis outlier** data in time series using cbt tool

If a node fails, no data is lost.

**Access** can be defined only at **project level**

Command lines example:

*gcloud bigtable instances create pde-bt-instance1*

*--cluster=pde-bt-cluster1 \*

*--cluster-zone=us-west1-a*

*--display-name=pdc-bt-instance-1 \*

*--cluster-num-nodes=6 \*

*--cluster-storage-type=SSD \*

*--instance-type=PRODUCTION*

**You can also use cbt command**

*cbt createinstance pde-bt-instance1 pdc-bt-instance-1 pde-bt-cluster1 west1-a 6 SSD*

**Cloud Firestore**

**Serverless**

Managed **Document database** (replace cloud datastore)

2 modes:

* **Native mode** -> **real time** update, **mobile web** client library features available. Eventual consistency
* **Data store Mode** -> **strong consistency**, 1 write/sec to an entity group

Make random distribution of identifier to avoid hot spot

2 Query types:

* **Simple Query** -> simple index (color = “red”)
* **Complex Query** -> composite index -> define it in index.yaml (color = “red” AND age=”2”). Multiple indexes lead to greater storage sizes.

If index.yaml is NOT complete => no return

**Cloud Memory Store**

Managed **Redis Service**

Memory capacity range fom 1 to 300 GB

**TTL parameters** specify how long **key will be kept in cache** before being eligible for eviction (the longer => can query be faster because more likely to be in cache)

Useful as a **cache** for storing data that should not been lost **when a machine fails**

**Ingestion**

**Cloud Dataflow**

**Serverless**

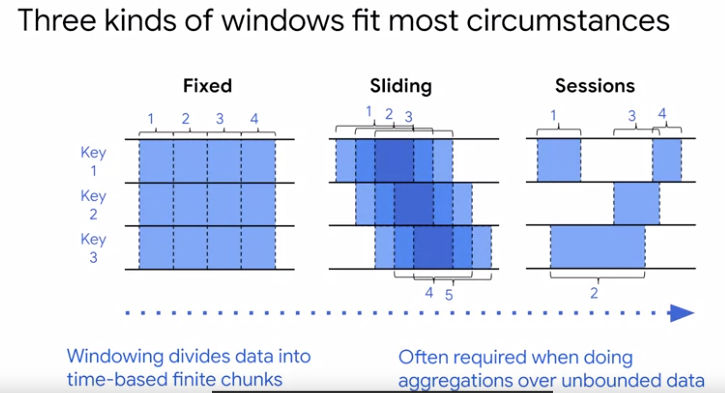
Managed stream and batch processing service

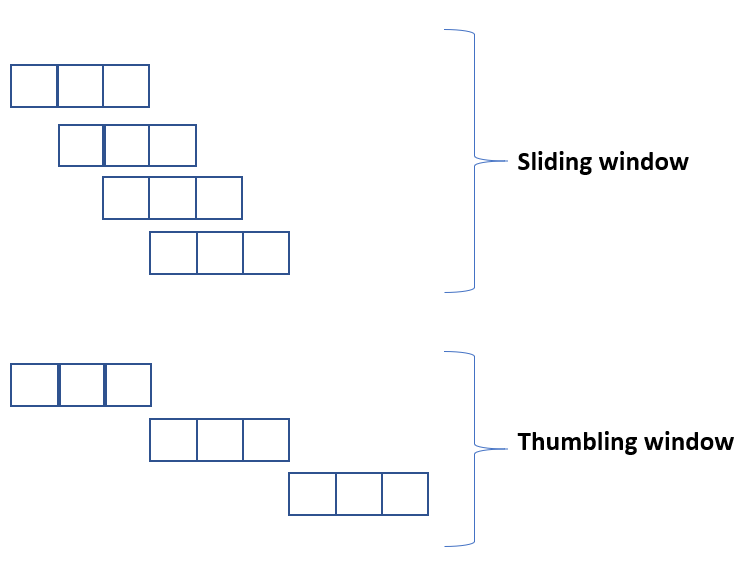
**Autoscaling**

Use Apache Beam (currently Python or Java)

Link with different Cloud services, Apache Kafka…

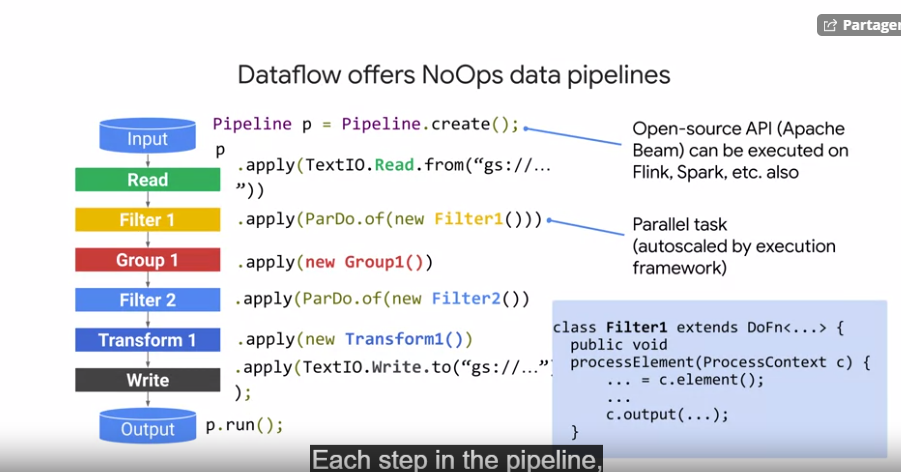
Window **default = Single, Global Window**





Elements in Dataflow:

* **Watermarks** -> timestamp indicate that no older data will ever appear in stream
* Pcollection -> contain fixed set of data
* ParDo -> Transform in parallel. the ParDo transform processes elements independently and possibly in parallel
* Transforms -> operation like loops, condition etc
* Pipeline I/O -> reading, writing data (cannot use ParDo to write output in parallel)
* Aggregation
* UDF -> User defined function
* Runners -> software that execute pipeline jobs
* Triggers -> trigger a job (control when the elements for a specify key or window are output) (not based on element size bytes)



User can run jobs using template with parameters (Google template or new templates)

you can also specify number of nodes to use by default when executing pipeline as well as max workers

There is no Dataflow cross pipeline. Is different pipeline need to share data, you have to use storage like google cloud Storage or an in-memory cache like App Engine.

Cloud dataflow developer role on a project as it permits to work on the pipeline without giving access to the data

A drain parameter permit to stop instance and wait all process is finished. Useful if you need to modify the code.

**Cloud Dataproc**

**Server-based**

Managed Hadoop and Spark service where a preconfigured cluster can be created with a command line or a console operation

Equivalence:

* Cloud Storage -> HDFS . **But if you need heavy I/O, use local HDFS in cluster**
* Bigtable -> HBase
* Flink -> Dataflow

You can use Hadoop, Spark Pig, Hive…. In clusters

Cloud Dataproc allow possibility use “ephemeral” cluster -> run a task and the destroyed the cluster

* Only **90 seconds to run a cluster**

It is a good approach to use Cloud storage to store data instead of copy data when cluster run

**Initialization** can run when cluster is created with a specific **script files** located in a cloud storage bucket

Support **Autoscaling** specify in a **yaml file** (max instance, scale up factor)

Jobs are submitted using API, gcloud command or console.

Migrating Hadoop and Spark jobs to GCP:

* First step -> migrate some Data to cloud storage
* Then deploy “ephemeral” clusters to run jobs

Migrating HBase:

* Sequence file copy to cloud storage
* Next import **files in Bigtable using Cloud Dataflow**
* If **size > 20TB** -> use **Transfer Appliance**
* If size **< 20 TB** AND there is at least 100 Mb of network bandwidth available the **distpc, a Hadoop copy command** is recommended

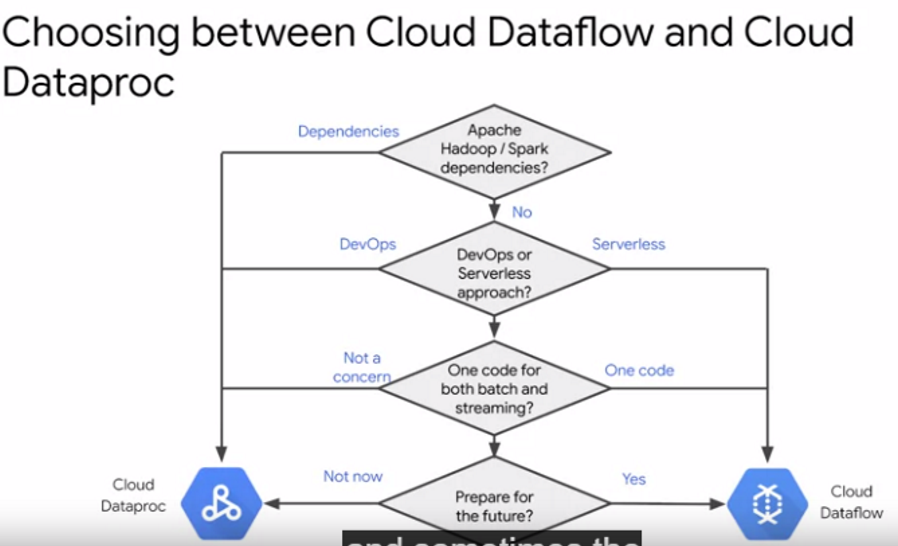
Instance created with command line SDK or REST API

* Name, region, zone, cluster mode, machine type, autoscaling policy

Cluster node determine number of masters:

* **Standard mode** -> 1 master / some workers
* **Single mode** -> 1 master
* **High availability mode** -> 3 masters and some workers

**Cannot change mode after a deployment** -> have to recreate to be modified. You **can change number of worker** (a number or autoscaling parameter) but **not the number of master node**.



It is possible to use Bigquery with Bigquery connector from Dataproc. Tempory files are not automatically deleteted if jobs failed.

Command lines example:

*gcloud dataproc clusters create pde-cluster-1 \*

*--region us-central1 \*

*--zone us-central1-b \*

*--master-machine-type n1-standard-1 \*

*--master-boot-disk-size 500 \*

*--num-workers 4*

*--worker-machine-type n1-standard-1*

*--worker-boot-disk-size 500*

* If you create a Dataproc cluster with [internal IP addresses only](https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/network#create_a_cloud_dataproc_cluster_with_internal_ip_addresses_only), attempts to access github.com over the Internet in an initialization action will fail unless you have configured routes to direct the traffic through [Cloud NAT](https://cloud.google.com/nat/docs) or a [Cloud VPN](https://cloud.google.com/network-connectivity/docs/vpn). Without access to the Internet, you can enable [Private Google Access](https://cloud.google.com/vpc/docs/private-google-access), and place job dependencies in [Cloud Storage](https://cloud.google.com/storage); cluster nodes can download the dependencies from Cloud Storage from internal IPs.
* You can use [Dataproc custom images](https://cloud.google.com/dataproc/docs/guides/dataproc-images) instead of initialization actions to set up job dependencies.
* Yarn can graceful decommissioning a node for minimal cost impact
* Yarn Access -> via SOCKS proxy

**GCP Cloud Composer**

Managed service implementing Apache Airflow which is used to manage workflows

* Workflows are defined using Python -> DAGS
* Before you can run workflows with Cloud Composer -> need to create environment in GCP
* Environments are **standalone deployments based on Kubernetes**
* **Airflows log -> stored in Cloud Storage** (logs folder)
* **Streaming logs -> stored in stackdriver** (use logs viewer to check it)

Cloud Composer is an Apache Airflow managed service, it serves well when orchestrating interdependent pipelines, and **Cloud Scheduler is just a managed Cron service.**

**Cloud Pub/Sub**

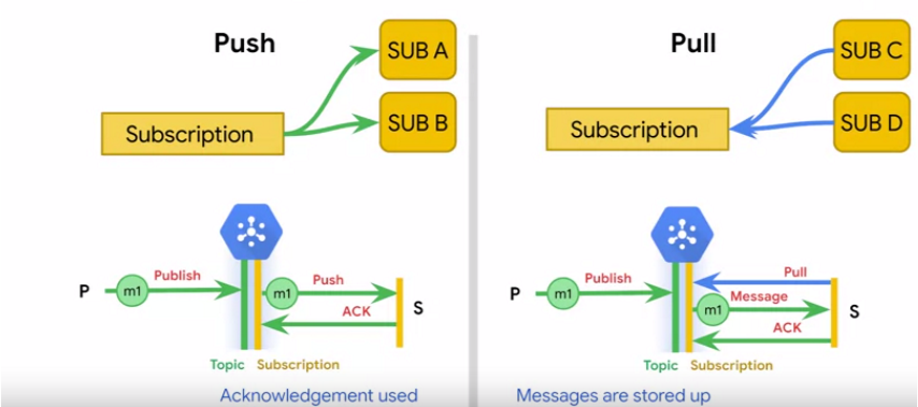
Messaging queues are used in distributed system to decouple service in a pipeline

Kafka like. Difference: with Kafka you can reread/replay messages (**not possible with pub sub**)

Clients can publish up to 1000 Mb per seconds

Subscription:

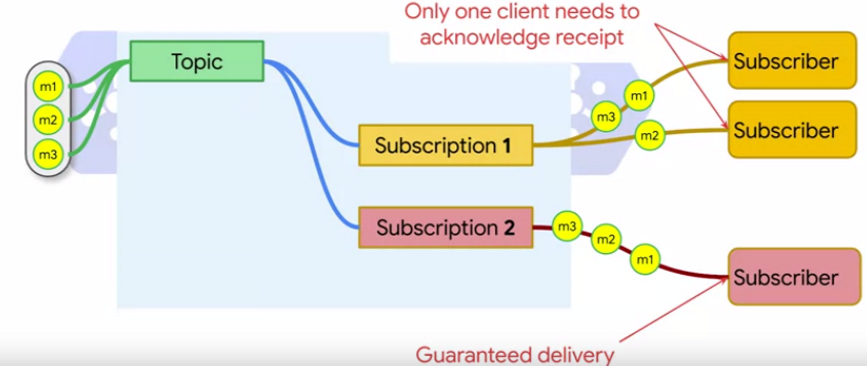
* **Push subscription** -> deliver message **to endpoint** (like cloud Functions, App Engine, Cloud run services…). Consume up to 100 MB per second. If connected **with app engine or cloud function** or same webohook.
* **Pull subscription** -> You will consume **the N next messages** Consume up to 2000 MB per second. If much **more than 1 message per second, or throughput of messages are critical, no HTTPS endpoint** within non-self signed SSL. The subscriber control the rate of delivery (dynamically feasible)
* **Automatically deleted after 31 day of in activity.**



**Dataflow cannot** beconfigured as **push** endpointwith cloud pubsub

**Topics** can be created in console or command line (*gcloud pubsub topics create top1*)

**Subscription** can be made in console or command line (*gcloud pubsub subscription create --topics top1 subs1*)



**Acknowledgments** indicates message have been read and processed so that it can be removed from topic

Message can be stored **up to 7 days (can be configured between 10 min and 7 days)**

If need guarantee messageId not count in duplicates time:

* use Cloud Dataflow Pubsub I/O
* assure messages are processed in order

If need to continue using Kafka -> link Cloud Pub/sub and Kafka with Cloud Pub/sub connector

**Watermarks** -> a message queue could be configured to drop data that is received later than threshold period

**Attach timestamp to events are not automatically** made. (need to active the attachment)

Command lines example:

*gcloud pubsub topics create pde-topic-1*

*gcloud pubsub subscriptions create --topic pde-topic-1 pde-subscripton-1*

*gcloud pubsub topics publish pde-topic-1 --message "data engineer"*

*gcloud pubsub subscriptions pull --auto-ack pde-subscripton-1*

**Cloud Transfer Service / Transfer Appliance**

Uploading large volume of data

**Cloud transfert service** -> **quickly import online large dataset** to Google Cloud Storage

Bigquery data transfert service -> planned automatic transfer from your Saas to Bigquery

To use to migrate google storage bucket to other google storage buckets

**Appliance Transfer service** -> migrate large database (**> 20 TB**) to Google Cloud Storage. Hydrator can be used to decrypt the data.

**Gsutil** is a tool for programmatic usages. Copy MB/GB of data. **Not TB Not reliable for recurrent transfer**

**Kubernetes**

**Server-based**

Container orchestration system and Kubernetes engine is a managed Kubernetes service

Google maintain cluster, installations, configuration platform on cluster

User can precisely tune the allocation of cluster to each instance

It can run in multiple environments including in other cloud providers

Application must be containerized to run on Kubernetes

**Cluster master** run 4 services that control cluster:

* Controller manager -> run services (deployments, replicas)
* API server -> make calls to the master
* Scheduler -> when to run a pod
* etcd -> distributed key-value across the cluster

**Nodes**:

* Instances that execute workloads. Implement **as Compute Engine VMs that run within a MIGs**. They communicate with the master through an agent call kubelet

**Pods:**

* the smallest computation unit managed by Kubernetes. Pods contain generally 1 container (could be more if more than one operation like ETL)
* deployed to nodes by the scheduler in **groups of replicas**
* support **scalability**
* **ephemeral**. A service keeps tracks of its associate pods. If one is down -> running a new one

**Replicaset** is a controller that manages the number of pods running for a **deployment**

If **taints** is assigned to a pod -> you can control when a pod is launched

**Autoscaling** adjust the number of replicas -> a deployment can be configured to autoscale

Configuration can be done with command line, cloud console, REST API. You can specify parameters in yaml file.

*kubectl autoscale deployment myapp –min2 –max 3 –cpupercent 65*

Google recommend **deploying model** in production with **Cloud ML engine.**

**Kubernetes** is only used to deploy model if **you have already planned to work with containers**.

Command lines example:

**gcloud container commands are used for interacting with Kubernetes Engine**

To create a Kubernetes cluster from the command line

*gcloud container clusters create "standard-cluster-1" --zone "us-central1-a"*

*--cluster-version "1.13.11-gke.14" --machine-type "n1-standard-1"*

*--image-type "COS" --disk-type "pd-standard" --disk-size "100"*

*--num-nodes "5" --enable-autoupgrade --enable-autorepair* ***--enable-autoscaling***

**kubectl is used to control Kubernetes components**.

*kubectl scale deployment pde-example-application --replicas 6*

*kubectl autoscale deployment pde-example-application --min 2 --max 8 --cpu-percent 65*

**Compute Engine**

**Server-based**

Iaas (infrastructure as a service)

Configure VM instance

* Number of CPUs/GPUs
* Specify region zone
* …

Can be configured using instance group

Compute Engine support up to 8 GPUs in a single instance.

You can recreate instance directly from a snapshot without restoring to disk first.

Command lines example:

*gcloud compute instances create instance-1 --zone=us-central1-a*

*--machine-type=n1-standard-1 --subnet=default --network-tier=PREMIUM*

*--image=debian-9-stretch-v20191115 --image-project=debian-cloud*

*--boot-disk-size=10GB --boot-disk-type=pd-standard*

**GCP cloud Functions**

**Serverless**

Managed compute service for running code in **response to events that occur in the cloud.**

Expl -> cloud Pub/sub, file to Cloud Storage, HTTP, Firebase, Stackdriver logging…

Write it using Python3, Javascript and Go

**Useful to ingest data into Cloud Pub/sub such an IOT ingestion pipeline**

You can have option configuration policies for scaling and set maximum number of concurrently running instances.

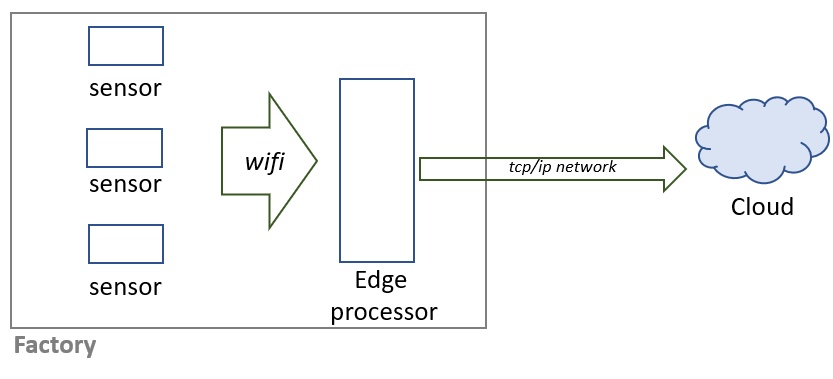
The amount of memory allocated to a function range **128 MB to 2 GB**

To avoid spike, configure –max instance when deploying the function

**Edge Computing**

Moving compute storage closer to the location at which it is needed.

Needed when low-latency data -> manufacturing equipment / autonomous cars



Basic components:

* Edge device
* Gateway (manage traffic access protocol)
* Cloud Platform GCP

Kind of data:

* Metadata (device ids …)
* State information (about device states)
* Telemetry (collected by devices)

For security -> authentication of device by token key or other mechanism device should be tracked in a central device registry.

In GCP service in use:

* Cloud Pub/Sub
* IOT ore MQTT Message Queue Telemetry Transfer
* Stackdriver Monitoring / logging

**AI Platform** is a manage to support deploying ML to the Edge

* Service prediction
* Deploy new models
* Manage API Keys

AI platform also expose Rest APIs -> online prediction

**Edge TPU** is ASIC (application specific integrated circuit) designed for running AI services at the Edge.

**Cloud IOT Core** is a managed service designed to manage connection to IOT devices

* Provide services for integrating edge computing with centralized processing services.

Device data is captured by cloud IOT core service before being published by Cloud Pub/sub

**Analytics**

**Bigquery**

**Serverless**

Fully managed, petabyte scale low-cost analytics data warehouse

Designed to support **data warehouse** and **datamart**

Max slots = 2000 = 100 GB -> more => flat-rate pricing (check with **stackdriver monitoring slots utilization** if more resources is needed (not the CPU)

Streaming:

* are available within **a few seconds for analysis but may be up to 90 min**.
* support deduplication -> insert by ID -> longer ingestion (disactivated deduplication for higher performance)

Cost:

* **Less than 90 days** stored data -> **archive** (0.2$/GB per month) **else long-term** (0.1$/GB per month)
* Streaming 0.01$ per 200 MB inserted
* Queries 5$ per TB scanned
* **Not pay scan** query -> use **bq head command line**. Because limit in query imply a scan for all data

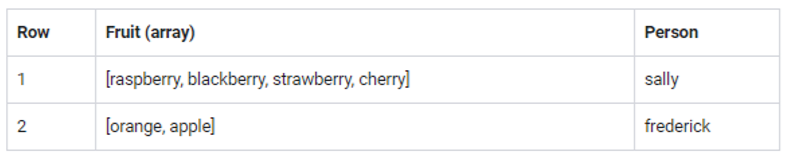
Table:

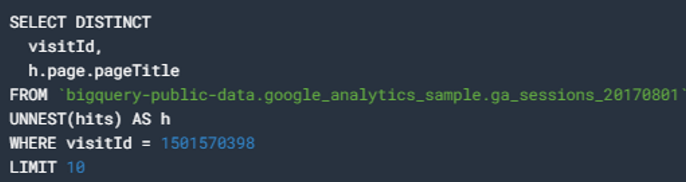
* **collection of rows** and **columns are stored** in columnar format known as **Capacitor format** in **colossus filesystem**
* When **coping** table -> **source and destination must be in the same location to use bq copy.**

**Else you need to use bigquery transfer service.**

*bq mk --transfer\_config --data\_source=cross\_region\_copy \  
--params='{"source\_dataset\_id": "iowa\_liquor\_sales", "source\_project\_id": "bigquery-public-data"}' \  
--target\_dataset=ch10eu --display\_name=liquor \  
--schedule\_end\_time="$(date -v +1H -u +%Y-%m-%dT%H:%M:%SZ)"*

* Higher performance: **Partition**: ingestion time, timestamp, integer / **Clustering**: works only for partitioned table
* Better than join -> **denormalization** with **nested and repeated columns**. **RECORD** type, use **STRUCT** SQL instruction. **Up to 15 levels of nested structs.** STRUCT is used to store ordered fields
* Data type **ARRAY** permit denormalization. Use **UNNEST()** on array field to query

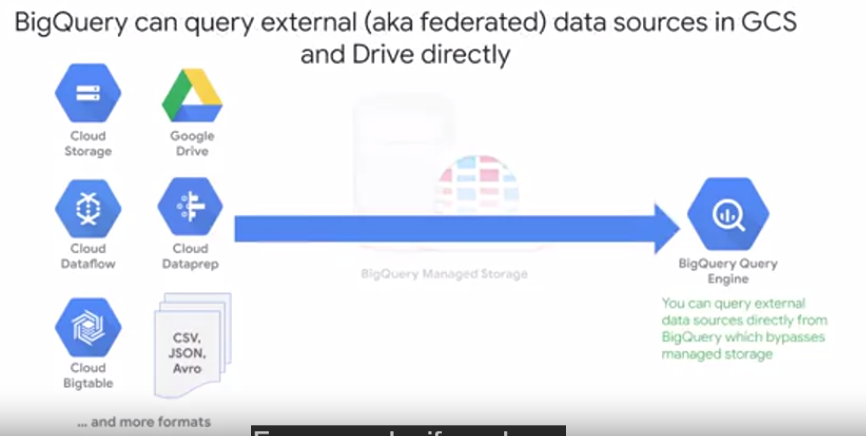




Queries:

* Bigquery is not relational database but SQL work
* Interactive queries (immediate, by default) / Batch queries (wait for resource available)
* Queries can run for **6 hours**, no longer
* Support parameters -> `*WHERE size = @param\_size*`
* **Wild card** can be used -> `*FROM myproject.mydataset.\** ` (different name in location => \* all table name) work only for tables, **not views**
* **Wildcard** are used only in **standard SQL**. **TABLE\_DATA\_RANGE()** can be used as wildcard in **legacy SQL**
* to avoid streaming duplicates : ROW\_NUMBER window function with PARTITION by unique ID along with WHERE row equals 1
* SELECT \* FROM `table`\* WHERE \_TABLE\_SUFFIX BETWEEN 3 AND 8
* to query a partionned table use \_**PARTITIONTIME** pseudo-column in WHERE close
* max 1000 tables in a query
* **Wildcards** or query with **destination** **do not support cache** (you have to pay each launch)

External data (**federated data source**) from (bigtable, cloud storage, google drive ….) (CSV, AVRO, JSON)

****

BigQuery does **not provide direct load from Cloud SQL**. The data needs to be loaded through Cloud Storage

**Stackdriver Logging support exporting logs to BigQuery by creating sinks**

**Bigquery Data Transfer Service import data from services** (Google Analytics,  Google Ads, Campaign Manager, Google Ad Manager and Youtube, Amazon S3, Teradata ) **NOT consider as federated data source**

Support UTF-8 (default). Also supports ISO- 8859-1 (need to be specify)

Schema:

* **can be changed** in a table **ONLY** to **add new column or relax a column required to nullable**.
* **cannot be** changed for others operations (delete, rename, modify data types….) you have to **create new table with new schema** and insert data from your first Bigquery table.
* can **automatically infer schema for CVS and JSON**

**External datasource** : useful to query it using a temporary table -> for **ad-hoc queries** (exemple **ETL processes**)

Permission:



Permission are in **dataset level** -> impossible in table level.

To share some result from a table you can use **authorized views** with dataViewer role.

Query cost:

* Evaluate cost -> view the query validator in the BigQuery GUI
* Evaluate cost -> use the --dry-run option with the bq query command.
* Limit cost -> set a maximum number of bytes billed for a query. in the GUI or in a bq query command using the --maximum\_bytes\_billed parameter.
* Limit cost -> use partitions

**Cloud Datalab**

Workbooks based on open source Jupyter notebook.

To start you can use development kit (SDK)

* datalab create –machine-type …. (--delete-disk to specify delete persistence instance)

use shell command in jupyter notebook to install packages (exp : ! pip install seaborn)

**Cloud Dataprep**

**Serverless**

Managed service to help reduce time to prepare data

* explore / clean / transform data
* Dataprep can be used to handle schema changes by Data Analysts without any programming knowledge
* Dataprep recipe can be export as as Cloud Dataflow template, and incorporate into a Cloud Composer job.

Can import all files type (CSV, JSON, TSV, Parquet, Bigqquery, Avro..)

Export Only CSV and JSON

**Serving**

**APP engine**

**Serverless**

Paas (Platform as a service) for running **application**

Languages : Go, Java, PHP, Node, JS, Python (NOT Ruby mentioned)

Instance classes run up to 2048 MB of memory / 4,8 Ghz CPU

* **App Engine Flexible**: can run in **Docker** containers
* **App Engine Standard**: when app is developed in one supported language and need scale up and down

Ensuring availability and scalability:

* With managed instance groups (**MIGs**) using a template
* When an instance in MIG failed -> replaced with an identically configured VM
* Global **load balancers** can be used to distribute workloads across region.
* **Autoscalers** add and remove instance according to workload

Configuration files:

* app yaml (configuration)
* cron yaml (cron)
* dispatch yaml (routing)

If working with **Bigtable** -> recommend in the **same zone**

**Cloud Data Studio**

Interactive reporting tool

3 kind of connector:

* Google connector -> access to other Google service (Bigquery…)
* Partner connector -> third parties (Facebook, Twitter, Github..)
* Community connector -> developed by anyone

Connections:

* Live connection -> automatically update with table changes (by default)
* Extracted data -> snapshot of data -> better performance
* Blended data -> combing from up to 5 data sources

Report can be available by ink sharing / PDF send by mail…

BigQuery queries are cached for 12 hours (no need to rerun a query) -> improve data studio performance. To get more than 12 hours cache you need to use **prefetch caching** and set up report to use **Owner’s** credentials.

**Monitoring and security**

**Stackdriver Monitoring**

Collect data more than 1000 metrics

**Stackdriver Logging**

Collect data from more than 150 applications

Retention:

* **Admin activity, system activity, access transparency logs -> 400 days retention**
* **Data access logs** -> **30 days retention**

**Stackdriver logging agent** **need to be installed in Compute engine instance** which have specific database like postgreSQL

**Stackdriver Trace**

How long it takes to process request, start job .. ?

* useful with compute engine, Kubernetes, App Engine

**Change data capture**

Save changes over time

**Data Catalog**

**Serverless** GCP metadata service for management data

* Collect automatically metadata from google services (Cloud Pub/sub, APIs, Bigquery, ….)
* Interesting to analyze table structures etc

**Identity and Access Management (Cloud IAM) and others**

Google cloud is fine grained identity and access management service that is to control which user can perform operation and resource within GCP

Roles:

* **Primitive roles** -> apply at the project level (Owner, Editor, Viewer) use it only in grained course level (expl developers will be responsible in dev environment)
* **Predefined roles** -> associated with a GCP service (APP Engine, Bigquery…)

*roles/appengine.application.get*

*roles/appengine.operation.\* (\* => all operations available)*

*roles/bigquery.dataset.create*

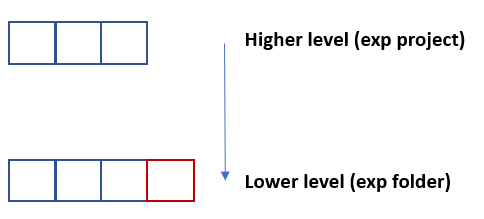
* **Custom roles** -> assign one or more permission to a role and then assign role to a **user** (email) or **service account** (email - expl for App Engine : <projectId>@appspot-gserviceaccount.com), authentication by a pair of public / private key

Access control with policies:



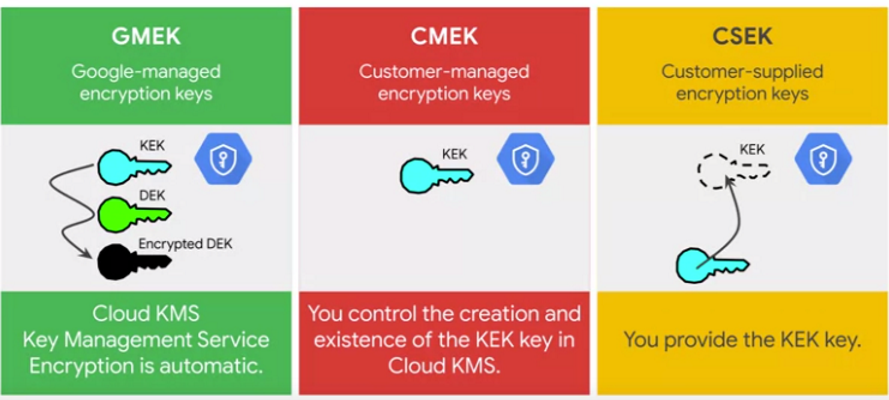
It is possible to audit policies configuration

IAM is additive only



Encryption:

* Data is encrypted with data encryption Key (**DEK**), and encrypt second time with second key – **envelop encryption (KEK)** (exp : each chunk is encrypted with identifier referenced by access control list ACL)
* **Default Key Management (GMEK):** Google manage encryption keys for users
* **Customer Managed encryption Key (CMEK):** use KMS a hosted key management service in Google Cloud to generate keys
* **Customer Supplied Encryption Key (CSEK):** supplied key from customer key system
* Data in transit is encrypted



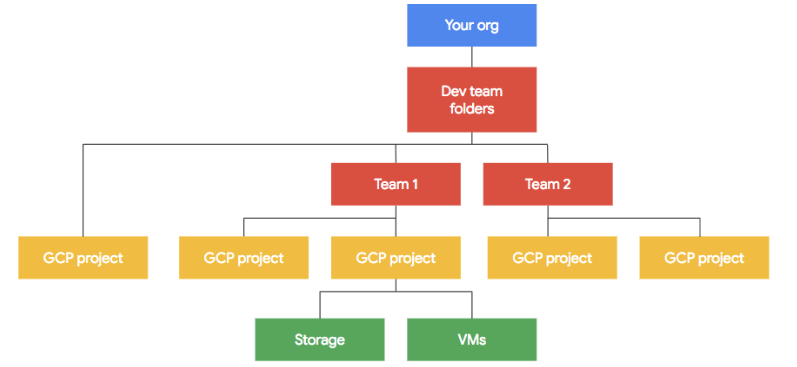
**Data Loss Prevention API** detect sensible data (credit cards..) -> can run job to detect Infotype and produce reports about utilization.

**HIPPA** -> Health Insurance Portability Accountability – **federal law in the US protect healthcare data**

**COPPA** ->children’s Online Privacy Protection Act – **US federal law children privacy**

**FedRAMP** ->federal risk and authorization management program – **federal government program**

**GDPR** ->general data protection regulation – **European private data of EU citizens**



Best practice is to **isolate the environments (dev, test, prod) in dedicated projects** to control billing and access control.. Migrate data when it is ready to deploy to the next stage

You can **automate project creation** with **Cloud Deployment Manager**. Where you define parameterized templates that are reusable building blocks. It also can access control permission set control through IAM

**AI Services**

**Kubeflow**

Open source project

* develop orchestra, deploy scalable and portable ML workloads
* designed for the Kubernetes platform
* can be used to run ML workloads in multiple cloud environments
* pipelines are packaged **as Docker images**
* good choice if you are already working with **Kubernetes engine**

**Cloud Machine Learning**

**AutoML**

Does not take in consideration train/test. Put in auto ML only train dataset.

**Vision AI**

Analyze **images** and identify text using OCR (explicit image)

Call vision API with client library (Python, C# Go Java node-js PHP Ruby), REST, gRCP

Image are sent by URI path or by sending Base-64 encoded text

Detecting:

* Text in images
* Text in PDF
* Faces
* Logos
* Objects
* Landmarks
* …

Also provide batch processing

Multiple features can be detected in 1 call



**Video AI**

Annotate **Videos** content, extract metadata

Detecting:

* Object location, animals, products…
* Explicit content
* Text
* Transcribe videos (30 alternative translation for a word)

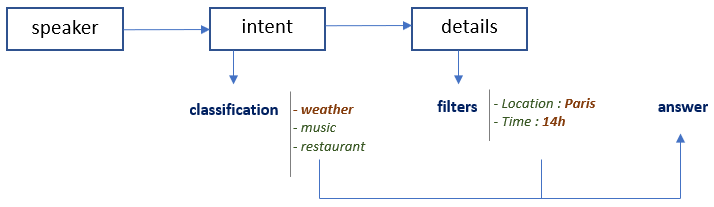
Call video AI API with client library, REST, gRCP

Image are sent by URI path or by sending Base-64 encoded text

Support MOV, MP4 AVI, MPEG4

**Dialogflow**

Used for **chatbot** interactive voice.



Call Dialogflow API with client library, REST, gRCP

**Cloud Text-to-Speech API**

Translate **Text** to speech

* More than 30 languages
* Based on speech synthesis technology Wavenet (Deepmind)

Use Base-64 encoded string

Call API with client library, REST, gRCP

**Cloud Speech -to-Text API**

Translate **Speech (audio)** to text

* 120 languages
* Deep learning technology
* Recommandation : audio should be sampling **16 Khz or higher** / use codec (FLAC, LINEAR16). **Native sample rate is recommended** over resampling if speech dataset is already build (even if 8 Khz).
* Synchronous mode is recommended for short audio files

Call API with client library, REST, gRCP

**Translation API**

Translate **Text** and html

* More than 100 languages

Text need to be encoded in UTF-8

Call API with client library, REST, gRCP

**Natural language**

Analyze **Text** as classification

* Extract information about people, places, events…
* 700 general categories (sport, painting, IT,…)

Analyze:

* Sentiment (general, by each person)
* Entities
* Context
* …

Document can be up to **10 MB**

**Recommendation API**

Ingest catalogItems / tags **userEvents from Google Analytics, Tealium**…

* Give : “other you may like”, “frequently bought together”, “reco for you”, “recently viewed”..
* Metrics clicks, conversion rate…….

Metric guide:

Different metric to assess the different targets

“other you may like” : **likely to purchase**

* **Conversion rate** metric

“frequently bought together” : often purchased in the **same session**

* **Revenue per order** metric

“Recommend for you” : **likely to engage**

* **Click-through-rate (CTR)** metric

**Cloud inference API**

Real time analysis of **time-series**

Exemple : telemetry from IOT

**Glossary**

**Data warehouse**

Centralized, organized repositories for analytical data for an organization

**Datamarts**

Subset of Data Warehouse that focus on business lines departments

**Data Lakes**

Less structured data stores

**CPU (central processing units)**

Recommend for **simple model, C++ operation, limited available I/O models**

**GPU (graphic processing units)**

Accelerator that have multiple logic units (ALUs)

* Implements adders and multipliers
* Benefit from massive parallelization

Intermediate calculation results use registers or shared memory -> lead to Neumann bottleneck

Recommend for **tensorflow, code difficult to change, medium and large models**

**TPU (tensor processing units)**

Accelerators based on ASICs created by Google. Ddesign for tensorflow framework (available only in GCP).

* 1 TPU = 27 \* 8 GPUs (result in one benchmark made with specific conditions)
* Scale horizontally using TPUs pods
* Reduce Neumann bottleneck
* Better than GPU for large scale in deep learning
* Used with Compute Engine, Kubernetes, AI engine

Recommend for **matrix with multiplication models, tensorflow, models take weeks to train on CPUs/GPUs, very large models.**

**NOT** Recommend for **high precision arithmetic**



**Tensorflow**

Support both synchronous and asynchronous training.

You can use container to generate models that are trained with different hyperparameters -> in parallel with GKE managing deployments of pods.

**VM preemptible**

VM that can be killed by GCP in any moment for any reason (maintenance, ressources etc)

* Deploy not important jobs / jobs that can be crashed without impact
* Can be interesting -> economic solution
* No tolerance contract with GCP
* **Cannot** **store** data

**AVRO**

Serialization and de-serialization of data so that it can be transmitted, stored while maintaining structure object.

Avro binary format is the preferred format for loading compressed data. Avro data is faster to load because the data can be read in parallel, even when the data blocks are compressed. Compressed Avro files are not supported, but compressed data blocks are

**MIG**

Gcloud compute instance templates -> managed instance group

**OLTP**

Online **transactional processing**

Transaction type SQL (primary/secondary key)

**OLAP**

Online **analytical processing**

For data warehouse or datamarts (exemple looking in different direction / dicing)

**Hybrid Cloud**

Analytics hybrid cloud is used when transaction system continues to run on premises and data is extracted and transferred to the **cloud for analytics process.**

**Edge Cloud**

A variation of Hybrid cloud which use local computation resources in addition to cloud platform.

Used **when network is not reliable**

**SOA**

Service oriented Architectures -> driven by business operations and delivering business value

**Microservices**

Distributed architectures designed to implement a single function. This allow services to be updated independently. Can be deployed in a container

**Serverless function**

Extend principles of microservices by removing concerns for containers and management runtime environments.

Can be deployed on a Paas as Google Cloud Function without having to configure a container.

**Availability**

Measured as a percentage of time a system is operational.

**Reliability**

Measured as then mean of time between failures.

**Extract, Transform, Load (ETL)**

Pipeline begins with extraction.

Transformation can be done using Cloud Dataproc or Cloud Dataflow

**Extract, Load, Transform (ELT)**

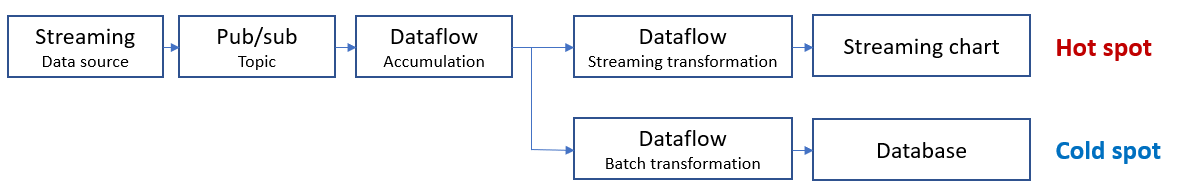
Load into a database before transform

* developers can query data to make data quality analysis, perform specific transformation

**Extract, Load (EL)**

When data does not require transformation

**Streaming**



* **Hot spot** -> as soon as possiblr ingestion (expl : online retailer)
* **Cold spot** -> if need to save data even if not available on time for hot spot
* **Event time** -> time data is generated
* **Processing time** -> time data arrive at endpoint

**SSH Tunnel**

A [Secure Shell](https://en.wikipedia.org/wiki/Secure_Shell) (SSH) tunnel consists of an encrypted tunnel created through an [SSH protocol](https://en.wikipedia.org/wiki/Secure_Shell) connection. Users may set up SSH tunnels to transfer [unencrypted](https://en.wikipedia.org/wiki/Unencrypted) traffic over a network through an [encrypted](https://en.wikipedia.org/wiki/Encrypted) channel. It is a software-based approach to network security and the result is transparent encryption.

**SLA**

Partner interconnect is useful for data > 10 Gbps

**Dedicated Interconnects**

Requirements are a maximum of 20 Gbps of data and a Service Level Agreement (SLA) of 99%.

**Aggregated sinks**

An aggregated sink that can export **log entries from all the projects, folders, and billing accounts of a Google Cloud organization**. To use the aggregated sink feature, create a sink in a Google Cloud organization or folder and set the sink's ***includeChildren*** parameter to ***True***.

**Supported destination:**

* Cloud storage
* Pub/sub
* Bigquery
* Cloud logging bucket

**MQTT (**[**Message Queuing Telemetry Transport**](https://en.wikipedia.org/wiki/MQTT)**)**

Publisher subscriber pattern, in which clients connect to a broker and the remote devices publish messages to a shared queue. The protocol optimizes towards message size, for efficiency.

[**Google Cloud IoT Core**](https://cloud.google.com/iot-core/) currently supports **device to cloud communication through two protocols: HTTP and MQTT**. The real advantage of MQTT over HTTP occurs when we reuse the single connection for sending multiple messages in which the average response per message converges to around 40 ms and the data amount per message converges to around 400 bytes.