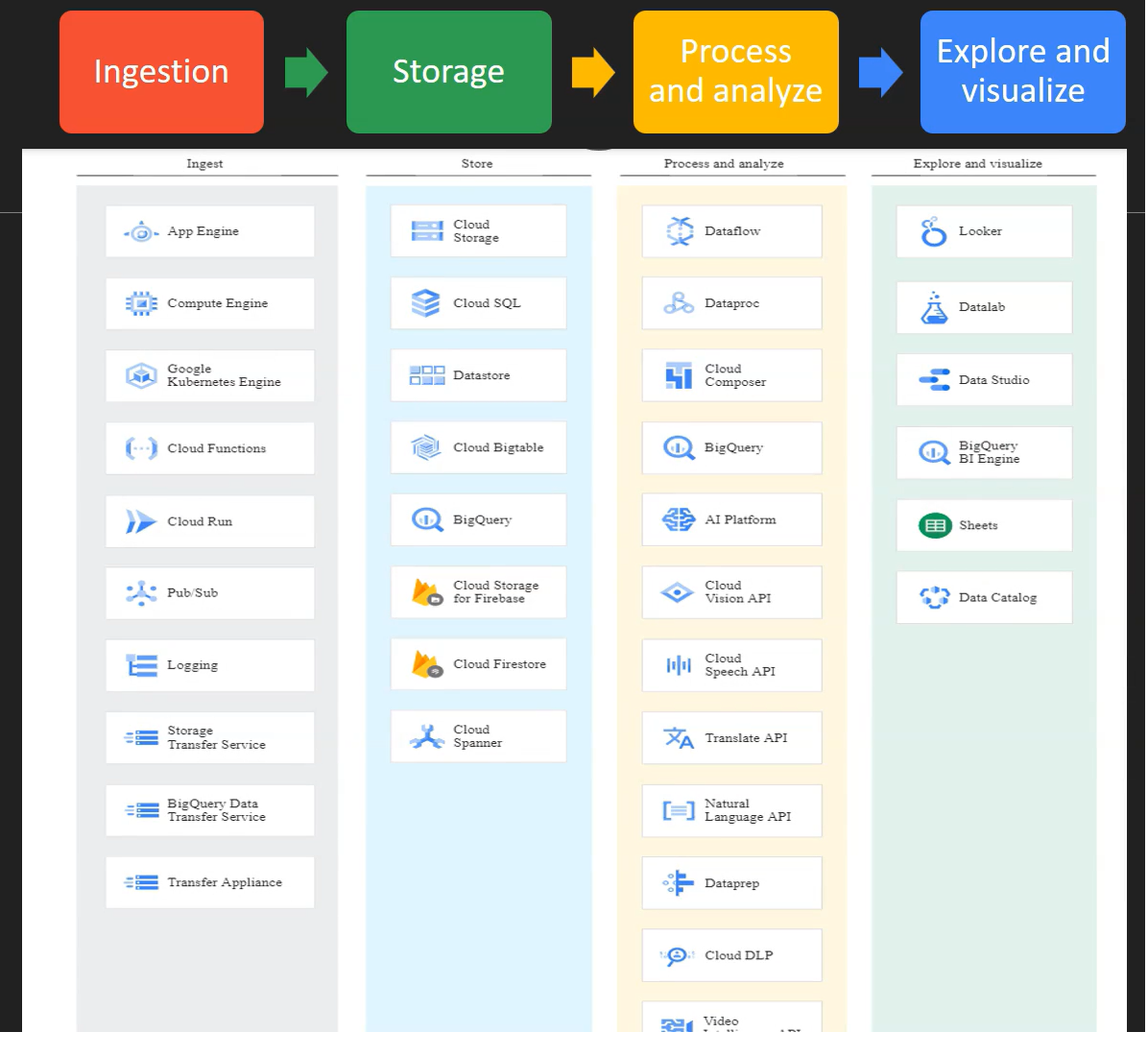
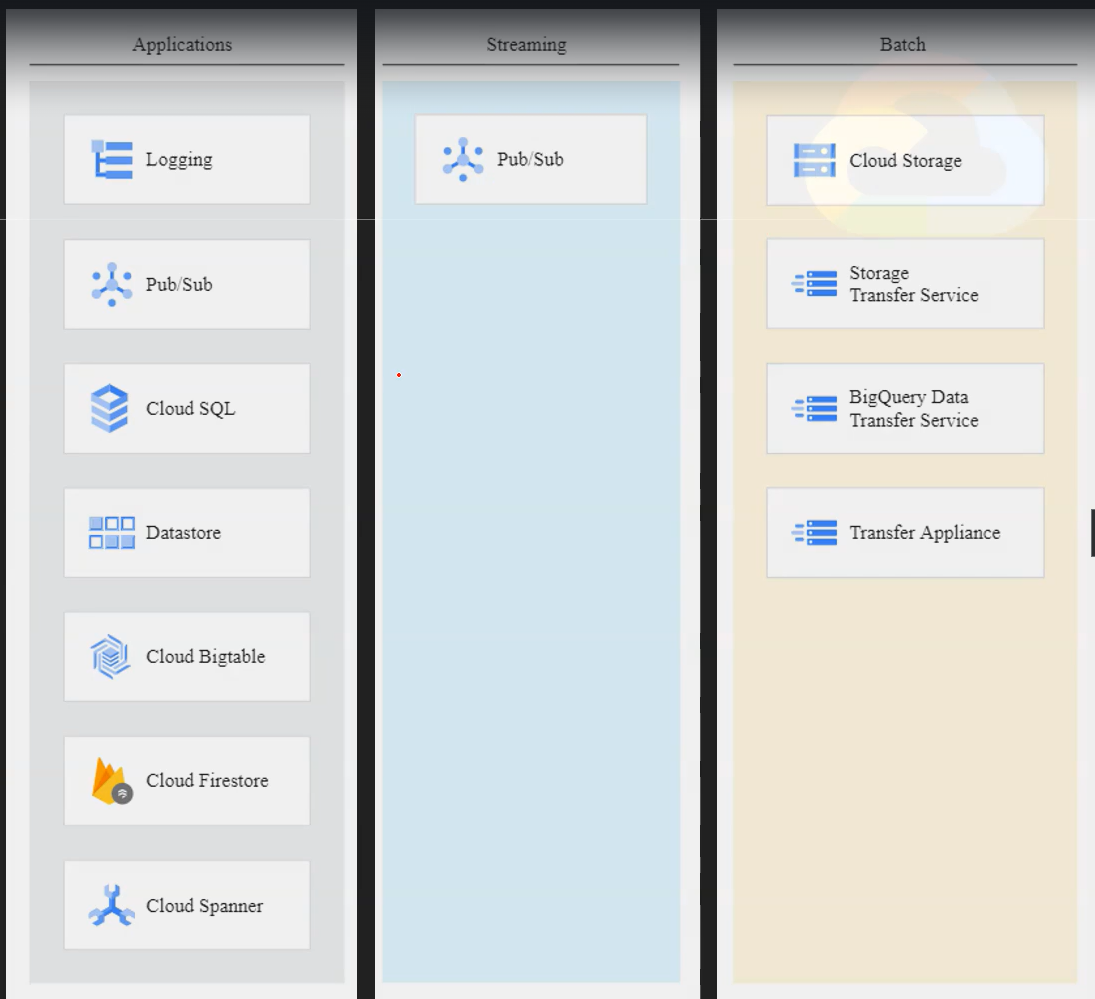
A Data Engineer designs and builds systems that collect and transform the data used to inform business decisions.

Basic steps and involved services in data engineering:

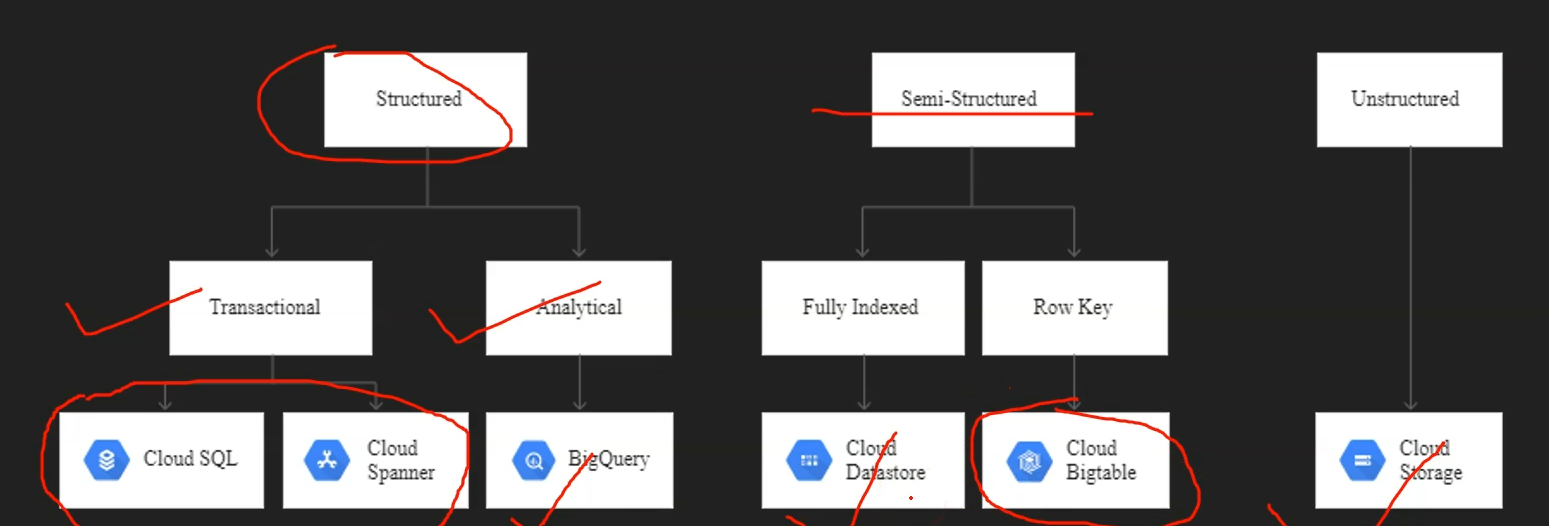


Injection: Gathering data from multiple sources. And storing the data cost effectively and durably.

Based on nature of the data, following are the list of cloud services involved:



Based on variety of the data, following are the services:



Structured Data:

* Rows and Columns
* Fixed schema and each row have same number of columns.
* MS SQL Server, MySQL, Postgres

Semi-structured data:

* Entities/Records and Properties
* No fixed schema and record have variable number of properties.
* Stored as key-value pairs with JSON.
* MongoDB, Cassandra, Redis, Neo4J
* Big Table, Cloud Datastore and Memory Store

Un-structured data:

* No predefined structure in data
* Ex: Video, Image, Natural language etc.
* Cloud Storage and File Store

Batch Data Processing:

* Defined start and end so size is known.
* Process high volumes of data at periodic intervals and takes long time to process
* Ex: Payment processing

Streaming Data Processing:

* No end is defined, and it is unbounded so size is unknown
* Process as it arrives and takes milliseconds to seconds
* Ex: Stock data processing

Regions and Zones:

* Zone – Independent data center
* Region – Geographical area
* Low latency
* High availability
* Follow government regulations
* Disaster recovery

Infrastructure Services:

IAM

* Identity and access management
* WHO can do WHAT on WHICH resources
  + WHO – Identity: Google account, google groups, cloud identity, workspace account, service account:
    - Google account – Normal g-mail account
    - Google groups – Group of users
    - Cloud identity – Organization level
    - Workspace account – Organization level and like above except we can get all apps like g-mail, video calling, drive etc.
    - Service account – When a resource wants to interact with another resource, we use service account to assign roles.
  + WHAT – Action: Create, Update, Delete
  + WHICH – Resource: Compute Engine, App Engine, Kubernetes, Cloud Storage etc.
* Roles – Collection of permissions. We can assign role to identity but not permission directly.
  + Primitive – Owner, Editor, Viewer
  + Pre-defined – Storage Viewer, Compute Admin etc.
  + Custom – Our custom collection of permissions
* While adding user, we can add condition so that user/roles will expiry.
* Service account – For non-human like apps and services
  + It is identity for compute engine
  + It uses keys for authentication. Max 100 services accounts per project and max 10 keys per service account.

Compute Engine - IAAS

App Engine – PAAS

* Serverless, fully managed service
* Can deploy HTTP based web application at high scale
* Two flavors:
  + Standard – Support few runtimes
  + Flexible – Support all runtimes, docker based
* App Engine 🡪 Service 🡪 Version 🡪 Instance
* Auto scaling, Load balancing, Versioning, Traffic splitting

Kubernetes (GKE)

* If 100s of containers are required to scale our app, GKE is the solution which fully manage lifecycle of containers.
* Orchestration system for containerized application and open source

Cloud Storage

* Used to store unstructured data
* No capacity planning is required and scale to exabytes
* Can be used for long term archival storage
* Data is encrypted; both data at rest and in-transit
* High durability i.e., 11 9s
* It is a object storage solution
  + No minimum object size; 5TB is maximum object size
  + Objects are stored in buckets - Bucket name should be unique
  + URL – [https://google.cloud.com/[bucket-name]/[object-name](https://google.cloud.com/%5bbucket-name%5d/%5bobject-name)]
  + Storage 🡪 Bucket 🡪 Folder 🡪 Object
  + Objects are immutable and can be versioned
* Storage location
  + Regional – Lowest latency; Availability
  + Duel-regional – Low latency; High availability
  + Multi regional – Latency; Highest availability
* Storage Class – How frequently and how much data accessing
  + Standard – very frequently; Hot data i.e., best for short-term
  + Nearline – Once in 30 days; Backup
  + Coldline – Once in 90 days; best for disaster recovery
  + Archive – Once in Year; Offline data and backup i.e., best for long-term
    - Storage cost decreases and access cost increase from standard to archive
* Object life cycle management – Based on condition what action to be performed
  + Condition – Object age, no. of versions, after specific date etc.
  + Action – to Archive, to Coldline, Delete object etc.
* Encryption
  + Google Managed Encryption Keys
    - No configuration and fully managed
  + Customer Managed Encryption Keys
    - Create Key and Key Ring in KMS
    - Key management by customer like key rotation
  + Customer Supplied Encryption Keys
    - Create key using openssl rand -base64 32
    - Encrypt data manually prior to transfer using gsutil
      * gsutil -o “GSUtil:encryption\_key=<key>” cp 1.txt gs://mybucket/
* Versioning – Help to prevent accidental deletion of objects
  + Enable/disable versioning at bucket level
    - gsutil versioning set on <bucketname>
    - gsutil versioning get <bucketname>
  + Object key + version number 🡪 older file
    - Don’t need earlier version, delete for cost efficient
  + Only object key 🡪 latest file
  + List of all files in a bucket along with all versions
    - gsutil ls -a gs://mybucket
* Controlling access - Who can do what and which level
  + Uniform level access
  + Fine grained