

let's  
start  
something  
**brillio**

**GCP - Data Lake  
Design Play Book**

30/01/2022

Version 1.0



## Contents

1	Introduction .....	4
1.1	Document Objective .....	4
1.2	Document Scope .....	4
1.3	Audience .....	4
2	Data Lake - Definition .....	4
2.1	Building Phases of Data Lake .....	4
3	Data Lake – Reference Architecture .....	5
3.1	Solution Design .....	5
3.2	Migration Strategy and Path .....	7
3.3	Moving data into Google cloud .....	8
3.4	Define Project folder structure .....	8
3.5	Data Ingestion Layer .....	9
3.5.1	Cloud Storage .....	9
3.5.2	Cloud Pubsub .....	11
3.5.3	Cloud Run .....	11
3.5.4	IoT Core .....	11
3.6	Data Transformation and Processing Layer .....	11
3.6.1	Cloud Dataflow .....	11
3.6.2	Google Big Query .....	12
3.6.3	Partitions and Clustering .....	15
3.6.4	Automatic re-clustering .....	16
3.6.5	Denormalize tables with nested and repeated fields .....	16
3.6.6	Materialized view .....	17
3.6.7	User Defined Functions .....	17
3.6.8	Separate projects for consumption billing .....	17
3.6.9	Big Query ML .....	17
3.7	Data Services Layer .....	18
3.7.1	Cloud Run .....	18
3.7.2	Cloud Endpoint .....	18
3.8	Frameworks and Accelerators .....	18
3.9	Data Orchestration .....	18
3.10	Data Security .....	19

3.11	Data Compliance .....	21
3.12	Code deployment .....	21
3.12.1	Automated Code deployment.....	21
3.12.2	Infrastructure automation .....	22
3.13	Monitoring and Troubleshooting .....	22
3.13.1	Metrics.....	23
4	Appendix.....	24
5	Glossary .....	24
6	Document References .....	24

# 1 Introduction

## 1.1 Document Objective

The purpose of this document is to provide design guidelines to setup data lake in Google Cloud platform. This document will also provide best practices and recommendations to implement various data lake components.

## 1.2 Document Scope

This document covers the most recommended Google Cloud services (available at present) to setup a data lake and the High-Level Design guidelines to setup data lake using GCP services. This will not contain low level details such as scripts and configurations steps. It is recommended to also refer google release notes for latest product update and changes.

## 1.3 Audience

This document is intended for Data Engineering team (data engineers, leads, etc.,) who can build a data lake by following the recommendations. These recommendations and best practices gained from past implementations which can be customized/replaced based on actual requirements.

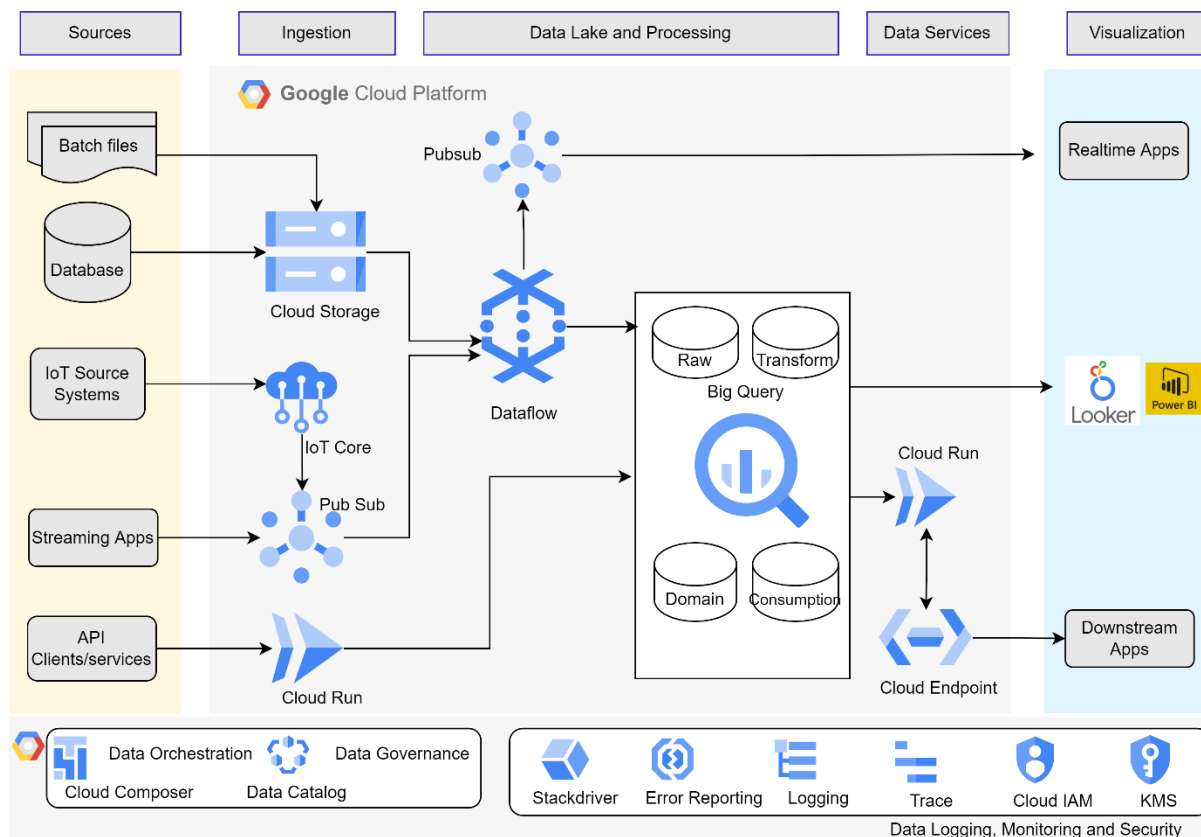
# 2 Data Lake - Definition

Data lake is a centralized repository of raw data from various data sources which can be structured, semi-structured or unstructured. This document covered with design approaches to setup data lake on Google Big Query.

## 2.1 Building Phases of Data Lake

- a) Identify various sources of data and format to be consolidated
- b) Assess various phases involved in moving data into cloud
- c) Decide suitable cloud services to design the data pipelines.
- d) Establish connectivity from data sources to cloud data lake.
- e) Define identity access and data security
- f) Define list of metrics to be monitored
- g) Define Data governance and compliance policies

### 3 Data Lake – Reference Architecture



Data Lake on Google Big Query

#### 3.1 Solution Design

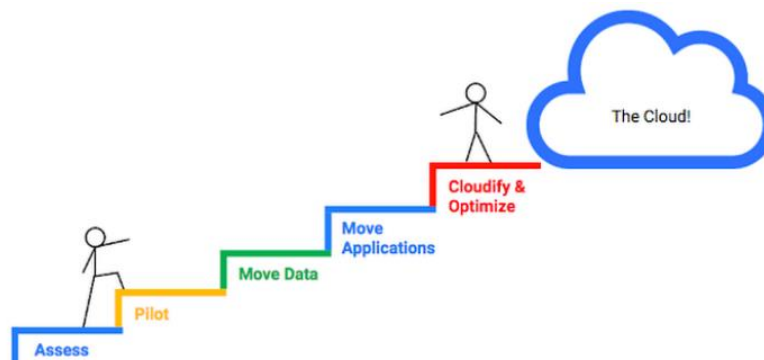
The Data Lake Platform would primarily comprise of the following components:

Component	Description
Source Systems	Various data sources and type ex:- batch files, Database(on-prem/cloud), IoT, Streaming Apps, API services etc.,
Ingestion Services	Data Ingestion is the process of moving source datasets into raw layer of data lake in as-is format. Depending on the nature of data source equivalent cloud services can be chosen as shown in the above diagram.
Processing Layer	Data Processing involves cleansing, de-duplicating, curating raw data into business required format.
Visualization Layer	Data Visualization Layer contains dashboard and analytical

	reports for various business requirement. BI tools ( ex:- power BI, Looker) ,Realtime apps, Downstream applications consumes the transformed and aggregated data for various business use cases.
Data Catalog Management	Data Catalog service can be used for data discovery and meta data management. Automatically catalog assets from Big Query, cloud storage, pub sub and data proc meta store. Policy tags can be created and assigned to PII columns of Big Query columns for column level security.
Data Security	Data Security and Compliance would be established by defining the roles, policies and specific data protection requirements. Other security features like Cloud KMS, data encryption at REST and VPC service controls can also be used.
Data Governance	Data Governance process starts from the data onboarding step till final layer of a system. Classifying, Tagging, Defining Taxonomy, Tokenizing / De-Tokenizing sensitive data like PII (Personal Identifiable Information), Data sharing are some of the important areas to be taken care.  Data Compliances can also be applied: -  <i>HIPAA</i> - Health Insurance Portability and Accountability Act <i>CCPA</i> - California Consumer Privacy Act – USA Region <i>GDPR</i> - General Data Protection Regulation – Europe region
Continuous Integration, Continuous Development	A DevOps pipeline can be created through cloud build triggers based on version-control commits to automate deployment process.
Data Orchestration	Data Lake ingestion process can be orchestrated through cloud composer service.
Monitoring Operations	Monitoring and troubleshooting of a data lake can be achieved through various services offered in Google cloud offers Operations Suite to raise alerts, autoscaling etc.,

### 3.2 Migration Strategy and Path

Google recommends following phases for cloud migration.



Reference: Google Cloud Documentation

Phase	Description
Assess	<ul style="list-style-type: none"> <li>a) Assess the feasibility of moving existing applications to cloud</li> <li>b) Identify suitable cloud services, hardware and performance requirements</li> <li>c) Estimate the cost for the usage, licensing etc.,</li> </ul>
Pilot	<ul style="list-style-type: none"> <li>a) Learn cloud pattern and design patterns</li> <li>b) Try moving one application to cloud and perform PoC to evaluate the performance</li> </ul>
Move data	<ul style="list-style-type: none"> <li>a) Identify dependencies of existing applications and decide type of cloud storage</li> <li>b) Move historical data into cloud. Please refer section 3.3 detailed with transfer services.</li> </ul>
Move applications	<ul style="list-style-type: none"> <li>a) Do the minimum necessary to move applications to cloud.</li> <li>b) Types of migrations are:- <ul style="list-style-type: none"> <li>i. <b>Lift and Shift</b> – No code refactoring. Requires less time for migration.</li> <li>ii. <b>Improve and replace</b> – Refactor existing code to take cloud benefits</li> <li>iii. <b>Rip and replace</b> – Decommission existing service and Redesign.</li> </ul> </li> </ul>
Optimize	Once application migrated to GCP, improvements can be done on performance, scalability, high availability by monitoring using operation suite offered by google.

### 3.3 Moving data into Google cloud

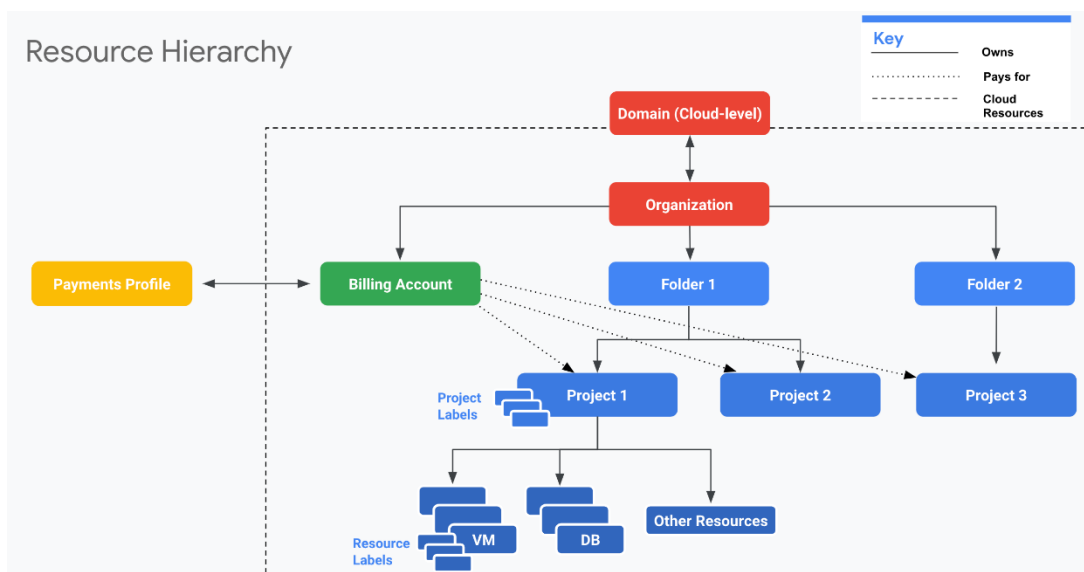
Google cloud provides the following transfer services to move data from on-prem or other clouds.

Transfer Type	Transfer Service	When to use
<i>On-prem to Cloud</i> (Peta bytes transfers)	Transfer appliance	Moving <u>offline</u> data, large datasets, or data from an on-prem source with limited bandwidth. Suitable for the transfers which would take more than one week to upload your data over the network (> 10 TB). Appliance shipped to the nearest google location and then uploaded to cloud storage.
<i>Other Clouds to Google Cloud</i>	Storage Transfer Service	To move data to google cloud storage from other clouds. Recurring transfer can be scheduled
<i>Other google or SaaS application to Google Cloud</i>	Big Query Data Transfer Service	To move data from software as a service (SaaS i.e youtube, ads360, google ads, salesforce, terradata, Amazon redshift) applications to BigQuery. (thru console, bq or BQ Data Transfer API ). This transfer process can be scheduled for a periodic data transfer.
<i>On-Prem to Google Cloud</i> (Tera bytes transfers)	Transfer Service for <b>on-premises</b> data	Transfer data from on-prem to cloud storage through transfer agent(docker based) installed in on-prem vm's mounted using NFS volumes.  gsutil supports transfer size < 1 TB.

### 3.4 Define Project folder structure

In google cloud resources are organized hierarchically. This hierarchy allows you to map your organization's operational structure to Google Cloud, and to manage access control and permissions for groups of related resources





Reference: Google Cloud Documentation

Resource Hierarchy	Description
Domain	Manage the users in your organization and is directly related to the organization resource.
Organization	Top level node. Represents entire company
Folders	Isolate different departments into folders
Projects	Bottom level node. Contains resources (compute,storage, etc.,)
Resources Labels	Resources can be identified using labels as a group.
Billing account	Billing account is connected to one or more projects which can be in same of different folders. Usage summary and cost details can be referred here
Payment Profile	Billing account is connected to payment profile which is attached to payment method

## 3.5 Data Ingestion Layer

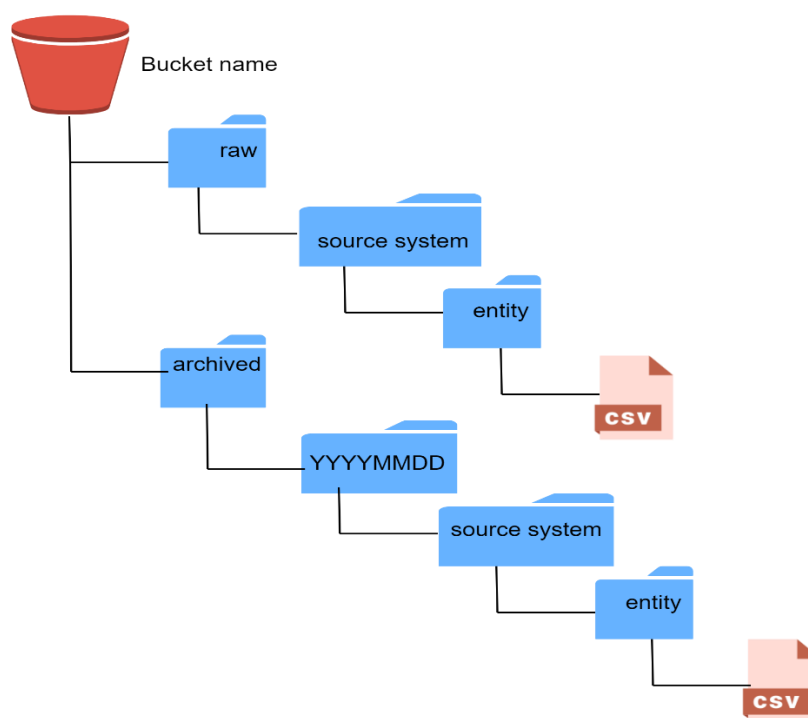
### 3.5.1 Cloud Storage

Purpose description	Recommendations
1) To store source system input files, database extracts, Historical migration data, and archive of processed data	a) Define folder structure as shown below diagram to organize source system data to manage raw, reject and archival data
2) Recommended storage to bring <b>unstructured</b>	b) Define life cycle management policies at bucket level to move file objects which are not frequently used to appropriate storage classes. This helps in saving storage cost.

<p><b>data</b> into data lake. Ex:- video / audio files, images, machine generated log files etc.,</p>	<p>Supported storage classes and example rules are : -</p> <ul style="list-style-type: none"> <li>• <b>Standard</b> – Up to 30 days older files which are frequently accessed data.</li> <li>• <b>Nearline</b> – 30 to 60 days older files which are accessed less than once a month.</li> <li>• <b>Coldline</b> – 60 to 180 days older files which are accessed less than once a quarter</li> <li>• <b>Archive</b> – files older than 180 days which are accessed less than once a year.</li> <li>• <b>Delete Object</b> - You can also choose a rule to delete older files which are no longer required.</li> </ul> <p>c) Define necessary IAM role and permissions to respective service accounts and account users.</p> <p>d) You can also define bucket level access policies (ACL) to enforce granular level access permissions.</p>
--	--

### 3.5.1.1 Cloud storage template

Cloud storage bucket and folders can be setup for data lake files as shown in the below template: -



### 3.5.2 Cloud Pubsub

Purpose description	Recommendations
Highly available, Scalable, Integrates various services in real time through message streaming	a) Use Pull subscription when processing large volume of messages per second. Push subscription when HTTPS endpoint receives a message. b) Create dead letter topics to handle failure messages c) Define appropriate acknowledge deadline to avoid duplicates and failure retries.

### 3.5.3 Cloud Run

Purpose description	Recommendations
Severless computing , containerized applications. Most suitable for stateless HTTP requests.	a) Set concurrency to 1 if parallel processing is not possible. By default, container instance can receive up to 80 requests at a time b) Define appropriate memory size (twice on no. of CPU's in GB's) c) Set appropriate min-instance to avoid cold start delays. Remember this will incur cost even when no serving request

### 3.5.4 IoT Core

Purpose description	Recommendations
Securely connect, manage, and ingest data from millions of globally dispersed devices.	a) Paired with pubsub and data flow to ingest and process streaming data in real-time.

## 3.6 Data Transformation and Processing Layer

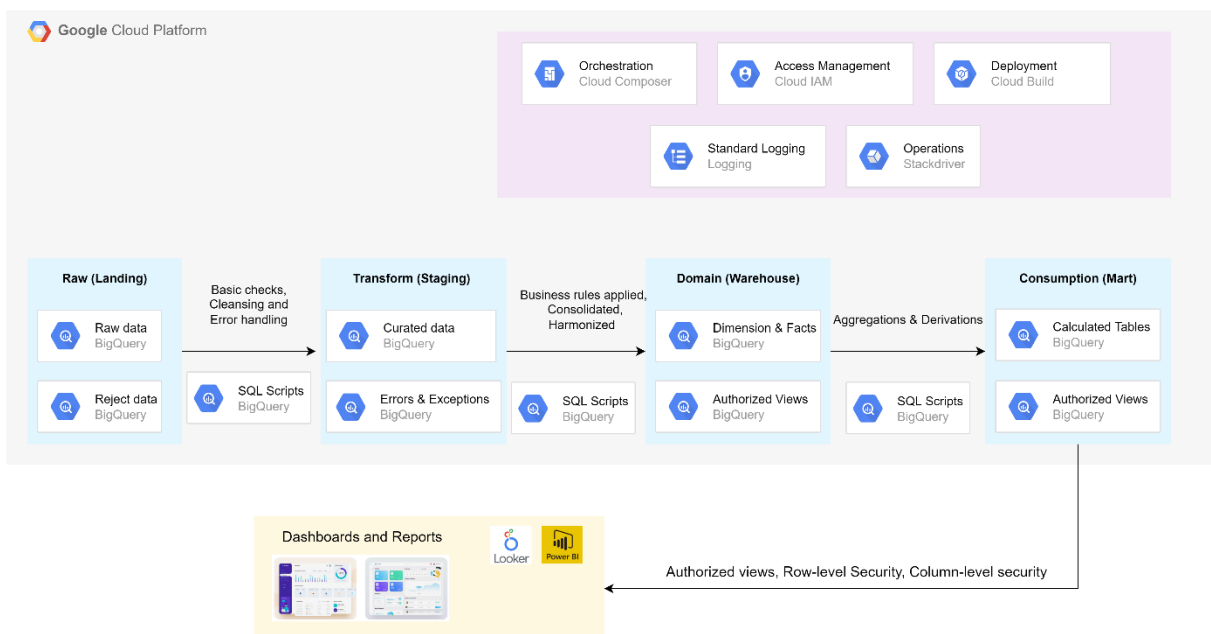
### 3.6.1 Cloud Dataflow

Purpose description	Recommendations
Unified data processing layer to process batch and streaming data from	a) Develop reusable functions to perform DQ transformations and database operations of data pipelines.

various data sources.	<p>b) Define appropriate scaling policy to manage minimum and maximum number of workers. Default number of workers are set to 3.</p> <p>c) Develop DoFn as idempotent to avoid impact due to retry failures.</p> <p>d) Provide appropriate display names for each pipeline branches</p>
-----------------------	---

### 3.6.2 Google Big Query

Big Query datasets can be created as shown in the following diagram to stage, curate, consolidate and transform source data.



Big Query Datasets	Description
Raw Layer	<p>Stores input data in source provided format. Google dataflow, cloud run services mentioned in the reference architecture diagram which reads data from various sources and ingests into raw layer in Big Query.</p> <p><b>Recommendations: -</b></p> <ul style="list-style-type: none"> <li>a) Create Ingest time partitions to store date wise snapshot of source data.</li> <li>b) Set partition_expiration_days if source data no longer</li> </ul>

	<p>required after specific period. (ex: &gt; 15 days)</p> <p>c) Design pipeline to move older partition to cold storage for storage cost benefits (ex: &gt; 15 days)</p> <p>d) Define source columns datatype as string to avoid data loss during ingestion</p>
Transform Layer	<p>Basic data validation checks (deduplication, cleansing, formatting, error handling) performed on raw data.</p> <p><b><u>Recommendations:</u></b> -</p> <p>Develop big query sql and execute through <i>bq query</i> utility. This utility can also be invoked from orchestration tool such as cloud composer.</p> <p>a) Identify the partitions to be processed and de-duplicate using row_number() function.</p> <p>b) Validate source data and convert source data from string into appropriate datatypes.</p> <p>Example:</p> <p>i) Parsing Date/Timestamp values with appropriate format YYYY-MM-DD HH:MI:SS</p> <p>ii) Number datatype with required precisions</p> <p>c) Format datetime to required format, add default values to source fields in case of blank/null</p>
Domain Layer	<p>Create required dimensions and fact tables to ingest transformed data. This will be the central layer enable us to serve various business requirements</p> <p><b><u>Recommendations:</u></b> -</p> <p>Develop big query sql and execute through bq query utility. This utility can also be invoked from orchestration tool such as cloud composer.</p>

	<ul style="list-style-type: none"> <li>a) Partition the table based on appropriate columns which provides amazing cost benefits. (refer <a href="#">section 3.6.3</a>)</li> <li>b) Perform clustering on the filtering columns which are required in addition to partition column. Defining order of columns are very important which decides the storage. (refer <a href="#">section 3.6.3</a>)</li> <li>c) Use merge transform to perform insert/ update /delete of source changes</li> <li>d) w.r.t fact tables, use merge with false predicate feature and delete and re-ingest the corresponding partition data instead of scan-update of each record.</li> <li>e) Create materialized views for the frequently used queries and aggregations (refer <a href="#">section 3.6.6</a>)</li> </ul>
Consumption Layer	<p>KPI's which requires processing time can be pre-calculated at this step to gain performance at visualization layer.</p> <p><b>Recommendations: -</b></p> <p>Develop big query sql and execute through bq query utility. This utility can also be invoked from orchestration tool such as cloud composer.</p> <ul style="list-style-type: none"> <li>a) Partition the table based on appropriate columns which provides amazing cost benefits. (refer <a href="#">section 3.6.3</a>)</li> <li>b) Perform clustering on the filtering columns which are required in addition to partition column. Defining order of columns are very important which decides the storage. (refer <a href="#">section 3.6.3</a>)</li> <li>c) Use merge transform to perform insert/ update /delete of source changes</li> <li>d) w.r.t fact tables, use merge with false predicate feature and delete and re-ingest the corresponding partition data instead of scan-update of each record.</li> <li>e) Create authorized views to fetch only allowed columns and records from respective tables. Grant IAM access on these views to BI users.</li> <li>f) Create materialized views for the frequently used queries and aggregations (refer <a href="#">section 3.6.6</a>)</li> <li>g) Colum and row level security can also be defined to restrict PII access at table level using data catalog.</li> </ul>

### 3.6.3 Partitions and Clustering

#### **Partitions:**

Partitioning table is a best practise to achieve performance and cost benefits in Big Query. This reduces the amount to be scanned and processed. All the four layers illustrated in above section recommended to have partitions.

Currently, Big Query supports integer(range) and data/timestamp partitions. String type partitions are not supported at present. Maximum no.of partitions supported per table are 4000.

Use partitioning under the following circumstances: -

- You want to know query costs before a query run. Partition pruning is done before the query runs, so you can get the query cost after partitioning pruning through a dry run. Cluster pruning is done when the query runs, so the cost is known only after the query finishes.
- You need partition-level management. For example, you want to set a partition expiration time, load data to a specific partition, or delete partitions.
- You want to specify how the data is partitioned and what data is in each partition. For example, you want to define time granularity or define the ranges used to partition the table for integer range partitioning.

#### **Clustering:**

Order of data sorting will be based on the clustering columns. So, choosing the clustering columns are important. BQ avoids scanning of unnecessary data using these sorted blocks. A table can have clustering in the following scenarios: -

Use clustering under the following circumstances: -

- You don't need strict cost guarantees before running the query.
- You need more granularity than partitioning alone allows. To get clustering benefits in addition to partitioning benefits, you can use the *same column for both partitioning and clustering*.
- Your queries commonly use filters or aggregation against multiple particular columns.

- The cardinality of the number of values in a column or group of columns is large.

Prefer *clustering over partitioning* under the following circumstances:

- Partitioning results in a small amount of data per partition (approximately less than 1 GB).
- Partitioning results in a large number of partitions beyond the limits on partitioned tables.
- Partitioning results in your mutation operations modifying most partitions in the table frequently (for example, every few minutes).

You can also combine partitioning with clustering. Data is first partitioned and then data in each partition is clustered by the clustering columns.

When the table is queried, partitioning sets an upper bound of the query cost based on partition pruning. There might be other query cost savings when the query actually runs, because of cluster pruning.

#### 3.6.4 Automatic re-clustering

As data is added to a clustered table, the newly inserted data can be written to blocks that contain key ranges that overlap with the key ranges in previously written blocks. These overlapping keys weaken the sort property of the table.

To maintain the performance characteristics of a clustered table, BigQuery performs automatic re-clustering in the background to restore the sort property of the table. For partitioned tables, clustering is maintained for data within the scope of each partition.

#### 3.6.5 Denormalize tables with nested and repeated fields

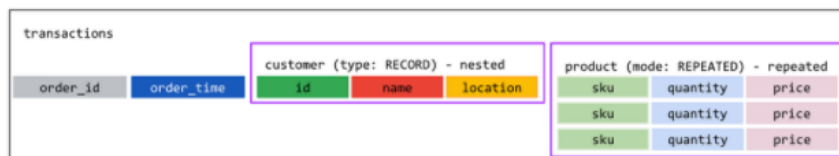
Multiple table joins on large volume fact and dimension table might slow down the querying performance. It is advised to take Big Query's nested and repeated structure to store all related (dimension & fact) data in same table rather than many tables. This avoids one-to-many relationship joins. Big Query supports multi level nested structures storage, It is always recommended to assess the usage (Querying / Visualization requirements) then design the structure accordingly. Looker (BI tool) supports nested / repeated values.

Example:-

Storing orders, customers and product data in same denorm table will provide amazing query performances



### Nested Repeated Fields



- Combining nested and repeated fields denormalizes a 1:many relationship without joins.
- Use dot notation to query a nested column and UNNEST() to flatten the repeated data.

Reference: Google Cloud Documentation

### 3.6.6 Materialized view

In Big Query, Materialized view is an option to store the results of frequently used queries and aggregations. If any of the queries to source table can be served from materialized view then Big Query automatically reroutes the query to materialized view which saves querying cost and gains performance. Performs incremental data refresh automatically in background at the regular intervals to update the base table changes. It costs for additional storage and data refresh as per pricing model.

### 3.6.7 User Defined Functions

Big Query supports temporary / persistent User Defined Functions (UDF) using sql expressions or Java script. UDF's are suitable to define some reusable expressions, calculations, data parsing etc., UDF functions can be invoked through select statement, accepts table columns/expressions as input parameters and returns output values.

### 3.6.8 Separate projects for consumption billing

Create separate projects based on the data consumption requirements. Required authorized views / materialized views can be defined to fetch data from consumption datasets. This enables to configure different billing account for the consumption related charges.

### 3.6.9 Big Query ML

Machine Learning models can be created, trained and evaluated easily using SQL like statements through Big Query ML. As the data already available in Big Query, machine learning model can be built quickly without data movement operation. It supports various machine learning models like Regression, Classification, K-means clustering, Matrix factorization, etc., Predictions can be performed through SQL

statements. Trained models can be deployed into AI platform service for real time predictions.

### 3.7 Data Services Layer

#### 3.7.1 Cloud Run

Define cloud run service to respond requests from downstream applications.

#### 3.7.2 Cloud Endpoint

Protect API's through JSON Web Tokens and define Quota's on API usage.

### 3.8 Frameworks and Accelerators

Some of the frameworks and accelerators which can be used are:-

- **Data Validation Tool (DVT):-**

Data migration between different data sources can be validated through google provided python framework called Data Validation Tool(DVT) which saves lots of coding and manual efforts

- **Dataflow ingestion framework:**

Ingesting data from various sources into Big Query can be automated through dataflow re-usable pipelines based on config driven. Also, repeated data quality cleansing / transformations can also be automated through this framework. This avoids coding tasks for each data sources.

### 3.9 Data Orchestration

Entire ingestions data pipelines can be orchestrated through cloud composer. Builds DAG (Direct Acyclic Graph) based on the dependencies between tasks, parallel runs, scheduling etc., Cloud composer service built on top of Airflow tool and supports python scripting. Following are the high-level steps to define tasks in composer: -

- a) Import necessary python libraries
- b) Set default\_args and instantiate DAG object with appropriate parameters like start\_date , scheduling\_interval etc.,
- c) Define each tasks through invoking suitable airflow operator functions
- d) Define dependency between various tasks



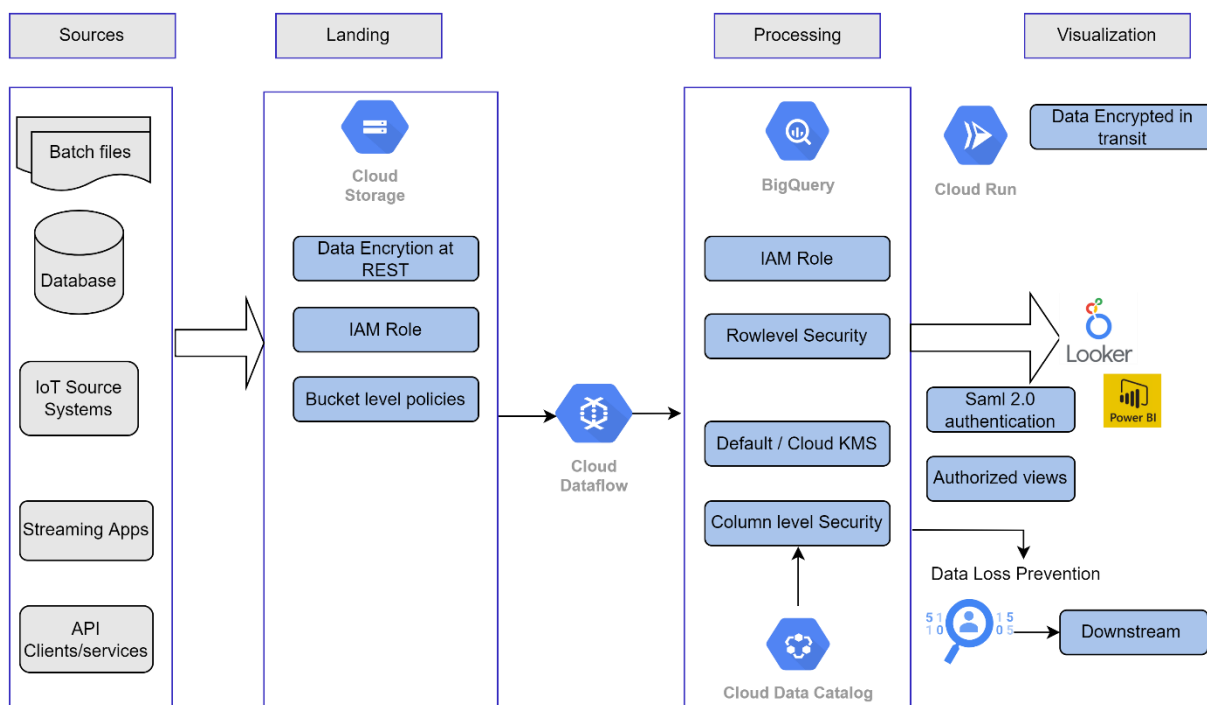
**Recommendations:** -

- a) Define execution sequence for tasks to take care of dependencies.
- b) Set "retries" and "retry\_delay" properties to enable automatic retries if in case of DAG failures
- c) Avoid executing resource intensive operation in composer which will consume resources of worker nodes and also impacts scheduling operations.
- d) Always set start\_date parameter to future date to avoid immediate run after deployment
- e) "gcs\_to\_bq", bash\_operator, python\_operator, bigquery\_to\_gcs, mysql\_to\_gcs, etc., are the most used airflow operators related to big query operations.

### 3.10 Data Security

Data security can be defined at each stages of data lake as suggested in the following diagram and reference table:-

- Define user groups in IAM and assign necessary pre-defined roles
- Assign most required privileges to service accounts and provide access to individual user who needs to use service account
- KMS, row/column level security, authorized views are some of the most used security features in google cloud



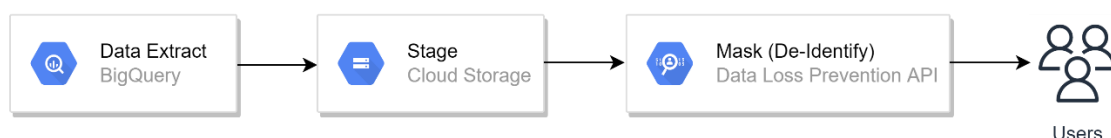
Stage	Component	Data in Transit	Key Storage	Data at Rest	Key Storage	Authentication	Authorization	Anonymization
<b>Data Ingestion</b>								
Cloud Run	Fully fully managed HTTP drivers containers	SSL / TLS	-NA-	-NA-	-NA-	Configure IAM role	-NA-	-NA-
Cloud IoT Core	IoT Ingestion from devices	MQTT, TLS handshake	-NA-	-NA-	-NA-	Per device public/private authentication using JSON Web Tokens - device security, IAM role	Access control with IAM	-NA-
Cloud Pub/Sub	Migration service	SSL / TLS	KMS	-NA-	KMS	Service account, user account	Access control with IAM	-NA-
Google Cloud Transfer Service	Transfer data from on Prem to cloud	Use TLS encryption for HTTPS connection	KMS	After transfer SSE/ S3 or KMS	-NA-	Cloud IAM	Access control with am	-NA-

<b>Data Processing</b>								
Cloud Dataflow	Data Processing	VPC	-NA-	-NA-	-NA-	Service Account	IAM Role	-NA-
<b>Storage</b>								
GCS	Files storage	SSL/TLS	Cloud KMS	Server Side Encryption	Cloud KMS	Cloud IAM	User / Bucket Policy	-NA-
<b>Data Governance</b>								
Cloud Data Catalog	Metadata management	-NA-	-NA-	-NA-	-NA-	Cloud IAM	IAM Roles	-NA-
<b>Data Warehouse</b>								
Big Query	Serverless Data Warehouse	TLS	Cloud KMS	AES, CMEK, Client Side Encryption	Cloud KMS	Cloud IAM	-NA-	-NA-

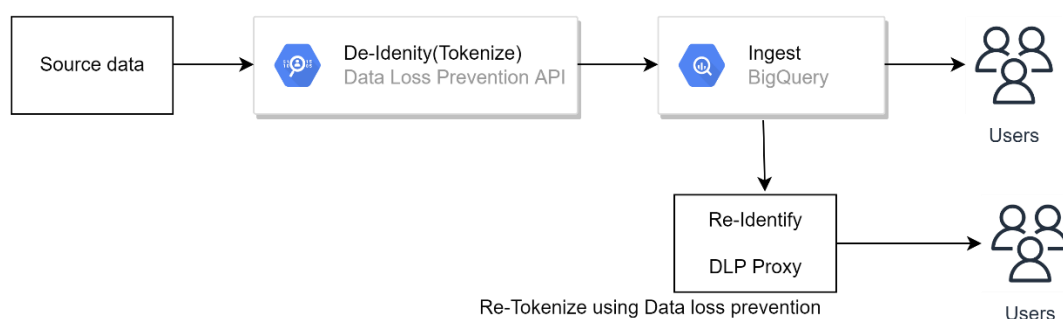
### 3.11 Data Compliance

Securing and managing Personal Identifiable Information (PII) are the most important activity of data lake design. PII data leakage can be avoided by using google cloud service DLP (Data Loss Prevention) API. DLP can also be used to detect and catch specific patterns in regular data to send warnings of possible PII violations.

De-Identify PII data before sharing data to downstreams/users



Dynamic unmasking through DLP Proxy

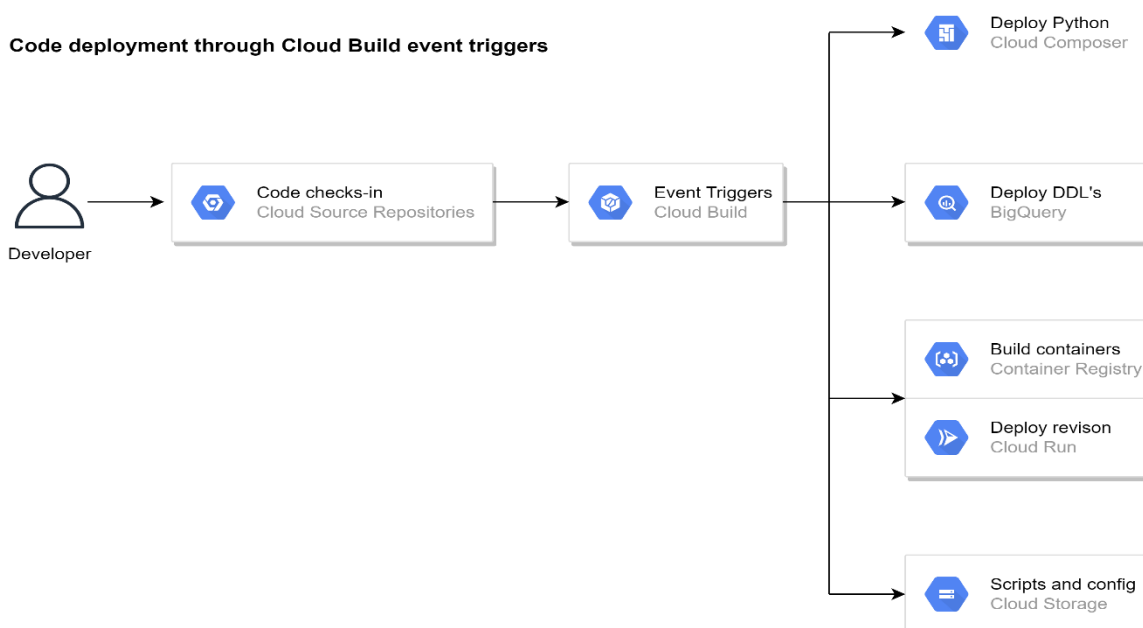


### 3.12 Code deployment

#### 3.12.1 Automated Code deployment

Google cloud provides following services to support **automated code deployments**:

Code deployment through Cloud Build event triggers



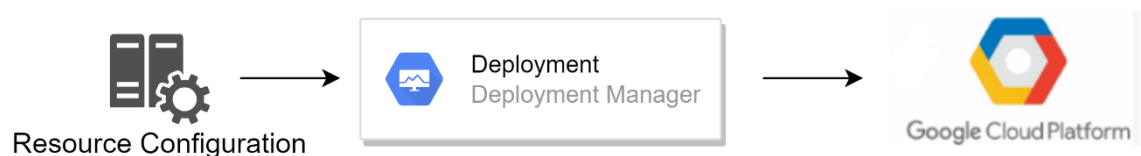
- a) **Cloud Source Repository(CSR)**: Version control similar to github where developer can create branches and check-in script changes
- b) **Cloud Build** : Create container images and checks-in to Google Container Registry(GCR)

### 3.12.2 Infrastructure automation

Google cloud supports infrastructure as code through Cloud Deployment Manager (CDM) service. The services of entire data lake and its associated configurations can be defined using resource templates (Jinja and Python).





This template can be checked into version control to manage version history.


#### Infrastructure deployments using CDM



### 3.13 Monitoring and Troubleshooting

Google cloud operations suite offers integrated application monitoring and troubleshooting for services which runs on cloud and on-prem.

Services	Description
 Monitoring	Collects metrics, events metadata from applications and generates insights, charts and alerts
 Logging	Store, search, analyse, monitor and alert on log data and events
 Tracing	Collects latency data from app engine and other applications and displays in near real time
 Debugging	Inspect the state of an application at any code location without stopping or slowing it down

 Error Reporting	Aggregates and displays errors produced in cloud services
--	---

### 3.13.1 Metrics

Some of the recommended metrics for cloud services part of data lake reference architecture are: -

Service name	Example Metrics to collect and monitor
Cloud Storage	Totalbytes Objectcount Requestcount Receivedbytes
Pubsub	Performance: Message Size Oldest message age acked/unacked Messages Dead Letter Retained messages Publish requests Latency: subscription/seek_request_count subscription/ack_latencies subscription/expired_ack_deadlines_count
Dataflow	Data freshness System latency Element count is_failed Autoscaling Throughput CPU utilization Worker error log count Input and Output Metrics
Big Query	Project Query count Slots used by project, reservation and job_type Statement scanned bytes

	Stored bytes Table count
Cloud Run	Request count Request latency x <sup>th</sup> percentile Billable instance time Container memory allocation x <sup>th</sup> percentile

## 4 Appendix

## 5 Glossary

## 6 Document References

Google cloud documentation: <https://cloud.google.com/docs>