|  |  |  |
| --- | --- | --- |
| Storage | Object storage | GCS |
| Compute | Function as a Service | Cloud Function |
|  | Batch Scheduling | Cloud Scheduler |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Storage – Google cloud Storage (GCS):-**

* Introduction:
  + It is an Object storage service where you can store objects inside buckets. Buckets are nothing but Container to store your objects.
  + Bucket name should be unique across globally in GCP environment.
  + you can store your objects in any format with unlimited Size.
* Concepts
  + Buckets
  + Objects
  + Object Versioning
  + Storage classes
    - Standard Storage 🡪 Suitable for Frequently accessed.
    - Nearline Storage 🡪 Suitable for accessed once in a month.
    - Coldline Storage 🡪 Suitable for accessed once in a quarter.
    - Archive Storage 🡪 Suitable for accessed once in a year.
  + Encryption
  + Replication
  + Event Notification
  + Life cycle policy.

**Compute - Cloud function:-**

* Introduction
  + Cloud function is a function as a Service which provides a Serverless Run time environment for executing your code.
  + Suitable for Simple requirement can be done via multiple programming languages like Python, Java, C#, Ruby etc.
  + It is Event driven compute Service.
* Limitations
  + There is a Time limit to run cloud function such as 9 mints for 1st gen. and 10 mints for Second gen.
  + Memory for processing - 8 GB (1st Gen) and 16 GB (for 2nd Gen)
* How to Trigger Cloud function
  + Using HTTP
  + Trigger via Event driven approach Such as GCS, Pub/Sub.
* How to Debug and Monitor Cloud function?
  + Use Cloud Logging and Cloud Monitoring
* Performance Improvement
  + Use limited Logs in the code, So that Execution time will be used for functionalities
  + Use Global variables to pass arguments externally.

**Cloud Scheduler:-**

* Cloud Scheduler is a Cron based job scheduler which helps to schedule any job including batch jobs.

**Databases - relational:-**

* Cloud SQL
  + Cloud SQL is a relational database service which helps to Setup, Manage relational databases in GCP. which supports Structured data, OLTP, ACID Transactions and Strong Consistency.
  + Automated Backups/Scheduled Backups and restore.
  + We can Import and Export data from Cloud SQL.
  + Suitable for Small scale Applications and Store <10TB. There is no Horizontal Scalability.
  + Supports Single Region.
* Cloud Spanner
  + Cloud Spanner is Advanced SQL than Cloud SQL. which supports small- and large-scale application.
  + Provides Horizontal Scalability
  + Supports Single and Multiple regions.

**No SQL Databases – Big Table:-**

* Introduction:-
  + BigTable is a column oriented No SQL Database in GCP which helps to Store large volume of Structured, Semi Structured and Un-Structured data at Scale.
  + BigTable is designed based on HBase compatibility.
* Concepts
  + Row Key
    - Row key will be act as primary key of the table. Default index will be done based on Row key.
    - There is no index or Secondary index will be there in Bigtable.
    - No Joins supported.
* Bigtable Instance
  + Bigtable instance is a container for Bigtable clusters which has more than one cluster uses replication. You can create clusters in up to 8 regions.
* Steps to Create BigTable
  + Basis Steps
    - Select the Google cloud Project.
    - Make sure that billing is enabled for your Google Cloud project.
    - Enable the Cloud Bigtable API.
    - Install/Open Google Cloud CLI. --> Initialize the gcloud CLI --> gcloud init
    - If you installed the gcloud CLI previously, make sure you have the latest version by running gcloud components update.
* Create Instance
  + Go to Create Instance and give details.
* Create Table
  + Open the Instance list and Select the Instance where you want to create tables.
  + Click Create table to create table.
* How to connect BigTable
  + Google cloud Console
  + Google cloud CLI
* Big table Architecture
  + Write operation.
    - Write operation is submitted to the Bigtable front end server, Front end server will route the request to Instance.
    - Instance will share the request to multiple tablets, each tablet writes the data into SSTable.
    - SSTable will be flushed to Colossus as shared logs.
* Read operation.
  + Read operation is submitted to the Bigtable front end server, Front end server will route the request to Instance.
  + Instance will split the Query into Mutiple small quries and routed to all tablets where actual data resides.
  + Each tablet will check the data present in SStable, if not, load shared log from Colossus into SSTable and search it.
  + Fetch the result and share it to instance, instance will collact all the results and send it to Client.

**BigQuery**

* Introduction
  + Big query is a Serverless data warehouse service which helps to manage large volume of data and perform Analytical queries.
  + BigQuery uses MPP Architecture to process data.
  + Columnar Storage
  + provides BigQuery ML and BI Engine
  + BigQuery resources such as Datasets, tables, Jobs are associated with projects. Projects is a high-level container in GCP.
* Concepts:-
  + Datasets
    - Dataset is a container for organizing one or more tables in BigQuery.
* Tables
  + Tables are used to store data in Rows and Columns. They can be either native BigQuery tables or external tables (e.g., Bigtable, Google Cloud Storage, Google Drive). It store data in various formats, including Avro, CSV, JSON, ORC, Parquet, and more.
* Jobs
  + Jobs are nothing but operations performed in Bigquery like table creation/deletion, query execution and data import/export. you can monitor their progress and status using the BigQuery web UI, command-line tools, or API.
* Partitioning and Clustering:
  + Partitioning helps to divide table into multiple chunks based on partitioned column defined. clustering helps to organize data within partitions based on one or more clustering columns.
* Authorized Views
  + Authorized views are used to share query results with users who do not have direct access to the underlying data.
* Data Transfer Service
  + Data Transfer Service helps to automate the ingestion of data from external sources (e.g., Google Ads, YouTube, Salesforce) into BigQuery. It is having pre-built connectors and scheduling options for loading data incrementally or in batches.
* BigQuery ML
  + BigQuery ML helps to train, deploy, and serve machine learning models directly from BigQuery data.
* Pros of BigQuery
  + Serverless Architecture
  + Scalability
  + High Performance
  + BigQuery supports real-time data ingestion and analysis through streaming data capabilities.
* Cons of BigQuery
  + Cost will be very high when you are dealing with very huge data.
* BigQuery Architecture
  + Colossues (Storage)
    - Colossues is a DFS which helps to store data.
* Dremel (Execution Engine)
  + Dremel is an Execution engine which converts the SQL query into Execution tree. Slots/leaves will read data from Colossues and Mixers/Branches will do aggregation and share it to client.
  + In-between Shuffle, use Jupiter to move data between tasks.
* Jupiter
  + Jupiter is used to Distribute the workloads.
* Borg
  + Borg is responsible for allocating resources to the jobs.
* BigQuery Execution flow:-
  + User is submitting a query to BigQuery. BigQuery parses the SQL statement and generates an execution plan. That execution plan will be executed by underlying resources.
  + BigQuery uses MPP architecture to creates fast execution plan for executing the tasks very faster.
* How to Schedule BigQuery queries.?
  + We have an options to schedule queries in BigQuery Console.
* Types of Views
  + Standard View
  + Materialized views
* Performance Tunning
  + Design table schema to minimize data shuffling and reduce the number of JOIN operations to improve query performance.
  + Use Partitioning and Clustering wherever required.
  + Avoid Full Table Scans
  + Use Table Decorators (e.g., \_PARTITIONTIME, \_TABLE\_SUFFIX) to query specific partitions or date ranges within tables. This can reduce the amount of data scanned.
  + Break down complex queries into smaller, to improve performance.
  + Use wildcard functions (e.g., \_TABLE\_SUFFIX) to query multiple tables with similar schemas in a single query. It will reduce query complexity.
  + Apply filters and predicates (e.g., WHERE clauses, HAVING clauses) to limit the amount of data scanned by queries.
  + Use caching feature wherever required.

**Dataflow:-**

* Introduction
  + Dataflow is a Serverless Data processing Service which helps to build data processing Pipeline using Apache Beam SDK.
  + Supports both Batch and Streaming process.
  + Dataflow is also a Runner which helps to execute the Apache beam libraries.
* How does dataflow work?
  + Data processing involves three steps,
    - Read data from Source (Files, BigQuery, BigTable, Pub/Sub, Custom Source)
    - Transform data.
    - Write data back to the Sink (Files, BigQuery, BigTable, Pub/Sub, Custom Source)
* How dataflow internal working.?
  + When you are executing the data flow job, It will spin up a cluster of virtual machines in Dataproc.
  + Distributes the tasks in your job to the VMs, and dynamically scale the cluster based on how the job is performing.
* Concepts
  + Pipeline
    - It is a DAG based Pipeline which performs one or more PTransforms (processing transforms) connected by data sources and sinks.
* PTransform (Processing Transform):
  + PTransform is a transformation which can be applied on top of DataFrame.
* PCollection (Parallel Collection)
  + PCollection is nothing like dataframe.
* Windowing
  + Windowing is used to partition data streams into finite, logical time intervals called windows.
* Triggers
* Watermarks
* Side Inputs
  + Side inputs are additional data sources that can be accessed by PTransforms during pipeline execution.
* Use cases of Data flow
  + dataflow is a great choice for any batch and Streaming data that needs processing and enrichment for the Downstream systems such as analysis, ML or Datawarehouse.
* How to Schedule Dataflow jobs?
  + Cloud Scheudler --> Cloud function to trigger --> Dataflow job
  + Composer --> Define dataflow jobs in Workflow and Schedule it through Composer.
  + Cloud Scheduler jobs to trigger Dataflow jobs at predefined intervals or specific times.
* How to Monitor Data flow jobs?
  + Execution of scheduled Dataflow jobs using Cloud Monitoring or Cloud Logging. Also, you can view job status, logs, and metrics in the Google Cloud Console or programmatically using the Dataflow API.
* How to Debug Dataflow jobs?
  + Utilize Cloud Logging and Cloud Monitoring to monitor the execution of Dataflow jobs in Google cloud console.
* How to pass external parameters to Dataflow job?
  + Use Environment Variables to pass arguments to the data flow job.
  + Configuration Files
* How to pass External Libraries to Dataflow jobs?
  + Download and upload JAR files to GCS.
    - gsutil cp target/my-pipeline.jar gs://my-bucket/
* Specify JAR Location in Dataflow Job Submission
  + gcloud dataflow jobs run my-job \

--gcs-location gs://dataflow-templates/latest/Word\_Count \

--region us-central1 \

--jar gs://my-bucket/my-pipeline.jar \

--parameters input=gs://input-bucket/input.txt,output=gs://output-bucket/output.txt

* Import library name to the code.

**Data Fusion:-**

* Introduction
  + Data fusion is a data integration service for quickly building and managing data pipelines to cleanse, prepare, transfer, and transform data without infrastructure setup.
  + It provides a Visual Interface to build and manage Pipelines. No Coding is required.
  + Supports Batch and streaming data processing.
* Use cases of Data fusion
  + Data Fusion to create data pipelines for data integration and data migration tasks and use Dataflow to create data pipelines for batch and stream processing tasks.
* How to Schedule Data Fusion jobs?
  + Cloud Scheudler --> Cloud function to trigger --> Data Fusion job
  + Composer --> Define Data Fusion jobs in Workflow and Schedule it through Composer.
  + Cloud Scheduler jobs to trigger Data Fusion jobs at predefined intervals or specific times.
* How to Monitor Data Fusion jobs?
  + Data fusion UI
  + Execution of scheduled Dataflow jobs using Cloud Monitoring or Cloud Logging. Also, you can view job status, logs, and metrics in the Google Cloud Console or programmatically using the Dataflow API.
* How to Debug Data Fusion jobs?
  + Utilize Cloud Logging and Cloud Monitoring to monitor the execution of Dataflow jobs in Google cloud console.
* How to pass external parameters to Data Fusion job?
  + Define Pipeline Parameters

- How to pass External Libraries to Data Fusion jobs?

- Download and upload JAR files to GCS.

- gsutil cp target/my-pipeline.jar gs://my-bucket/

- Specify JAR Location in Data Fusion Job Configuration

**DataPrep:-**

* Introduction
  + DataPrep helps to visually exploring, cleaning, and preparing structured and unstructured data for analysis, reporting, and machine learning.
  + we can read from Multiple Datasource and Clean and Combine data through Visually and write data to Sink.
  + DataPrep job internally uses data flow to transform your data.

**Dataproc:-**

* Introduction
  + Dataproc is Bigdata processing service which helps to perform processing through popular Hadoop tools such Hive, Impala, Spark, Flink.
* Concepts
  + Dataproc Metastore
    - It is a centralized metadata repository which can be accessed by different Query engines such as Hive, Impala, Spark etc.
* Integrate other GCP Services:-
  + We can easily read from and write data with other Google Cloud managed services such as BigQuery, Cloud Storage, Pub/Sub, and Dataflow.

**Pub/Sub:- (Messaging and Notifications)**

* Introduction
  + It is a Messaging System which helps to Publish and Subscribe the message through Topics.
  + Event Driven Architecture.
* Different Pub/Sub Patterns:-
  + Many to one (Many Publishers and One Subscriber)
  + Many to Many (Load balancing)
  + One to Many
* Architecture
  + Publisher publish the message to the Topics and message is stored in Message Storage.
  + Subcriber will consume message through Subscription and give Ack to Subscription, message will be deleted from the message storge.

**Data Catalog:-**

- Data catalog is metadata management service which helps to catalog the data assets stored in GCS. It syncs technical metadata automatically and create tags for Business metadata. It provides Search interface.

- Single source of truth for all your data assets.

- Data catalog is a tool which brings the metadata from multiple data assets.

- with Data catalog, Along with Technical metadata, you create a Tag templates which you can add multiple tags about your assets. (Business metadata - Business owner, Confidential, retention, has\_pii)

- if your data assets are exists in different system and looking for common Data catalog to explore, Data catalog would be good choice. It is not only supporting GCP. They are having open source connectors to connect Hive, Oracle, Impala etc.

- You will be able to search all your business metadata in Single unified view.

- It is a centralized metadata management service which fetch technical metadata from multiple GCP Services such GCS, BigQuery, Pub/Sub and Data proc metastore. Along with that, it supports for On-prem tools as well like Hive, Impala, Oracle, SQL Server, MySQL, Teradata, Looker, Tableau.

A screenshot of a computer

Description automatically generated

**Composer:-**

* Introduction
  + Composer is a workflow orchestration service built on Apache Airflow. which helps to orchestrate multiple data pipelines programmatically (Python), ETL processes on Schedule basis and you can monitor the jobs as well.
* When to use Composer?
  + Collection of jobs from different services needs to be run in right order.
* Concepts
  + Workflows
    - Workflows are nothing but series of Tasks which performs Ingestion, Curation and Transformation. In Airflow, Workflow is created by DAGs.
* DAGs
  + DAG is a Collection of Tasks which we want to Schedule and run. DAG is created by using Python scripts.
* Tasks
  + Each task in the DAG performs some operations through Jobs.
* Cloud Composer Environment:-
  + To run workflows, first we need to create an environment. environment is created in Google Kubernetes Engine. They will use connectors to connect GCP resources. You can create multiple environments under single project.