<https://massmutual.atlassian.net/wiki/spaces/SRE/pages/1644561340/Kubernetes%2B-%2BIntroduction%2Band%2BMore>

Kylin -

OLAP engine at scale .

Incremental build of cubes .

Provide the Hive Dimension and Fact table .

User friendly webUI for monitoring .

Stores Underlying Hbase .

 Data analysts were exporting data out of Hadoop into OLAP and other SQL query-based systems, so they could find what they wanted, but that added steps to a process that needed to occur faster.

OLAP (online analytical processing) technology is not new. Building data cubes that can be viewed from a variety of angles was a well-established practice before Hadoop was invented. But Kylin enabled cube-building on a massive scale. Before the views can be achieved, hundreds of billions of rows in Hadoop must be indexed. Kylin’s ability to build "smart indexes" on that scale is one of the things that sets it apart, said Saha.

Debashis Saha, vice president of commerce platform and infrastructure at eBay.

With the indexes already built, Kylin users can then achieve faster views and more useful results from large amounts of Hadoop data. "You can take a more granular level of the data and find results that satisfy these (specific) criteria," he said.

Diagram

Description automatically generated

<https://www.tigeranalytics.com/blog/apache-kylin-architecture/>

<https://medium.com/@gopinath530/apache-kylin-on-kubernetes-bf25ed045f7c>

Docker -

The most popular container technology utilized today is Docker.  Simply put, Docker allows you to run each component in a separate container with its own libraries and its own dependencies all on the same VM and O.S. ***but within separate environments or containers***.

**Why Docker ?**

* + Assume you want to Deploy end to end components like node.js(web server), database component(SQL Server), messaging server(Kafka) and reporting(tableau) to a server machine .Each of these components need different version of libraries and different version OS to install. this creates a huge issue to manage.
  + After sometime , we started deploying each components in different virtual server where the web server will be hosted in separate VM , Database is separate VM and reporting in VM . each VM communicate accordingly.
  + Docker helps to deploy each components in a separate container and link between those containers(thru orchestration) within the same machine . Each containers will be run with separate Lib and separate OS as needed within same machine.

<https://massmutual.atlassian.net/wiki/spaces/TERSUN/pages/1920144761/Docker+and+Kubernetes>

GCP –

2 types of Encryptions -

* + **Data** encryption: By default **GCP** always encrypts all customer **data** at rest as well as in motion. This encryption is automatic, and it requires no action on the user's part. Persistent disks, for instance, are already encrypted using AES-256, and the keys themselves are encrypted with master keys

* + *Encryption at rest —*used to protect data that is stored on a disk (including solid-state drives) or backup media.
  + *Encryption in transit* — used to protect data that is traveling over the Internet, moving within Google’s infrastructure

<https://medium.com/google-cloud/understanding-data-encryption-in-google-cloud-c36d9095fb38>

A picture containing diagram

Description automatically generated

Client Side Encryption -

Diagram

Description automatically generated with medium confidence

Google GCP comes with customized storage classes namely:

* + Regional – for frequent use.
  + Nearline – for infrequent use.
  + Coldline – for long-term storage.

**Big Query -**

<https://medium.com/dataseries/costs-and-performance-lessons-after-using-bigquery-with-terabytes-of-data-54a5809ac912>

<https://cloud.google.com/bigquery/docs/slots>

Big Query has 2 components - Both components billings are different.

[**Analysis pricing**](https://cloud.google.com/bigquery/docs/slots#analysis_pricing_models)  - is the cost to process queries, including SQL queries, user-defined functions, scripts, and certain data manipulation language (DML) and data definition language (DDL) statements that scan tables.

[**Storage pricing**](https://cloud.google.com/bigquery/docs/slots#storage) - is the cost to store data that you load into BigQuery.

**Analysis Pricing -**

By Default , its on demand pricing model.

Slots - A BigQuery slot is a virtual CPU(not vms) used by BigQuery to execute SQL queries. BigQuery automatically calculates how many slots are required by each query, depending on query size and complexity.

You have a choice of using an [on-demand pricing model](https://cloud.google.com/bigquery/pricing#on_demand_pricing) or a [flat-rate pricing model](https://cloud.google.com/bigquery/pricing#flat_rate_pricing). Both use slots for data processing. The flat-rate model gives you explicit control over slots and analytics capacity, whereas the on-demand model does not.

**Flat pricing Model** Pricing- Fixed number of Slots(at project level, check this) .

1) User Query -> Google big query put DAG and decides the no of slots . If it requires 1000 slots and 500 is available . It takes the 500 and starts the stages and wait for the slots to be available in the query . If the slots are not available, the request is put in queue.

In Flat pricing , you own the cpus(slots). Based on the monthly/weekly options chosen.

**On Demand Pricing** - On demand pricing doesn't works via slots . Its billed based on the no of bytes processed. your queries run using a shared pool of slots, so performance can vary (assume big query optimizer tells it needs 1o slots but 8 slots available then it tries to run the query with available so the performance might vary) Ensure partitioning and clustering are correctly used.

**Storage Pricing -**

* + **Active:** A monthly charge for data stored in tables or in partitions that have been modified in the last 90 days. **$0.020 per GB**
  + **Long-term:** A lower monthly charge for data stored in tables or in partitions that have not been modified in the last 90 days. **$0.010 per GB**

**How partition helps in storage cost?**

It makes sense , partitioning table helps in reads by consumption? But how in this case helps in storage ? This is because if a specific partition is not used then it will be consider as long term plan Only the updated partitions will be considered for active storage . In case if the table is not partitioned then the entire table will be consider for active storage pricing.

**Big Query important Concepts -**

Partitioning and Bucketing - Same as Hive . Better to partition and use the partitioning in queries to avoid the byte reads.

**Big Query Performance Tuning in terms of Cost -**

* + Avoid Select \*. Choose only needed columns ( no of bytes read will be less)
  + Pls note , APLLYING LIMIT doesn’t help. Entire Data is processed before applying limit. Charges are based on the amount of bytes read.
  + Use the query Validator before running the Queries to check the bytes used for the query . Optimize before running it.
  + Don’t preview the data(this takes bytes as well) , use query validator or table view to check the data or use dry run parameter.
  + Use pricing calculator (provide the no of bytes and get the charges associated with it).
  + Use clustered and partitioned tables.
  + Materialize the query results to stages .

If you create a large, multi-stage query, each time you run it, BigQuery reads all the data that is required by the query. You are billed for all the data that is read each time the query is run.

Instead, break your query into stages where each stage materializes the query results by writing them to a [destination table](https://cloud.google.com/bigquery/querying-data#permanent-table). Querying the smaller destination table reduces the amount of data that is read and lowers costs. The cost of storing the materialized results is much less than the cost of processing large amounts of data.

* + Use streaming insert with caution -

There is no charge for loading data to Bigquery . There is a charge, however, for [streaming data](https://cloud.google.com/bigquery/streaming-data-into-bigquery) into BigQuery. Unless your data must be immediately available, load your data rather than streaming it.

**DataFlow -**

Google Dataflow is a [managed service from Google Cloud](https://itsvit.com/services/cloud-computing/google-cloud-services/) using Apache Beam SDK to build serverless batch and streaming data processing pipelines with unlimited horizontal scalability.

Main concepts -

<https://cloud.google.com/dataflow/docs/concepts/beam-programming-model>

* Pipelines -
* PCollections-
* Transforms -
* ParDO -
* Pipeline I/O -
* Aggregations -

How to Load from a CSV file to Cloud datastore ?

<https://medium.com/google-cloud/large-data-processing-with-cloud-dataflow-and-cloud-datastore-839aae5ee372>

Dataflow runs with serverless(on demand cluster) and I don’t think we can set up a 24 \* 7 the cluster for dataflow (check this).

**Cloud Functions -**

Cloud functions are the glue code that can make your managed service backed applications or workflows more efficient and insightful.

That means you can **host your functions**(yes, read it again — functions) on the cloud and execute them **only when you need them**. You **don’t have to maintain your own server** for that. And the most impressive part is that — **you’ll be billed for only what you’ve used**(memory and disk usage, executions, bandwidth, etc)! Also It’s our friend **nodejs** on the server!

<https://medium.com/infancyit/introduction-to-google-cloud-functions-8029b072fa21>

PUB\_SUB -

Who are the components here?

1. Topic

2. Subscription.

Data is published to the topic, the topic can have multi subscriptions, each subscription serves one consumer.

Needless to say that any subscription is isolated from the others, it just sets the stream state for any consumer.

Scaling:

You don’t have to worry about scaling, the PubSub “knows” to adapt itself by the rate of published messages. no need to “dirty your hands” with shards/partitions, GCP is arranging it for you.

Monitoring

You can monitor PubSub with StackDriver metrics, you can do it at the topic level or the subscription level.

These 2 metrics show that the subscription has messages that it can’t ack in time.

**PubSub**:

* **zero operation**, no need to define

**PubSub**:

* **zero operation**, no need to define partitions and shards.
* **Scaling** is built-in without any required operation(at least check that you don’t hit some quota limit).
* **Pay** only for usage.
* **Monitoring** is friendly and simple.
* **Producer code** is straight forward, without handling partitions/ shard errors/ backpressure.

<https://medium.com/google-cloud/things-i-wish-i-knew-about-google-cloud-pub-sub-part-2-b037f1f08318>

Diagram

Description automatically generated

In Google Cloud Platform, PubSub is a messaging queue service that allows you to send and receive messages between independent applications. PubSub plays a critical role in NOT making apps heavily rely on each other; PubSub decouples apps, meaning apps performing its task independently but still working together as a part of the system deployed in cloud. As in the previous scenario, instead of letting the apps directly interacting with each other, PubSub comes in between the database and the API sitting in App Engine or somewhere else and the database publishes messages thru PubSub and the subscriber, the API, receives messages via the same means. In this fashion, even if the subscriber breaks down, PubSub holds messages until it comes back online and sends them once it come back online, meaning they are decoupled.

What is Bigquery Omni -

<https://cloud.google.com/blog/products/data-analytics/introducing-bigquery-omni>

<https://blog.openbridge.com/bigquery-omni-distributed-query-engine-comes-to-google-cloud-f23b34c87362>

Amazon Athena, Amazon Redshift, PrestoDB, and others support this model. Google BigQuery supported this pattern as well before Omni, except only it could only query data resident within GCP.

However, what is novel about the approach Google has taken goes beyond the distribution of a query but where the actual compute resources that execute those queries reside. Truly distributed queries leveraging native compute resources is a unique offering. Assuming the aforementioned Omni service works as advertised, this will extend the reach for current BigQuery customers that operate in AWS and Azure.

BigQuery Omni, a flexible, multi-cloud analytics solution that lets you cost-effectively access and securely analyze data across Google Cloud, Amazon Web Services (AWS), and Azure (coming soon), without leaving the familiar BigQuery user interface (UI).

Behind the scenes, BigQuery’s query engine is running on our Anthos clusters within the BigQuery managed service. BigQuery gets the data from data storage within your account once you’ve authorized permissions via your other public clouds’ IAM roles. Note that data is moved temporarily within AWS from your data storage to the BigQuery clusters running on Anthos to execute queries.

Advantage -

* Can be used to query across different cloud platforms without not needing to move the data from one cloud to another cloud .
* Is this same as presto?

1)What is Google Cloud composes?

A fully managed workflow orchestration service built on Apache Airflow.

Advanatages -

1. What is BigQuery ML ?

BigQuery ML lets you create and execute machine learning models in BigQuery using standard SQL queries. BigQuery ML democratizes machine learning by letting SQL practitioners build models using existing SQL tools and skills. BigQuery ML increases development speed by eliminating the need to move data.

Advantages of Bigquery ML -

* Don’t need to move the Data.
* Create all ML models with the SQL in bigquery .

1. What is Cloud Spanner ?

2 types of Encryptions -

* **Data** encryption: By default **GCP** always encrypts all customer **data** at rest as well as in motion. This encryption is automatic, and it requires no action on the user's part. Persistent disks, for instance, are already encrypted using AES-256, and the keys themselves are encrypted with master keys

* *Encryption at rest —*used to protect data that is stored on a disk (including solid-state drives) or backup media.
* *Encryption in transit* — used to protect data that is traveling over the Internet, moving within Google’s infrastructure

<https://medium.com/google-cloud/understanding-data-encryption-in-google-cloud-c36d9095fb38>

A picture containing diagram

Description automatically generated

**What is DataProc ?**

Dataproc is a managed Spark and Hadoop service that lets you take advantage of open source data tools for batch processing, querying, streaming, and machine learning.

<https://cloud.google.com/dataproc/docs/resources/faq>

**Brief explanation of how does Dataproc works:**

It disaggregates storage & compute. Say an external application is sending logs that you want to analyze, you store them in a data source. From Cloud Storage(GCS) the data is used by Dataproc for processing which then stores it back into GCS, [BigQuery](https://www.youtube.com/watch?v=So-tVyBQt8E&list=PLTWE_lmu2InBzuPmOcgAYP7U80a87cpJd&index=20) or Bigtable. You could also use the data for Analysis in a notebook and send logs to [Cloud Monitoring and Logging](https://www.youtube.com/watch?v=Y7L2y6NVa9Y&list=PLTWE_lmu2InBzuPmOcgAYP7U80a87cpJd&index=9).

Since storage is separate, for a long-lived cluster you could have one cluster per job but to save cost you could use ephemeral clusters that are grouped and selected by labels. And finally, you can also use the right amount of memory, CPU and Disk to fit the needs of your application.

Diagram

Description automatically generated

A Dataproc cluster is made up of a master and slaves.

You can specify the number of processors, memory and storage and the master as well as the slaves.

How to run a pyspark jobs in Dataproc - steps.

<https://medium.com/@sourabhsjain/running-pyspark-jobs-on-dataproc-cluster-using-workflow-templates-7fba307aa663>

Ways to connect to DataProc ?

* browser-based [Google Cloud Console](https://console.cloud.google.com/?_ga=2.155474939.1174049484.1623074945-678233387.1622168214)
* As Dataproc is integrated with cloud SDK, we can use **gcloud** command line.
* Use Dataproc rest api - for programmatic access
* We can use ssh connections to master or worker nodes in your cluster.

How Dataproc Clusters(vms) Default storage ?

* DataProc uses HDFS as default storage .
* All Vm's comes with a default storage ( [local SSDs](https://cloud.google.com/dataproc/docs/concepts/compute/dataproc-local-ssds)-Persistent disks).
* Intermediate shuffles are stored in the persistent disks.
* Once the VM's removed, the persistent disk(hdfs) will be dropped as well.**VM Boot disks are deleted when the cluster is deleted.**
* If you really wanted a persistent store, then the data needs to be transferred cloud storage/bigquery.

<https://cloud.google.com/dataproc/docs/concepts/dataproc-hdfs>

**How can I get data in and out of a cluster?**

Dataproc utilizes the Hadoop Distributed File System (HDFS) for storage. Additionally, Dataproc automatically installs the HDFS-compatible Google Cloud Storage connector, which enables the use of Cloud Storage in parallel with HDFS. Data can be moved in and out of a cluster through upload/download to HDFS or Cloud Storage.

**Can I use Cloud Storage with Dataproc?**

Yes, Dataproc clusters automatically install the Cloud Storage connector. There are a number of benefits to choosing Cloud Storage over traditional HDFS including data persistence, reliability, and performance.

**How is Dataproc billed?**

Dataproc is billed by the second, and is based on the size of a cluster and the length of time the cluster is operational. In computing the cluster component of the fee, Dataproc charges a flat fee based on the number of virtual CPUs (vCPUs) in a cluster. This flat fee is the same regardless of the machine type or size of the Compute Engine resources used.

**What is Storage transfer service and when to use it ?**

<https://cloud.google.com/storage-transfer-service>

**Storage Transfer Service for cloud data -**

Transfer data from online sources like AWS S3 and Azure Blob Storage to [Cloud Storage](https://cloud.google.com/storage) in one simple process.

**Storage Transfer Service for on-premises data -**

Transfer petabytes of data from on-premises sources to [Cloud Storage](https://cloud.google.com/storage) over online networks.

**What is Transfer appliances and when to use it ?**

<https://cloud.netapp.com/blog/gcp-cvo-blg-google-cloud-migration-tools-copying-1gb-or-500tb-learn-how>

Google transfer appliances (offline storage)

Google provides the Transfer Appliance, a secure, high capacity storage server that is shipped from Google to your datacenter. You can fill it with data and ship it back to an ingest location, for it to be uploaded to Google Cloud Storage. The appliance comes in two sizes—100TB and 480TB, and encrypts data automatically to ensure data is secure.

Here is the process for transferring huge volumes of data in bulk using the Transfer Appliance:

* **Request one or more appliances** by submitting a request online.
* **Receive the appliance**—it is shipped by deliver carriers in a tamper-resistant package.
* **Transfer data to the appliance**—install it in your data center and capture data, the appliance will automatically encrypt ingested data.
* S**hip the appliance back to Google**—use the provided return label to send the device back to a Google ingestion center.
* **Google uploads your data to Cloud Storage**—you will be notified when it is ready.
* **Access your data**—log into Google Console to decrypt the data and choose a storage bucket to load it into.
* **Google securely erases the appliance**—Google follows the NIST-800-88 standar

**What is Big query Transfer services ?**

**Cloud Workflows -**

Orchestrate and automate Google Cloud and HTTP-based API services with serverless workflows.

* Orchestrate Google Cloud and HTTP-based API services into serverless workflows.
* Automate complex processes.
* Fully managed service requires no infrastructure or capacity planning
* Fast scalability supports scaling down to zero and pay-per-use pricing model

**Cloud Composers -**

**Difference between Cloud Composers and Cloud Workflows ?**

To oversimplify, if you want to manage your data processing, ETL or machine learning pipelines and integrate with data products like BigQuery or Dataflow — Cloud Composer is the way to go. However, if you want to process events or chain APIs in a serverless way, with busty traffic patterns, high execution volumes or low latency, you likely need to look at Workflows first (..) Workflows scales out automatically with no "cold start" effect and with a fast transition between the steps. This makes Workflows a good fit for latency-sensitive applications.

**Cloud Spanner vs Cloud SQL**

**Spanner is globally and horizontally scalable**

If you need horizontal scaling or a globally available system the choice is simple — use Spanner since Cloud SQL can not meet those requirements.

**Use Spanner if you have a lot of data**

Cloud SQL supports regional-level databases of up to 30 TB. If you have more data to store don’t consider using Cloud Cloud SQL.

**Use Spanner if you have more than 1 database**

You are limited with having everything on one server with Cloud SQL

**Cloud SQL is cheaper**

If you have just a bit of data to be stored and you don’t need a globally available or horizontally scalable system, watch out for overpayments. Cloud SQL should satisfy you if the above is correct.

The Battles Within

As we can see by simply reading the standard descriptions of the above databases, there is a good amount of overlap in the use cases these databases serve. Let’s review 5 cases that are not easily answered even by Google Cloud’s [Choosing a Storage Option](https://cloud.google.com/storage-options/) flowchart.

Cloud SQL vs. Cloud Spanner

Need for horizontal write scalability either in the same region or across multiple regions is the single driver for [choosing Cloud Spanner over Cloud SQL](https://www.quora.com/Whats-the-difference-between-Google-Cloud-Spanner-and-Cloud-SQL). But as we know in real-world cases, we can never predict when exactly we need to horizontal scale. Whenever such a situation arises, the application has to be re-written and the database has to be migrated from Cloud SQL to Cloud Spanner. Both these tasks can turn out to be painful in practice.

Cloud Spanner vs. Cloud BigTable

Google’s recommendation is to pick BigTable for single-region analytics use cases and Spanner for multi-region operational use cases. This distinction is also very hard to implement in practice. A time series app for a custom DevOps monitoring use case looks very similar to the early days of a SaaS infrastructure monitoring platform built by New Relic and Datadog. The latter use case requires multi-region scalability and strongly consistent secondary indexes. Going with BigTable on day 1 will not help such a use case but that’s what most users will pick by default.

Cloud Spanner vs. Cloud Datastore

The big distinction here is the nature of the data stored — Spanner is good for structured data that should be stored in relational tables while Datastore is good for unstructured data that should be stored in JSON documents. Another subtle but important distinction is that Spanner is fully ACID compliant whereas Datastore has only Atomic & Durable transactions (which means theres no Consistency and Isolation). What if we need to store the user reviews (unstructured data) along side the item details (structured data) in a product catalog? We need to use two databases in the Google Cloud world.

Cloud BigTable vs. Cloud Datastore

BigTable is optimized for high volumes of data and analytics while Datastore is optimized to serve high-value transactional data to applications. So even though both of them are NoSQL databases, issues similar to what we previously discussed in Cloud Spanner vs. Cloud BigTable arise.

Cloud Memorystore vs. Cloud BigTable

Both BigTable and Memorystore can be used to model key-value datasets, but Memorystore is in-memory only while BigTable has disk-based persistence. Using Memorystore for completely low latency caching situations make sense, but things get murky for use cases such as gaming leaderboards (where cache-to-DB consistency starts to matter) and stream processing (where data volume can get too high to fit into memory). In these cases, BigTable is a better option.