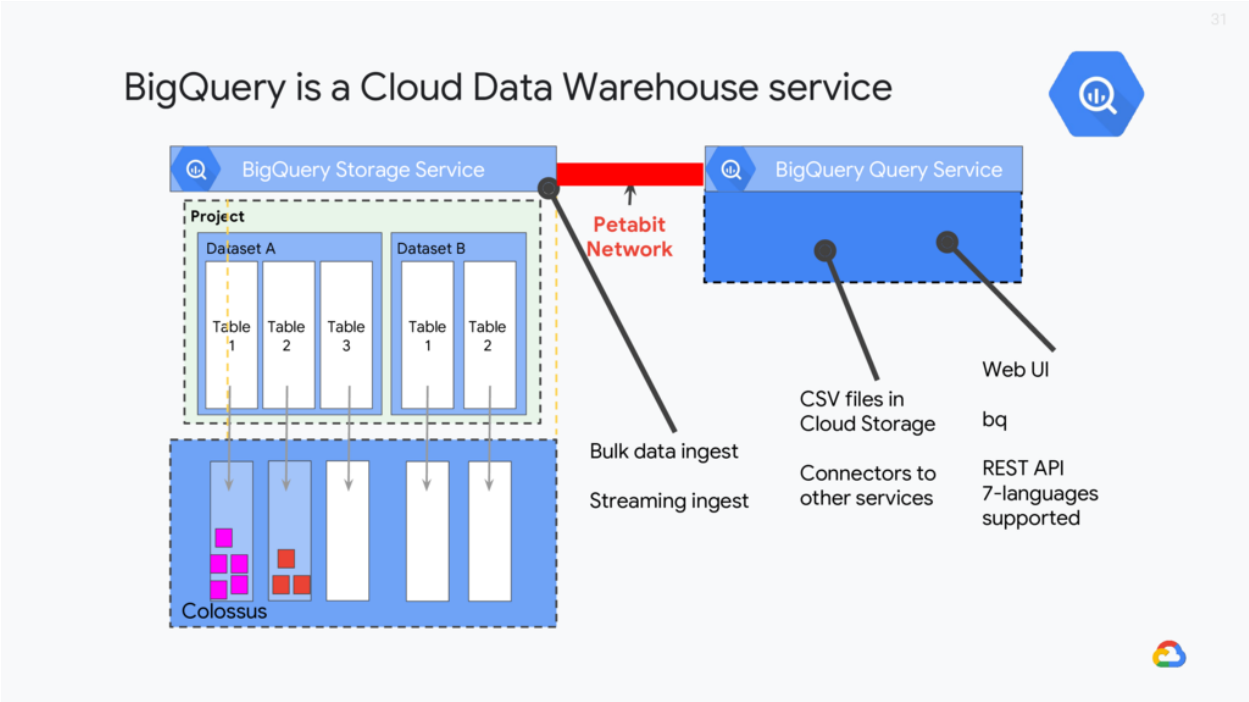
BigQuery



BigQuery is a Data Warehouse service. It is a "serverless" service, meaning that it is fully managed. So users do not have visibility or control over individual servers or clusters of servers.
BigQuery runs data processing jobs that can load, export, copy or query data.

BigQuery has two parts, a storage service and a query service, which work together. They are connected by Google's high speed internal network.

The storage service manages the data.
Data is contained within a project in datasets in tables. The tables are stored as highly compressed columns in Google's Colossus file system which provides durability and availability.

BigQuery Storage Service automatically shards and shuffles data in the underlying file system to provide a very high level of service at huge scales. The sharding occurs automatically and provides the advantages of data distribution while completely concealed from you at the Dataset and Table level.

The storage service supports bulk data ingest and streaming ingest. So it can work with huge amounts of data and also real-time data streams.

Thq query service runs interactive or batch queries that are submitted through console, the BigQuery web UI, the bq command line tool, or via REST API. The REST API is supported for seven programming languages.

There are connectors to other services such as Cloud Dataproc which simplify creating complex workflows between BigQuery and other GCP data processing services.

The query service can also run query jobs on data contained in other locations, such as tables in CSV files hosted in Cloud Storage.

BigQuery is most efficient when working with data contained in its own storage service.
The storage service and the query service work together to internally organize the data to make queries efficient over huge datasets of Terabytes and Petabytes in size.

The most important control over resource consumption and costs is writing a query that controls the amount of data processed. In general, this is done with SELECT by choosing subsets of data at the start of a job rather than by using LIMIT which only omits data from the final results at the end of a job.