Google Cloud for Data Engineers

Welcome!

Varoon Sahgal
Instructor, Pluralsight

About Me:

- Worked in industry for 20 years
- Training for the last 7 years
- Love tennis

Prerequisites

This course assumes you

- Have some very basic cloud knowledge
- Have some experience as a data engineer

We teach over 400 technology topics.

3 day Agenda

Daily Breakdown

- Day 1:
  - Cloud Terminology: IaaS, PaaS, etc
  - Google Cloud Basics: Regions, Zones, IAM, etc..
  - Compute Engine
  - Google Cloud Storage (GCS)

- Day 2:
  - Cloud SQL
  - Cloud Pub/Sub
  - Dataflow
  - Datastore
  - Machine Learning APIs - Vision, Natural Language

- Day 3:
  - Auto ML Vision
  - Dataproc
  - MapReduce
  - Dataflow
  - Big Query
  - Data engineering with GCP

Class Logistics

8 hours

1 hour lunch + breaks every hour

Class Logistics

Eliminate distractions

Enable video

Introductions

Introductions

- Please introduce yourself
  - Name
  - Role at S&P, Location
  - Experience with any cloud provider? (not required)
  - Fun fact about yourself

Infrastructure Options

Before we get into GCP

- Let’s ensure that we are on the same page cloud terminology wise
- And that we understand all the tradeoffs between public/private and IaaS/PaaS, etc..

Infrastructure Options

Infrastructure is the hardware & software that run our IT workloads and that provide our business users and customers a way to interface with the applications required to complete their daily jobs

What Are the Options?

On-Premise (in a Data Center)

Public Cloud

At the Edge

Hybrid Cloud

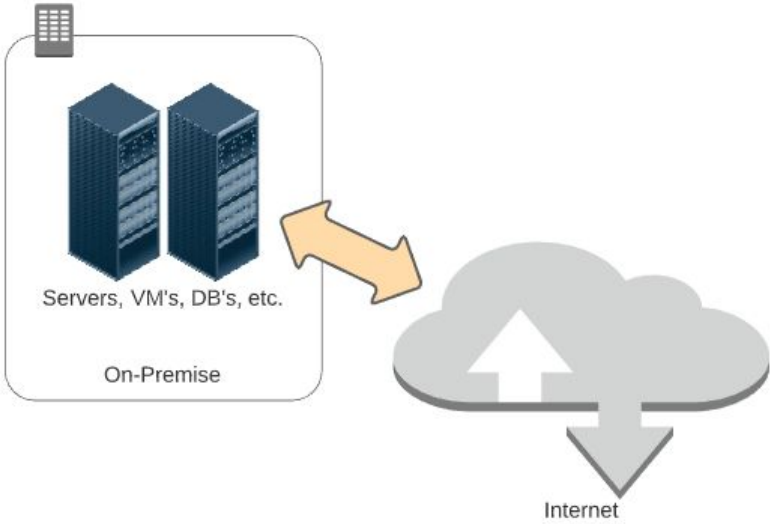
What do they all mean?

On-Premise

Can mean a few different things:

- In a wholly-owned Data Center
- In a COLO (or co-location Data Center)
- Sometimes called a “private cloud”

On-Premise



On-Premise

Why and What?

- How infrastructure has traditionally been done
- With this model, companies try and estimate current & future hardware capacity needed to support business operations

On-Premise

Why and What?

- Stakeholders plan out expected levels of consumption for the next 3 – 5 years (capacity to handle current volumes as well as expected growth)
- Some critical workloads may not be suitable for anything but a physical and directly-managed implementation (e.g., mainframe)

On-Premise – Discussion

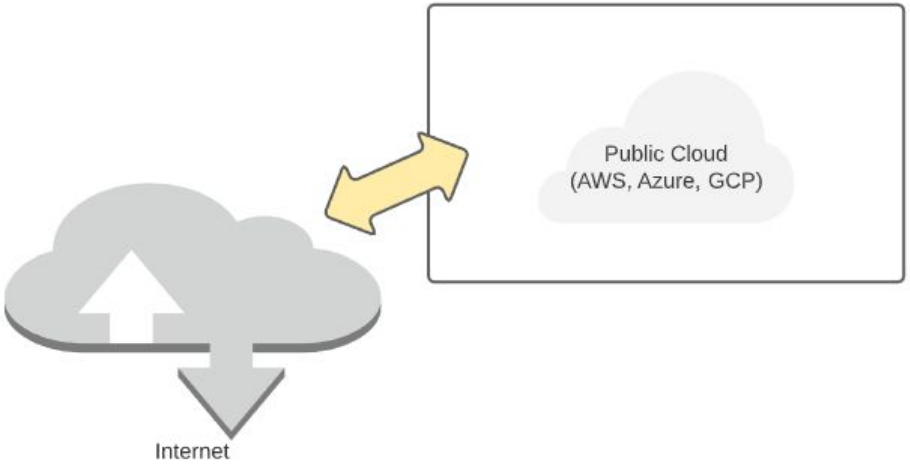
Pros?

- Discrete capacity planning (even if that planning was off)
- Some workloads (e.g., mainframes and certain legacy systems) are tailor-made for a physical data center
- With a move to COLO's, companies could begin to share expenditure

Cons?

- Sometimes difficult to know what is needed and when it is needed – if the plan was off (or unexpected spikes in demand occurred), difficult to adjust quickly
- Some workloads are just as effective (if not more so) in a virtual vs. physical implementation
- Harder to control costs and plan for costs – CAPEX (on-premise - buying servers) vs. OPEX(use public cloud - like GCP - renting servers)

Public Cloud



Public Cloud

Why and What?

- Platform using the standard “Cloud computing model” to provide infrastructure and application services
- Accessed and integrated via the Internet
- May provide a few different types of services – IaaS, PaaS, etc.

Public Cloud

Why and What?

- Usually supports a subscription or “pay as you go” (on-demand) pricing model
- Largest players in this space include Azure, AWS and GCP

Public Cloud - Discussion

Pros?

- Flexibility and elasticity in capacity planning – enables automated schedule-based or metrics-based adjustments to capacity when required
- In some cases, managed services can be leveraged reducing operations overhead
- Because services are PAYG (pay as you go), you're only charged for what you use, and those expenses are OPEX

Cons?

- Requires enough historical data for schedule-based planning or the right configuration for metrics-based planning
- With managed services you lose some levels of granular control
- Because of the flexibility/elasticity, it can be difficult to budget and, if Cloud services are not managed/monitored, costs can be high

At the Edge

Why and What?

- It’s about bringing the power of Cloud computing to you
- Enables additional processing closer to the sources of data while still supporting the offload of higher order processing to the Cloud
- Often involves setting up “Cloud-in-a-box” facilities on-premise

At the Edge

Why and What?

- IoT (Internet of Things) is a good example – devices in a facility reading massive amounts of data can incorporate processing at the edge to improve overall efficiency

- Helps inject lower latency, increased security and improved bandwidth into systems used to aggregate critical data for an enterprise

At the Edge - Discussion

Pros?

- Allows distribution of processing power across a larger surface area
- Can be used to bring critical latency, security and bandwidth improvements to specific types of business workflows
- Efficiencies gained “at the edge” can help with managing the cost of processing data

Cons?

- Requires more infrastructure and more configuration to support that distribution
- Increased distribution of processing power and activity can expand attack surface and requires the right configuration to ensure optimal interaction between system components (i.e., increased complexity)
- More components “at the edge” can lead to increased infrastructure costs

Hybrid Cloud

Why and What?

- In many ways, an amalgamation of the other options
- Supports distribution of system processing across on-premise infrastructure and the public Cloud

Hybrid Cloud

Why and What?

- Allows an enterprise to keep workloads/software applications that are best-suited for on-premise running on-premise while allowing migration of components that can move to the public Cloud

- Can help make an enterprise’s move to the Cloud more gradual and planful

Hybrid Cloud - Discussion

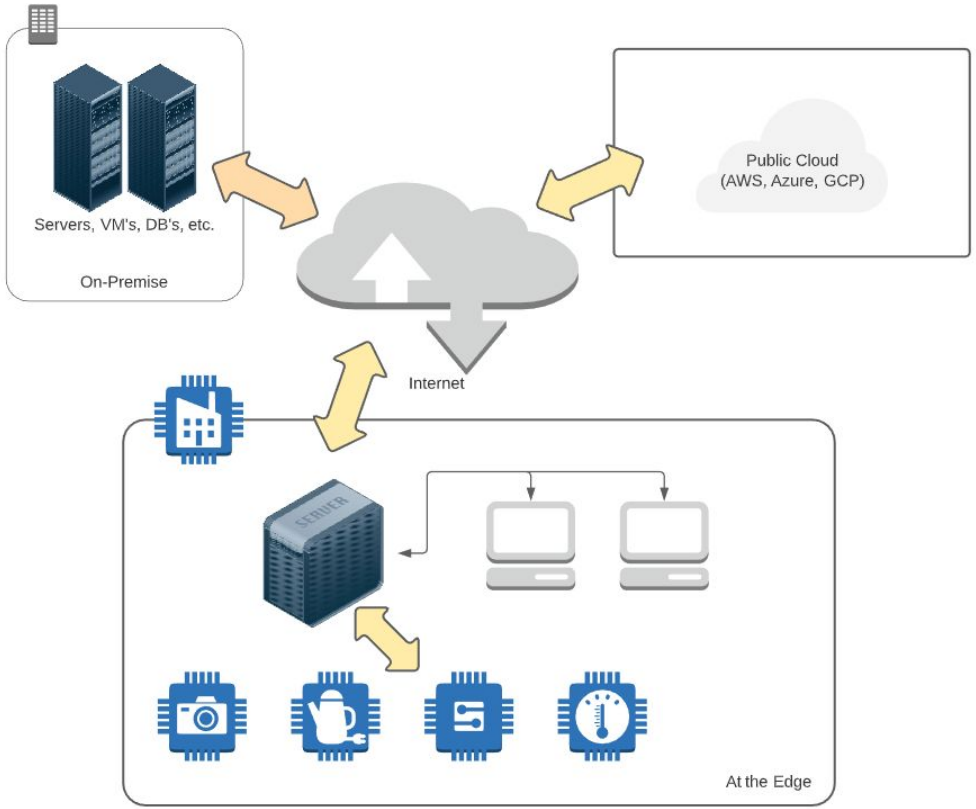
Pros?

- Allows distribution of processing power across a larger surface area
- Can allow a move to the Cloud to be more gradual and allow an enterprise to target optimal deployment platform while making the move
- The ability to support a gradual move enables an enterprise to assess and understand Cloud costs over time

Cons?

- Requires more infrastructure and more configuration to support that distribution
- As with Edge, can lead to increased complexity, often including required setup and maintenance of dedicated, secure connectivity between a data center and the Cloud
- If not managed optimally, costs can be higher due to need to pay for Cloud usage and data center (CAPEX + OPEX)

Hybrid Cloud



Application Hosting

Application Hosting

By Application Hosting, we mean the target infrastructure and runtime platform used for deployment and execution of an application or system; can include compute (CPU and server resources), storage, network, data and operating system

What Are the Hosting Options with Cloud?

- IaaS
- PaaS
- Serverless / FaaS
- SaaS
- Containers

What do they all mean?

Infrastructure-as-a-Service (IaaS)

- Involves the building out (and management) of virtual instances of:
  - Compute
  - Network
  - Storage

- Akin to spinning up a server (physical or virtual) in your location or data center complete with disks and required network connectivity

Infrastructure-as-a-Service (IaaS)

- The difference is in the where – instead of in your data center, it is created in a data center managed by one of the public Cloud providers

- Your organization is responsible for patching the OS, ensuring all appropriate security updates are applied and that the right controls are in place to govern interaction between this set of components and other infrastructure

Platform-as-a-Service (PaaS)

- Involves leveraging managed services from a public Cloud provider
- With this model, an enterprise can focus on management of their application and data vs. focusing on management of the underlying infrastructure
- Patching and security of the infrastructure used to back the managed services falls to the CSP (Cloud Service Provider)

Platform-as-a-Service (PaaS)

- Many managed services support automatic scale up or down depending on demand to help ensure sufficient capacity is in place
- Can be considered synonymous with the term “Cloud native”

Serverless / Functions-as-a-Service (FaaS)

- Also represents a type of managed service provided by the CSP
- Cost structure is usually consumption-based (i.e., you only pay for what you use)
- Supports many different coding paradigms (C#/.NET, NodeJS, Python, etc.)

Serverless / Functions-as-a-Service (FaaS)

- Typically, with Serverless (and PaaS), the consumer is only concerned with the application code and data – elements of the CSP's "backbone" used to support are managed by the CSP

- Includes more sophisticated automated scaling capabilities – built for Internet scale

Software-as-a-Service (SaaS)

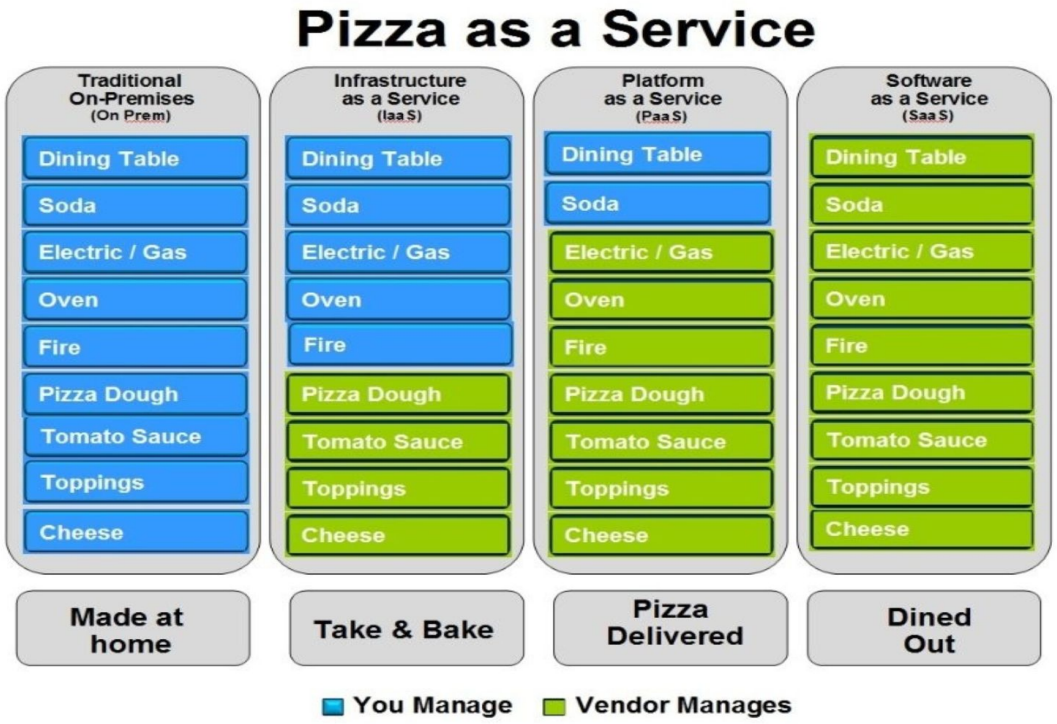
- Subscription-based application services
- Licensed for utilization over the Internet / online rather than for download and install on a server or client machine
- Fully-hosted and fully-managed by a 3rd party

Software-as-a-Service (SaaS)

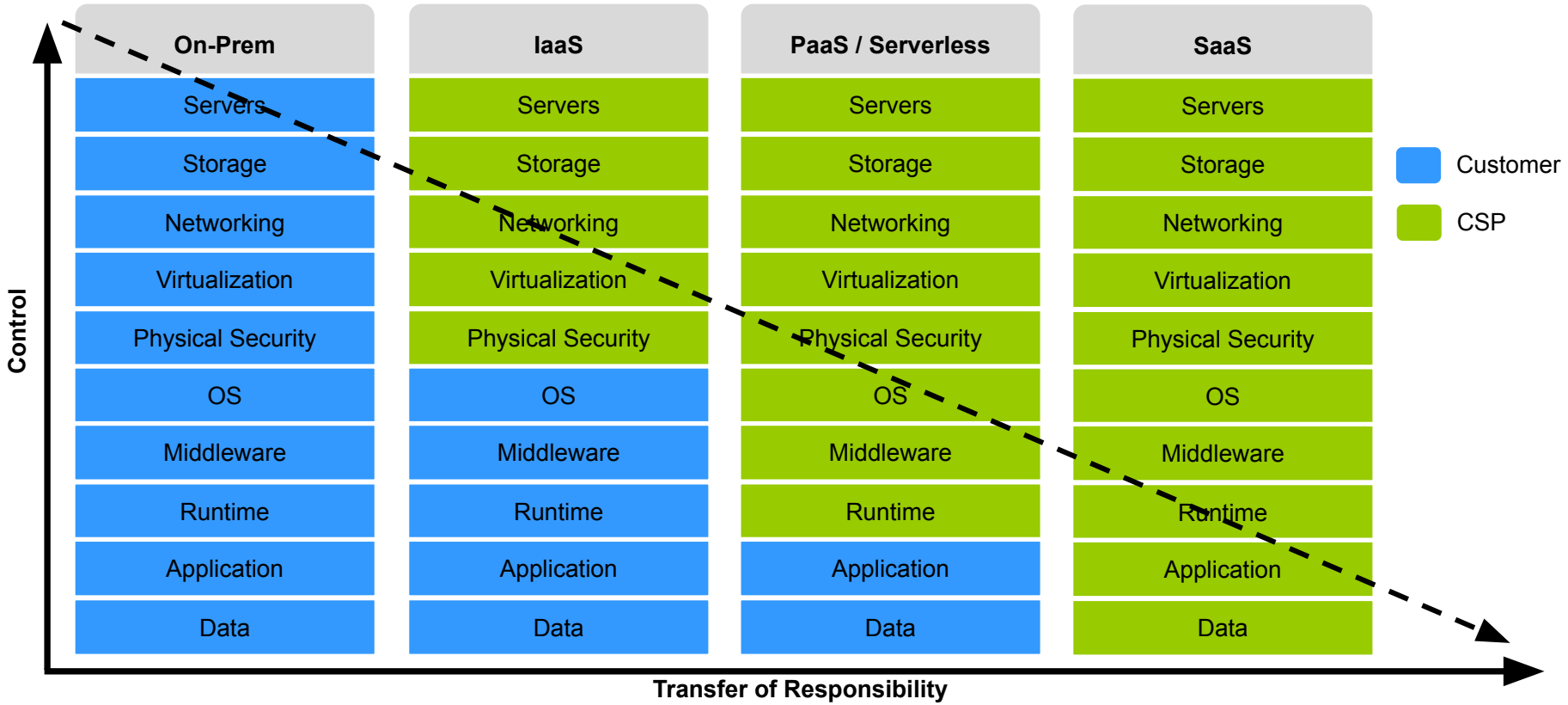
- Of those discussed, often the cheapest option for service consumers
- However, also offers minimal (or no) control, outside of exposed configuration capabilities

Pizza-as-a-Service

From a LinkedIn post by Albert Barron from IBM
(https://www.linkedin.com/pulse/20140730172610-9679881-pizza-as-a-service/**)**



Side-by-Side Comparison



Google Cloud Basics and History

GCP Intro - brief history

- Google Cloud Platform - suite of cloud computing services provided by Google of course
- GCP started by releasing App Engine in 2008
  - Which is a PaaS environment to build and deploy web applications without the need to manage infrastructure/servers
  - AWS equivalent is Elastic Beanstalk

GCP Intro - brief history

- 2010 - Google introduced Google Cloud Storage
  - Provides scalable and durable object storage
  - AWS equivalent == s3 (simple storage solution)
- 2012 - Google introduced Google Compute Engine
  - The first IaaS offering from GCP
  - Compute Engine == virtual machines in the cloud
  - AWS equivalent == EC2 (Elastic Cloud Compute) released in 2006

GCP Intro - brief history

- 2014 - Google introduced data + analytics services
  - Cloud Dataflow (which we will explore in more depth)
    - Allows for processing of both batch and streaming data
- 2015 - Cloud Pub/Sub
  - Messaging service
  - Asynchronous communication
  - Highly scalable
- Also in 2015- ML/AI services were released

GCP Regions

- Where do Google's servers physically live?
- In regions!
- Region == cluster of data centers

GCP Regions

- Regions are physical geographic areas that consist of multiple data centers within a specific geographical location.
- Each region is independent and isolated, offering high availability and fault tolerance.
- Examples of GCP regions:
  - us-central1: Located in Iowa, United States.
  - europe-west1: Located in Belgium.
- Let’s take a look at this in the Google Console

GCP Zones

- Regions are further subdivided into zones
- Zones are individual deployment areas within a region.
  - Most regions have 3 zones
- Each zone is isolated from failures in other zones, **providing redundancy and resilience.**
  - That is why we use zones!
- Zones are typically connected through high-bandwidth, low-latency networks.
- Examples of GCP zones:
  - us-central1-a: Zone A within the us-central1 region.
  - europe-west1-b: Zone B within the europe-west1 region.

How do we choose regions and zones?

When choosing regions and zones for your GCP resources, consider factors such as:

- Proximity to users: Select regions closest to your target audience to reduce latency.
- Compliance requirements: Ensure data sovereignty and compliance with regional regulations.
- Service availability: Some GCP services may have specific regional availability limitations.
- High availability and disaster recovery: Distribute resources across multiple zones to ensure redundancy and minimize downtime.

IAM - Identity and Access Management

- IAM is Google Cloud Platform's centralized access control system that manages user and resource permissions.
- IAM is VERY important - you will without a doubt encounter it in some fashion
- IAM allows you to control who has access to your resources and what actions they can perform.
  - ***Basically who/what can do what within your GCP account ?***
  - Want to follow **least privilege principle** - give users/services the least amount of access required to do what they need to
- All cloud providers have IAM - AWS, Azure, GCP
  - It’s essential to everything

IAM - good analogy

IAM - good analogy

- IAM is a bit like a hotel key card which gives you access to some areas of the hotel (like your room, the pool, gym etc)
    - But it denies you access to other areas like the kitchen, meeting rooms, other guest rooms, etc
- GCP and it's associated services is the "hotel" in this analogy

IAM - Identity and Access Management

- IAM is a global service

IAM in GCP - key terminology

- IAM Key Terminology:
  - Projects: The organizational unit for managing and organizing resources.
  - Members: Individuals or groups who can have specific roles and permissions.
  - Roles: Collections of permissions that define what actions can be performed.
  - Service Accounts: Special accounts used by applications and services to authenticate with GCP APIs.

IAM Roles

- IAM provides a wide range of predefined roles with different levels of permissions, such as Owner, Editor, and Viewer.
- Custom roles can also be created to grant granular access control tailored to your specific requirements.
- Example: Creating a custom role for a team of developers that allows them to create and manage Compute Engine instances but restricts access to other resources.

IAM Policies

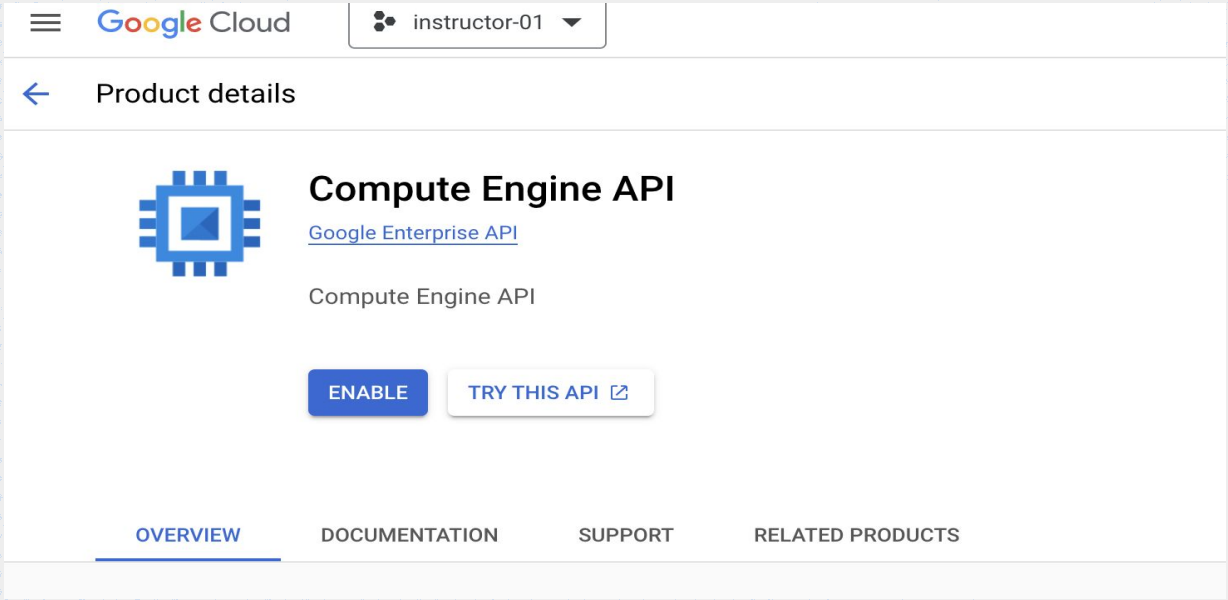
- IAM policies define who has what permissions on which resources.
- Policies are associated with resources (e.g., projects, folders, and individual resources) and determine access control at the granular level.
- Example: Assigning IAM policies to a bucket in Cloud Storage, specifying which users or groups can read or write objects in the bucket.

IAM Best Practices

- Follow the principle of least privilege: Grant only the necessary permissions to users, groups, or service accounts.
- Regularly review and audit IAM policies to ensure they align with your organization's requirements.
- Use service accounts for application authentication and access management.
- Implement multi-factor authentication (MFA) for added security.

Enabling API's in GCP

- You will notice that before interacting with a service we typically have to enable the API
  a. Why do we have to do that?



Enabling API's in GCP

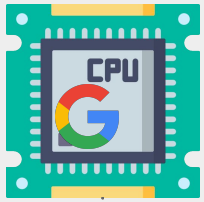
- There's a few reasons:
- **Security:** Enabling APIs inherently introduces potential security risks. By not enabling APIs by default, GCP ensures that users have control over which APIs are enabled and can manage the associated security implications.
- **Cost Management**: Enabling APIs can result in additional costs, especially for services that are billed based on API usage. By not enabling APIs by default, GCP helps users avoid unintended costs that may arise from inadvertently enabling APIs that they do not require.
- **Resource Utilization:** Enabling APIs consumes system resources, such as CPU, memory, and network bandwidth. By not enabling APIs by default, GCP minimizes resource consumption until explicitly requested by users. This approach helps ensure optimal resource allocation and avoids unnecessary resource utilization for services that may not be relevant to every user.
-

Agenda

- Virtualized Infrastructure
- Cloud Shell
- Persistent vs. Transient Storage
- Computer Engine User Interface
- Pre-emptible Instances

Compute Engine

Compute on GCP



Google Kubernetes Engine (GKE)

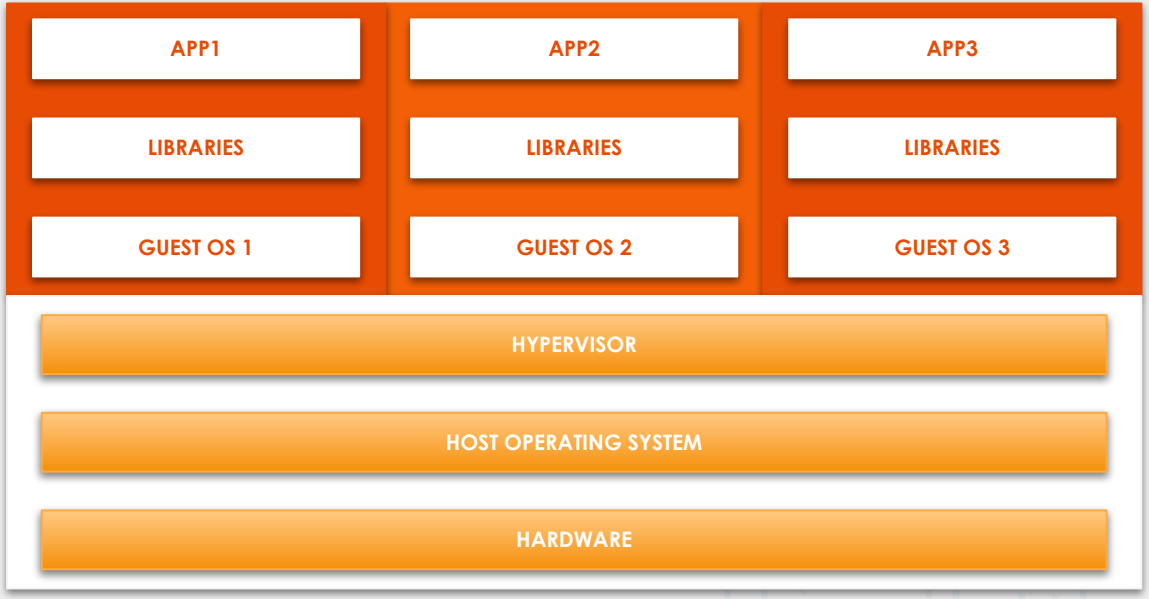
Google Compute Engine (GCE)

Google App Engine (GAE)

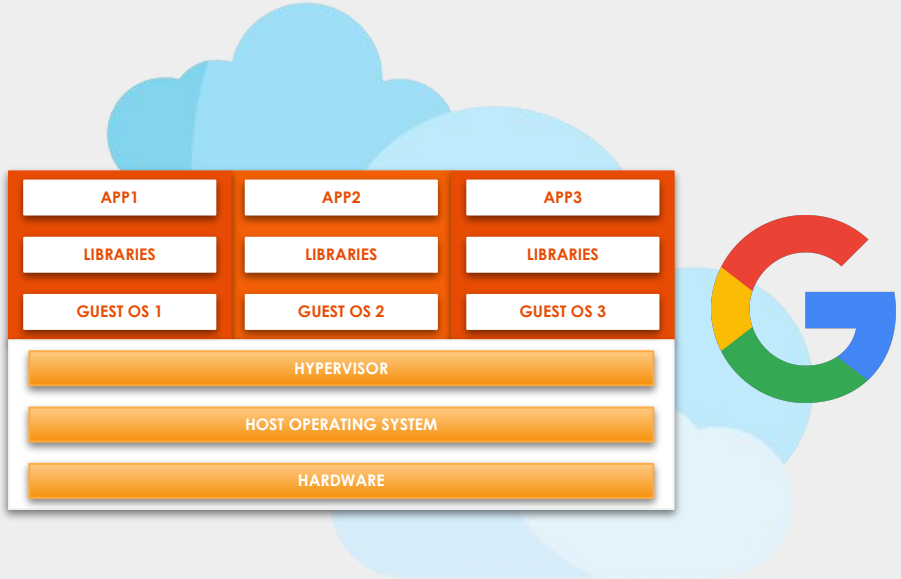
Google Ancillary Cloud Functions

Highly Customizable / Highly Managed

Virtualized Infrastructure

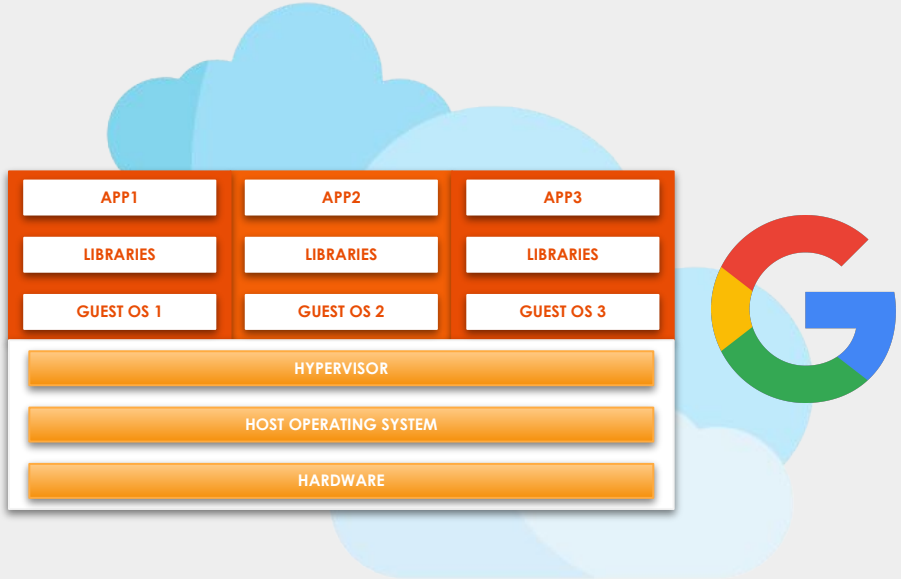


Google Compute Engine == GCE



Secure and customizable compute service that lets you create and run virtual machines on Google's infrastructure

GCE == IaaS



GCE is a service that gives us IaaS (Infra as a service) on the cloud - it's the lowest level of abstraction - and gives us a machine that we can

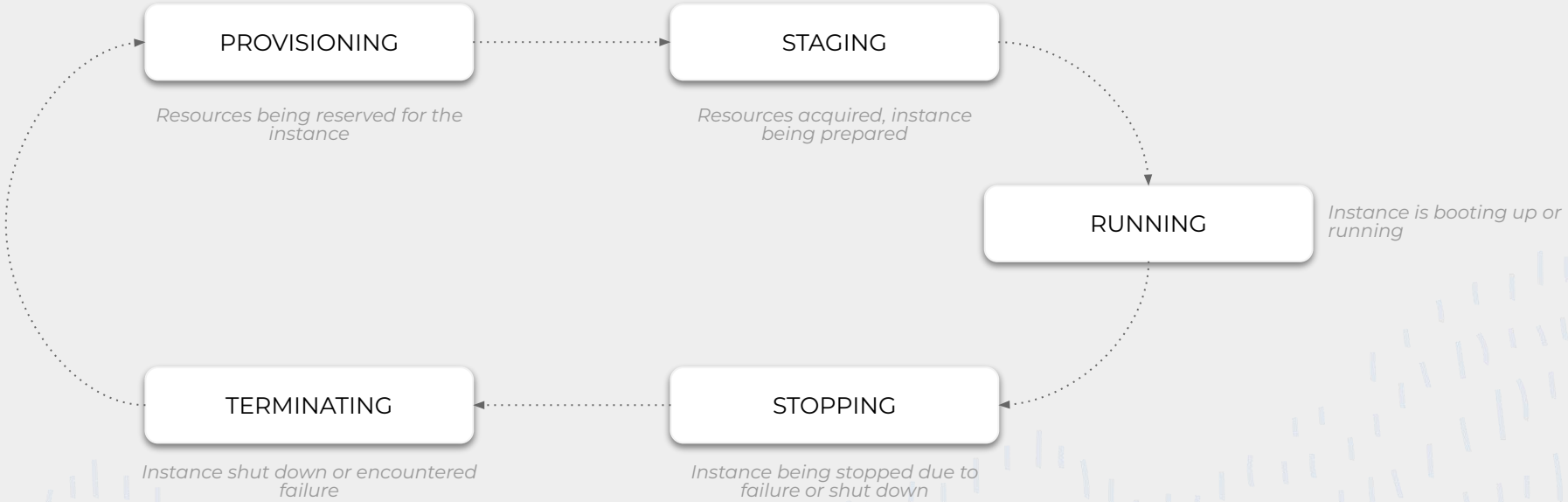
Compute Engine

- **Predefined machine types:** Start running quickly with pre-built and ready-to-go configurations
- **Custom machine types:** Create VMs with optimal amounts of vCPU and memory, while balancing cost
- **Spot machines:** Reduce computing costs by up to 91%
- **Confidential computing:** Encrypt your most sensitive data while it's being processed
- **Rightsizing recommendations:** Optimize resource utilization with automatic recommendations

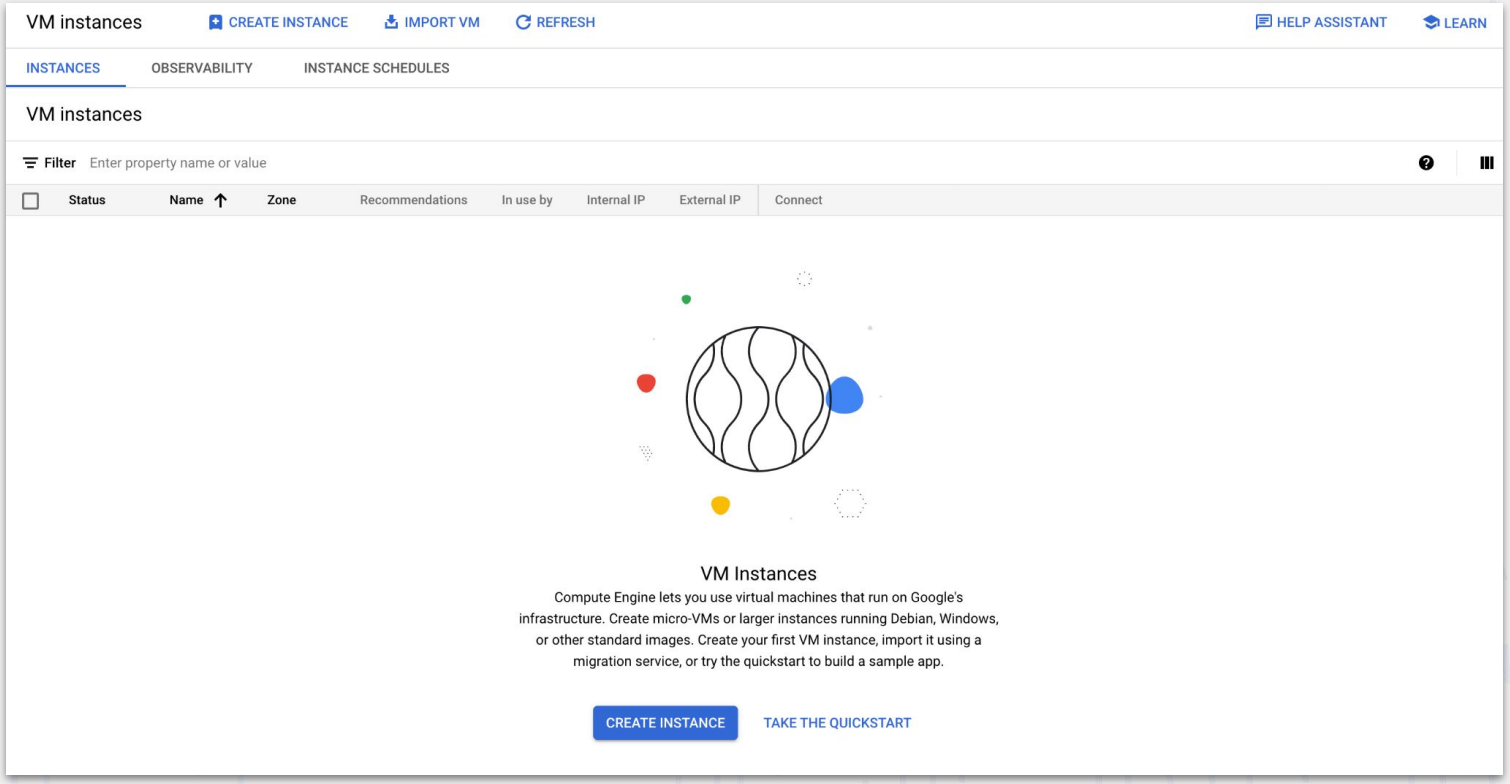
Choosing right machine type

Scale Out Workload	General Purpose	Ultra-High Memory	Compute Intensive	Most Demanding Applications
Tau VMs are the lowest cost solution for scale-out workloads on Compute Engine, with up to 42% higher price-performance compared to general-purpose VMs of any of the leading public cloud vendors	*C3, E2, N2, N2D, and N1 are general-purpose machines offering a good balance of price and performance, and are suitable for a wide variety of common workloads including databases, development and testing environments, web applications, and mobile gaming*	*Memory-optimized machines offer the highest memory configurations with up to 12 TB for a single instance. They are well suited to memory-intensive workloads such as large in-memory databases like SAP HANA, and in-memory data analytics workloads*	*Compute-optimized machines provide the highest performance per core on Compute Engine and are optimized for workloads such as high performance computing (HPC), game servers, and latency-sensitive API serving*	*Accelerator-optimized machines are based on the NVIDIA Ampere A100 Tensor Core GPU. Each A100 GPU offers up to 20x the compute performance compared to the previous generation GPU*

VM Life Cycle



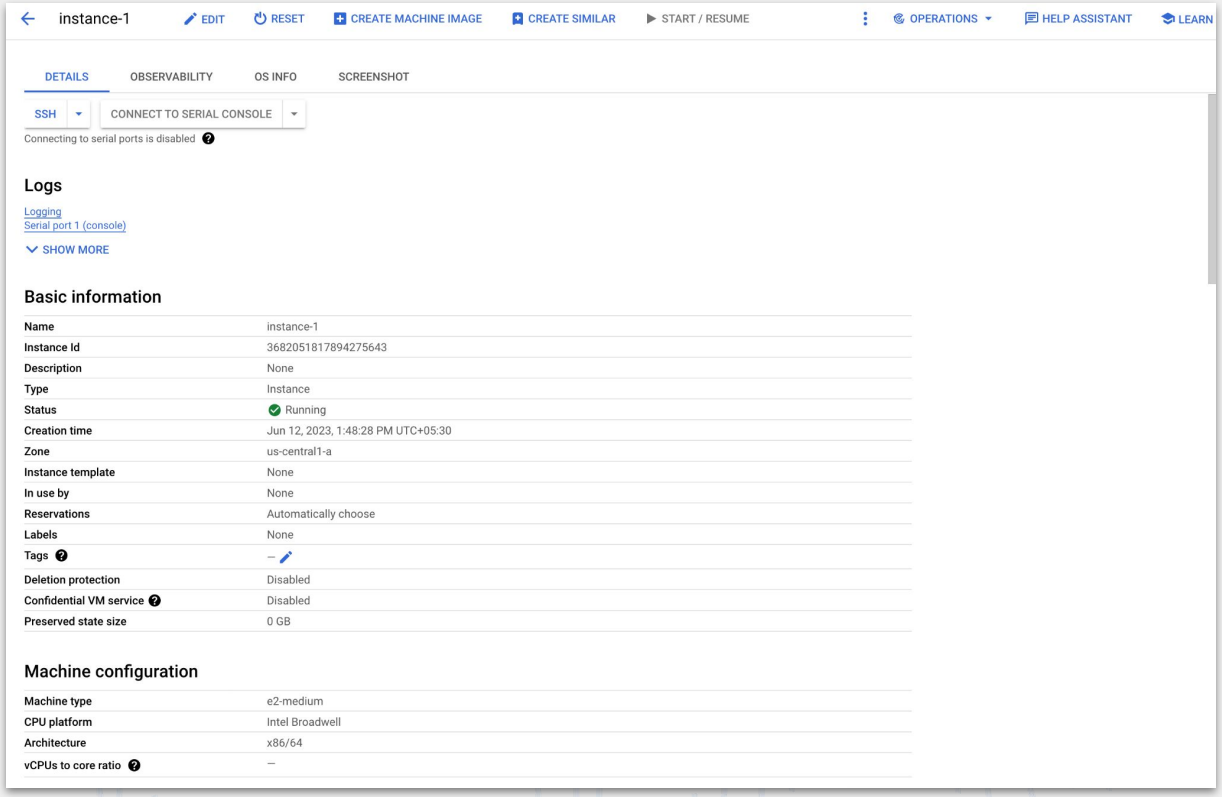
Compute Engine UI



Creating Compute Engine Instance

DEMO

Inside Compute Engine Instance



Preemptible Instances

- Preemptible VM instances are available at much lower price—a 60-91% discount—compared to the price of standard VMs
    - Equivalent in AWS == EC2 Spot Instnaces
- However, Compute Engine might stop (preempt) these instances if it needs to reclaim the compute capacity for allocation to other VMs
- Preemptible instances use excess Compute Engine capacity, so their availability varies with usage
- If your apps are fault-tolerant and can withstand possible instance preemptions, then preemptible instances can reduce your Compute Engine costs significantly

Preemptible Instance Limitations

Preemptible instances function like normal instances but have the following limitations:

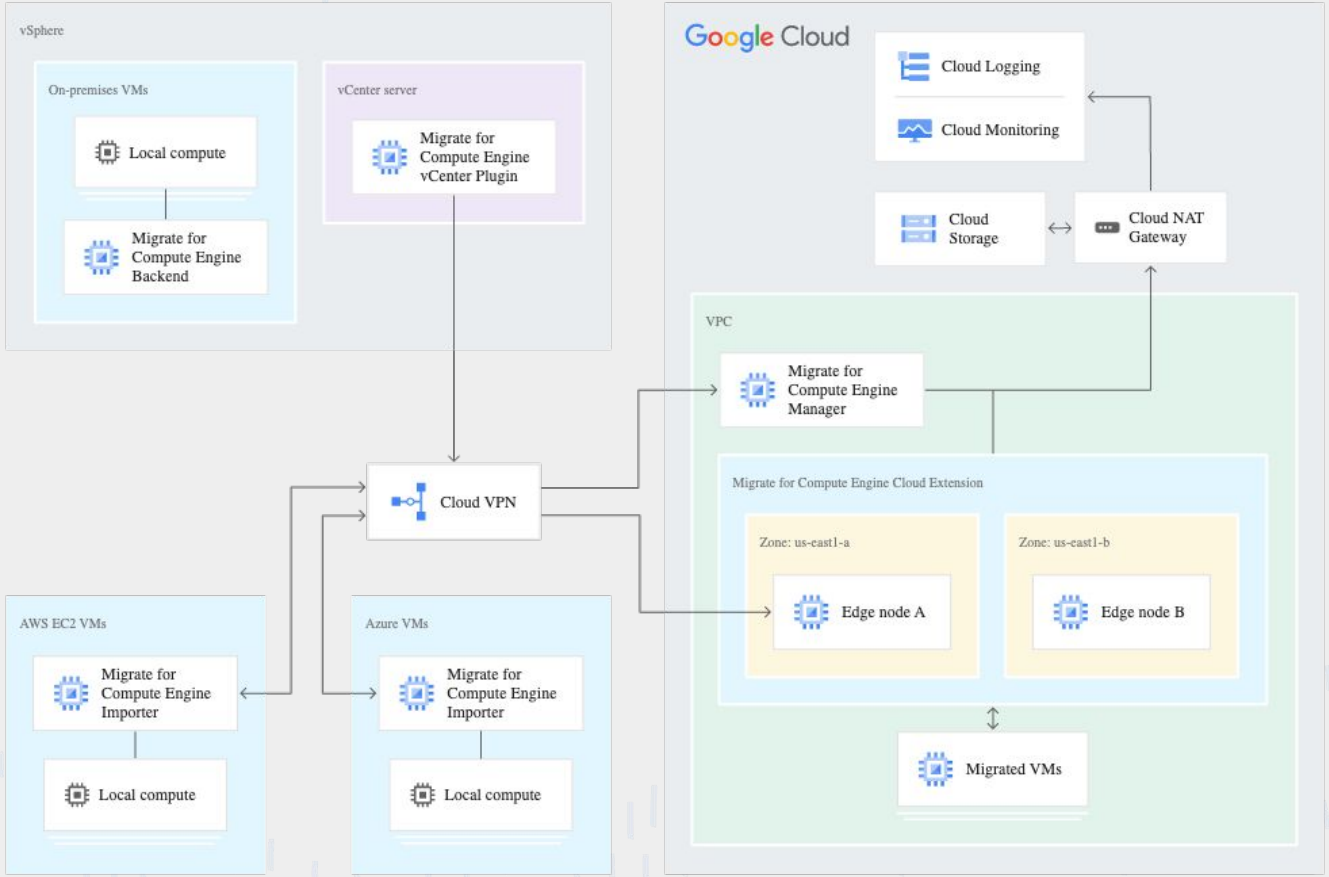
- Compute Engine might stop preemptible instances at any time due to system events
- Compute Engine always stops preemptible instances after they run for 24 hours
- Preemptible instances are finite Compute Engine resources, so they might not always be available
- Preemptible instances can't live migrate to a regular VM instance, or be set to automatically restart when there is a maintenance event
- Due to the preceding limitations, preemptible instances are not covered by any Service Level Agreement and are excluded from the Compute Engine SLA
- The Google Cloud Free Tier credits for Compute Engine do not apply to preemptible instances

Preemption of Preemptible Instance

Compute Engine performs the following steps to preempt an instance:

1. Compute Engine sends a preemption notice to the instance in the form of an **ACPI G2 Soft Off signal**. You can use a shutdown script to handle the preemption notice and complete cleanup actions before the instance stops
2. If the instance does not stop after 30 seconds, Compute Engine sends an **ACPI G3 Mechanical Off signal** to the operating system
3. Compute Engine transitions the instance to a **TERMINATED** state

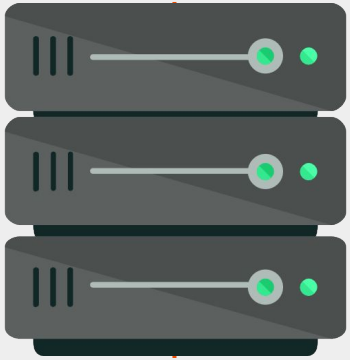
VM Migration to Compute Engine



Compute Engine Pricing

- **Per-minute billing**, sustained use discounts
- Preemptible instances
  - Live at most 24 hours
  - Can be pre-empted with a 30 second notification via API
  - Up to **80% discount**
- Custom machine types
  - Customize amount of memory and CPU
- Recommendation Engine
  - Notifies you of under utilized instance

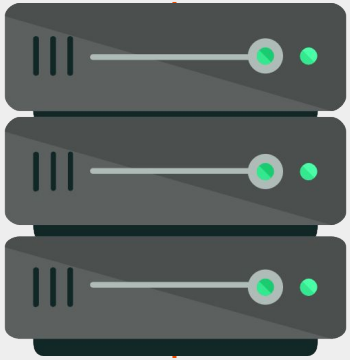
Charges for VM



- All machines are billed for a minimum of 10 minutes
- Per-minute charge, rounded up to nearest minute
- Lower price for preemptible instances
- Scaled discounts for sustained use

% of usage	Percent of base rate
First 25%	100%
Next 25%	80%
Last 25%	60%

GCE - Compute Engine Labs



- Getting started with Compute engine Lab

  (PDF sent over Zoom chat)

- https://github.com/varoonsahgal/sp-gcp-data/wiki/Connect-Compute-Engine-to-Cloud-SQL

Integration with GCP

Integration with Google Cloud services and tools

Product	Links
App Engine	Use Cloud Storage with App Engine.
BigQuery	Query, import, and export data: • Performing federated queries on data stored in a Cloud Storage bucket • Loading data into BigQuery • Exporting data from BigQuery • Transferring data to BigQuery • Loading data from Cloud Storage to BigQuery using Workflows
Cloud CDN	Use Google's global edge network to serve content closer to users, which accelerates your websites and applications: • Setting up Cloud CDN with a backend bucket

Integration with Google Cloud services and tools

Product	Links
Cloud Data Loss Prevention	Use Cloud DLP to detect and classify sensitive data stored in Cloud Storage objects: • Inspect a Cloud Storage location
Cloud Functions	Cloud Functions trigger lightweight, stand-alone functions in response to events in Cloud Storage: • Cloud Functions tutorial using Cloud Storage
Cloud Life Sciences	Use a Cloud Storage bucket to import your gene sequence variations data to a BigQuery dataset: • Storing and loading genomic variants
Cloud Load Balancing	Make a Cloud Storage bucket a load balancer backend: • Add a Cloud Storage bucket to an external HTTP(S) load balancer • Enable Cloud CDN on a load balancer with a Cloud Storage backend bucket

Integration with Google Cloud services and tools

Product	Links
Cloud Logging	Store logs from applications and services on the Google Cloud to Cloud Storage: • Exporting your Compute Engine logs and App Engine logs to a Cloud Storage bucket
Cloud SQL	Import and export data using Cloud Storage: • Importing and exporting data
Compute Engine	Use Cloud Storage from within a Compute Engine instance: • Using Service Accounts with Applications • Exporting an image to Cloud Storage • Using a startup script stored in Cloud Storage • Mount a bucket as a file system on a virtual machine instance
Config Connector	Configure your Cloud Storage buckets using Kubernetes tooling and APIs: • Explore the storageBucket schema.

Integration with Google Cloud services and tools

Product	Links
Dataflow	Perform data processing tasks of any size and use a Cloud Storage bucket to hold staging files and temporary data: • Getting Started
Development Tools	Add extensions to your development tools for fast access to your Cloud Storage buckets and objects within them: • Browsing objects in Cloud Storage with Cloud Code for IntelliJ
Error Reporting	Identify and understand your Cloud Storage errors: • View errors from the Google Cloud console • Send error notifications to selected Cloud Monitoring notification channels
Firebase SDKs	Securely and easily access your mobile and web app data using Firebase SDKs for Cloud Storage: • Get started on Android • Get started on iOS • Get started on Web

Integration with Google Cloud services and tools

Product	Links
Hadoop on Google Cloud	Run MapReduce jobs directly on data in Cloud Storage: • Cloud Storage Connector for Hadoop Overview
Media CDN	Use Google's modern and extensible content delivery platform to deliver exceptional, planet-scale experiences to your users: • Set up Media CDN with a Cloud Storage bucket
Pub/Sub	Pub/Sub Notifications allow you to track changes to your Cloud Storage objects: • Pub/Sub Notifications for Cloud Storage
Storage Transfer Service	Transfer data between Cloud Storage and other storage providers or filesystems: • Transfer data between Cloud Storage buckets • Transfer data to and from Cloud Storage • Use event-driven transfers to keep a Cloud Storage bucket in sync with an Amazon S3 or Cloud Storage source.

Integration with Google Cloud services and tools

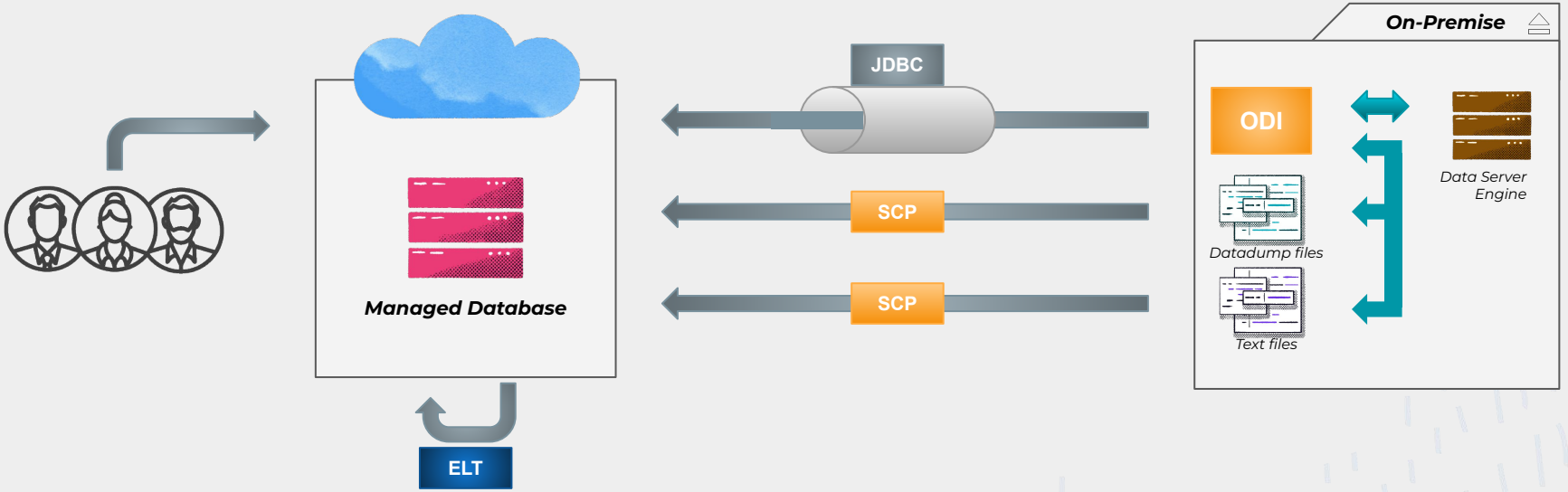
Product	Links
Vertex AI	Store your Vertex AI model data and output in Cloud Storage: • Create a dataset • Import models to Vertex AI • Get batch predictions
VPC Service Controls	Create perimeters to protect the resources and data of services that you specify, such as Cloud Storage: • Create a service perimeter that can protect buckets in the project you specify • Cloud Storage-specific considerations when using VPC Service Controls

Cloud SQL

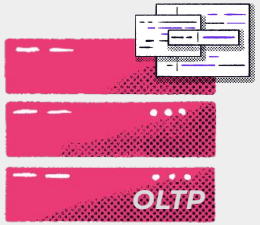
Agenda

- Provisioning Managed Database Infrastructure
- Configuration of MySQL on GCP
- Batch Data Import/Export with Cloud SQL
- Web-based Interface
- Integration of MySQL with GCP Services and Applications

Provisioning Managed Database Infrastructure



Transactional vs Analytical Workloads



Transactional

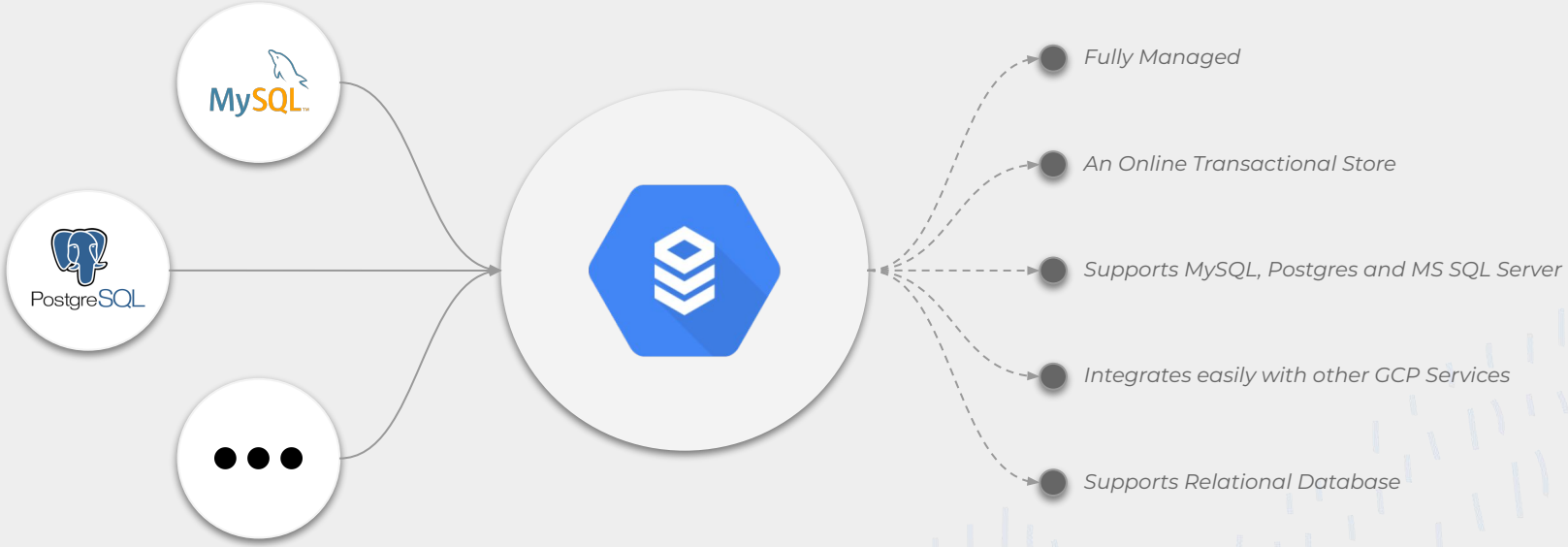
- *High volume of Transactions*
- *Fast Processing*
- *Normalized Data*
- *Many Tables*
- *“Who bought X?”*

Analytics & Reporting

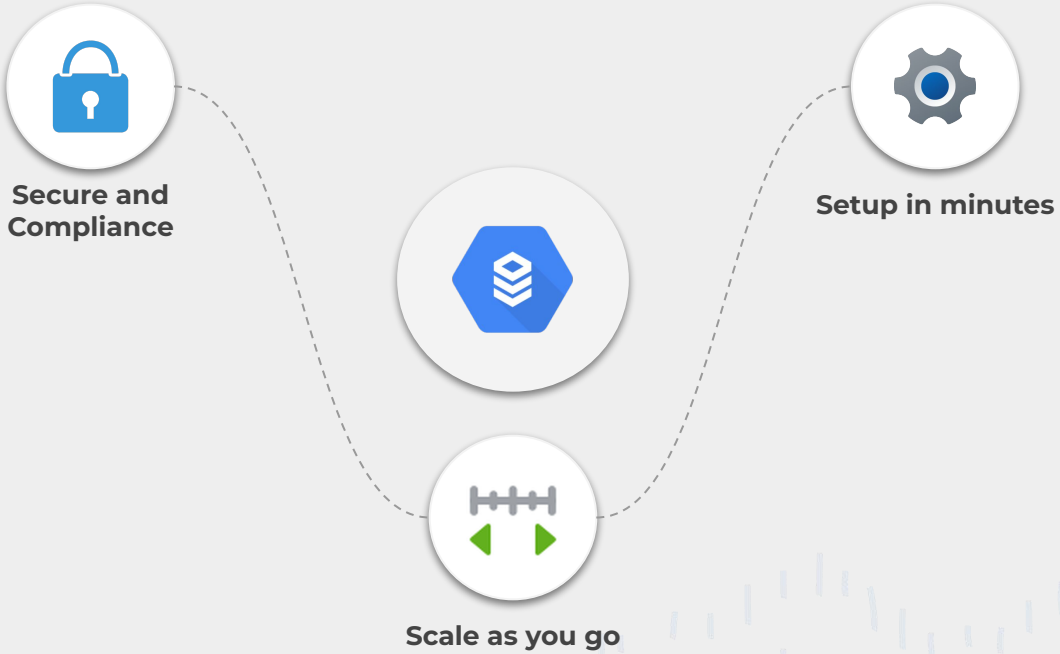
- *High volume of Data*
- *Slow Queries*
- *Denormalized Data*
- *Fewer Tables*
- *“How many people bought X?”*

Configuration of MySQL on GCP

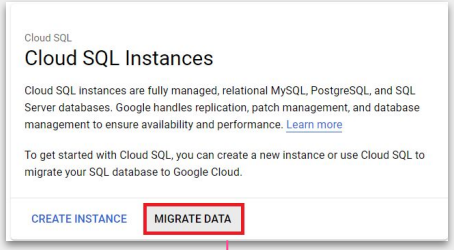
Configuration of MySQL on GCP

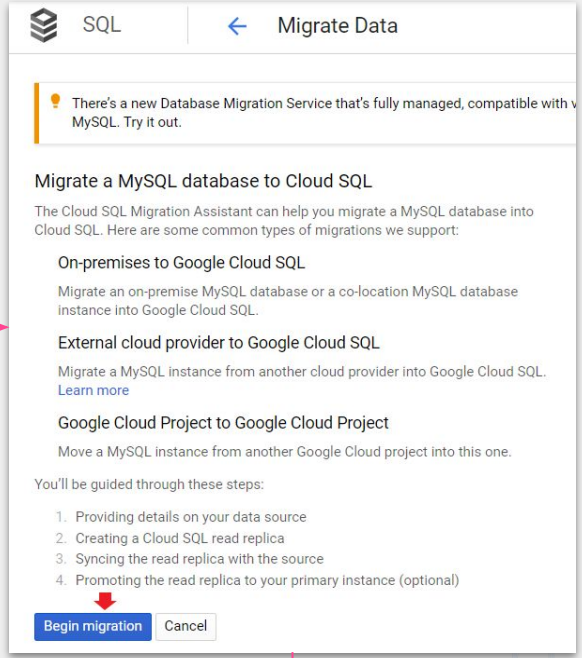


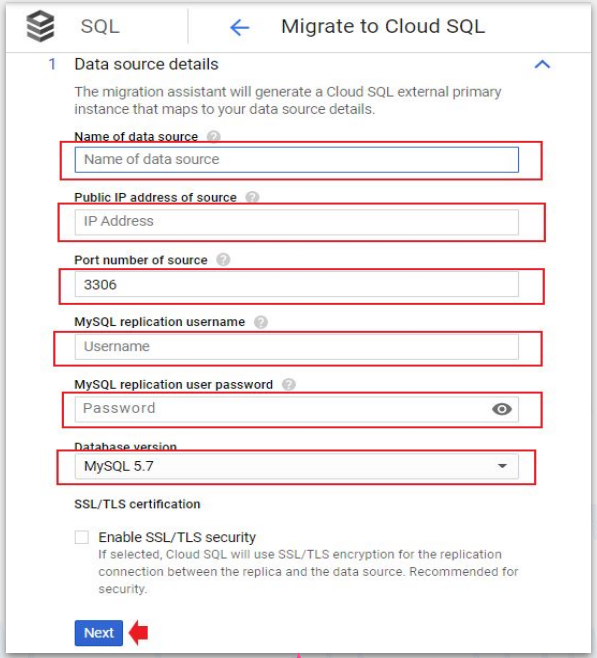
Configuration of MySQL on GCP



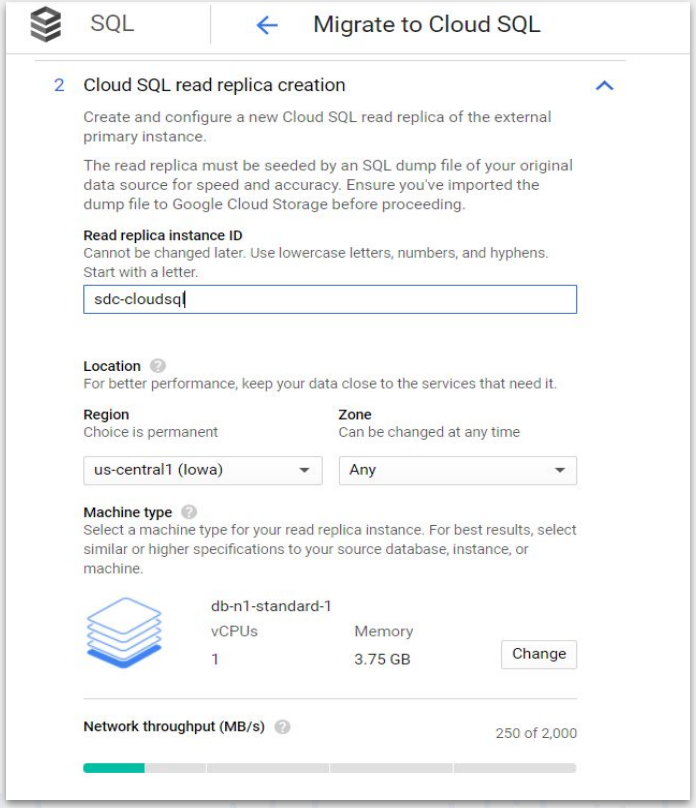
Batch Data Import/Export in Cloud SQL





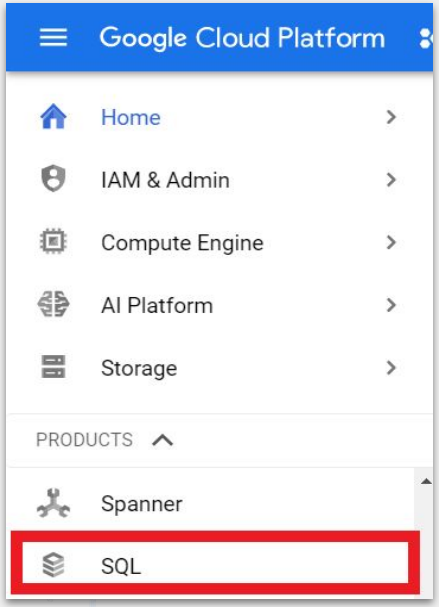


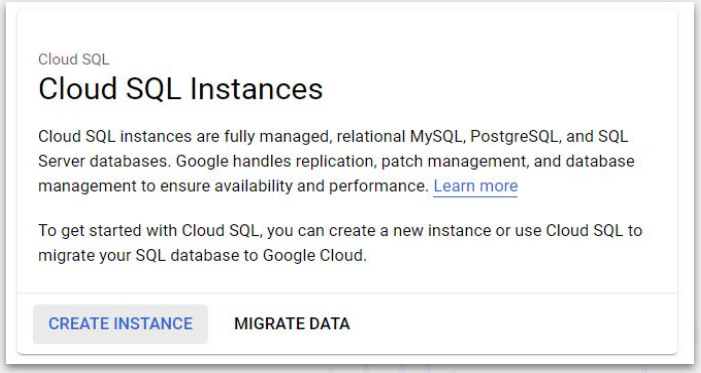
Batch Data Import/Export in Cloud SQL



Web-based Interface

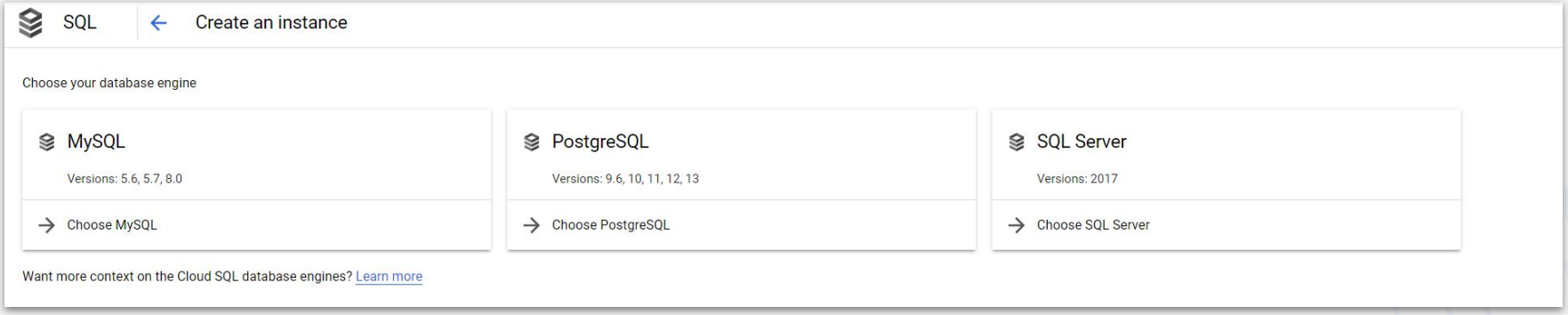
Log In to cloud.google.com and move to SQL from navigation bar



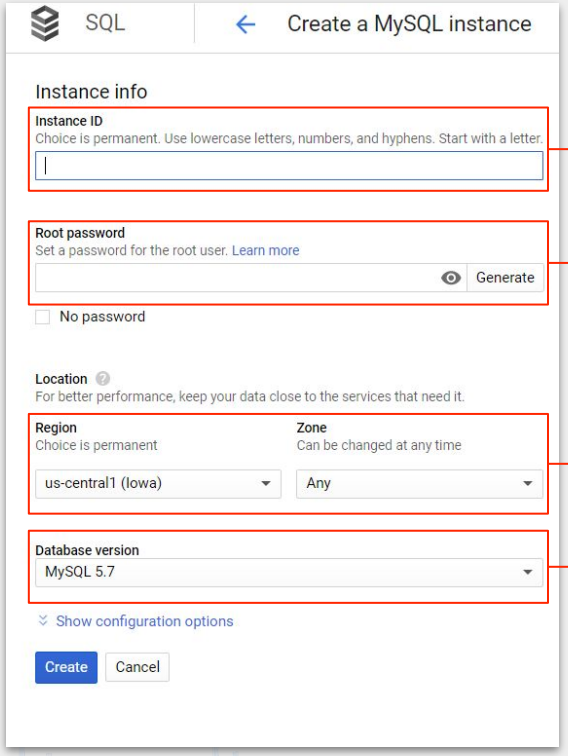


Web-based Interface

For creating CloudSQL Instances, we have three options i.e., *MySQL, PostgreSQL, SQL Server*



Creating MySQL Instances



Give your instance a name

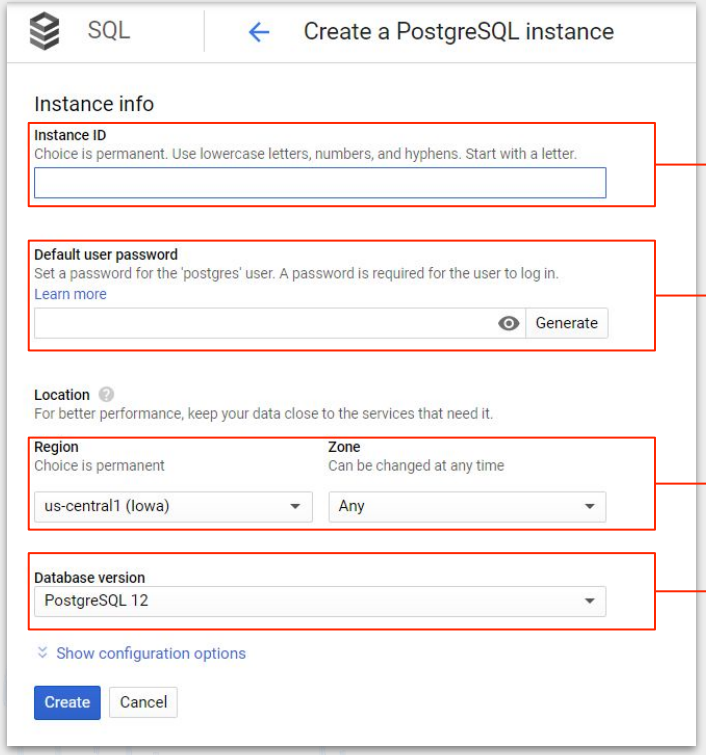
Create a password for securing your database

Select the desired region and zone

Select the desired version of MySQL server

Click on CREATE for creating the MySQL instance

Creating PostreSQL Instances



Give your instance a name

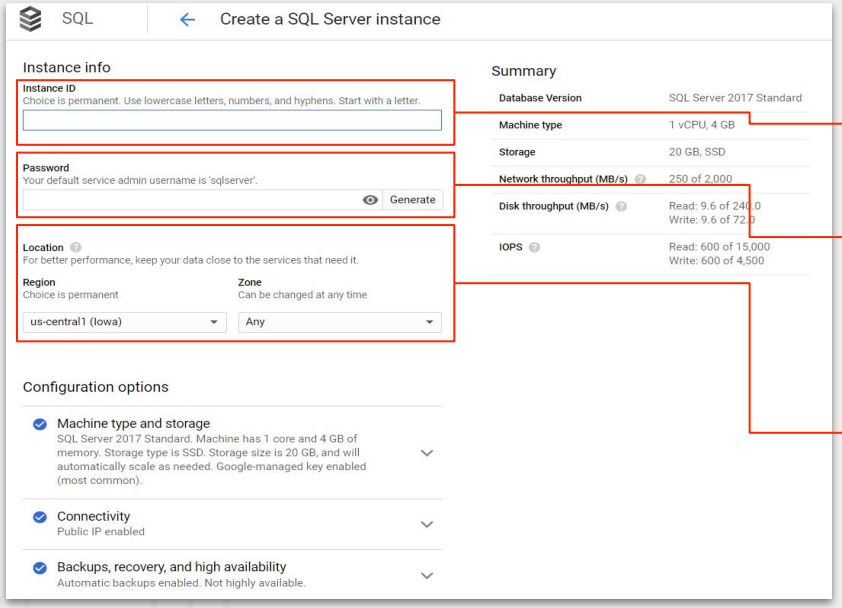
Create a password for securing your database

Select the desired region and zone

Select the desired version of MySQL server

Click on CREATE for creating the PostgreSQL instance

Creating SQL Server Instances



Give your instance a name

Create a password for securing your database

Select the desired region and zone

*Click on **CREATE** for creating the SQL Server instance*

Integration of MySQL with GCP Services

Google Cloud Storage

Google Cloud Storage

Question: What kind of data can i put in Cloud Storage?

- Answer: Technically any type of data, but typically **unstructured** data
  - Structured data -> database -> use Cloud SQL instead of Cloud Storage
- Examples of data that's good for Cloud Storage: media files - images, web asset libraries, stream videos, data lakes
  - Imagine a site which requires that people upload images frequently - how can they store those images cheaply?
    - Cloud Storage is a great option!
- Also great for archival storage: backups, media archives

Google Cloud Storage

CATEGORY	PRODUCTS	GOOD FOR
Object storage	Cloud Storage Object storage for companies. Store any type of data, any amount of data, and retrieve it as often as you'd like.	Stream videos Image and web asset libraries Data lakes
Archival storage	Cloud Storage Ultra low-cost archival storage with online access speeds.	Backups Media archives Long-tail content Meet regulation or compliance requirements

Google Cloud Storage

Question: Can i pay less for objects that I access less frequently?

- Answer: yes!
- With the use of Storage Classes
- Depending on how often you plan on accessing the data, you would choose a class
- On next slide you can clearly see how it gets cheaper with each storage class
    - more frequently accessed to less frequently accessed

Google Cloud Storage classes

Storage class		
Standard Storage Best for frequently accessed ("hot" data) and/or stored for only brief periods of time.	Starting at $.02 per GiB per month
Nearline Storage Best for service for storing infrequently accessed data.	Starting at $.01 per GiB per month
Coldline Storage Best for storing infrequently accessed data.	Starting at $.004 per GiB per month
Archival Storage Best for data archiving, online backup, and disaster recovery.	Starting at $.0012

Key Features of GCS

- Scalability: GCS can store and manage very large amounts of data, from small files to petabytes of data.
- Durability and Availability: GCS ensures that your data is highly protected and available at all times, even in case of hardware failures or disasters.
  - Data Durability: GCS guarantees that your data will be retained and protected.
    - GCS promises 99.999999999% data durability, which means that the chance of data loss is extremely low (11 9's of durability)
  - Data Availability: GCS ensures that your data can be accessed and retrieved whenever you need it, without any downtime.

GCS: terminology - buckets

- Definition: Buckets are the top-level containers used to organize and store data in GCS.
- Inside of buckets, we store ***objects*** - our actual data
- Characteristics:
  - Each bucket has a globally unique name, ***which must be unique across all GCS buckets.***
  - Buckets can be used to ***group related objects together*** for efficient data management.
- Example: Imagine a bucket named "my-bucket" that contains multiple objects.
- Example: Just like a physical bucket can hold multiple objects, such as books or toys, a GCS bucket can contain multiple objects like images, documents, or videos.

GCS: terminology - objects

- ***Objects*** are the individual pieces of data stored within GCS buckets.
- Characteristics:
  - Objects can be files of any type, such as images, videos, documents, or application data.
  - Each object has a unique key or name within its respective bucket.
  - Objects can range in size from a few bytes to terabytes of data.
- Example: Within the "my-bucket" bucket, we can have objects such as "image.jpg," "document.pdf," and "data.csv."

GCS: terminology - buckets + objects

- Buckets act as the top-level hierarchy, containing objects.
- Objects reside within buckets, representing the individual data files or resources.
- Buckets can be used to group related objects together for efficient data management.

GCS: regional storage

- When you first create a bucket, you permanently set a geographic location for storing your object data
  - You cannot change a bucket's location after it's created, but you can move your data to a bucket in a different location.

- You can select from the following location types:
  - *region* - a specific geographic place, such as São Paulo.
  - *dual-region* - a specific pair of regions, such as Tokyo and Osaka.
  - *multi-region* - a large geographic area, such as the United States, that contains two or more geographic places.

GCS: advanced features

- ***Object Versioning***: Automatically maintain previous versions of objects, allowing you to retrieve and restore earlier versions if needed, such as for document history or code changes.

- ***Lifecycle Management***: Define rules to automatically move data to different storage classes or delete it based on specific criteria, optimizing costs and data management.
  a. Imagine an e-commerce company that stores product images in GCS. With lifecycle management, they can:
     - Define a rule to transition objects older than 30 days to a lower-cost storage class, such as Nearline or Coldline.
     - Set another rule to delete objects older than one year to free up storage space.

GCS: advanced features

- ***Object Change Notification***: Set up notifications to trigger actions or processes when objects are uploaded, modified, or deleted in GCS, allowing you to react to changes in real-time.
  a. Image uploaded to GCS bucket -> invoke a Google Cloud function (AWS Lambda equivalent) -> the function grabs image metadata and stores it inside a database (Cloud SQL maybe)...

- ***Requester Pays***: Enable external users to access specific datasets stored in GCS while covering the data transfer costs themselves.
  a. Allows bucket owners to configure specific buckets to require requesters to cover the data transfer costs.

Google Cloud Pub/Sub

Before Pub/Sub...What is streaming data?

- Data that is generated continuously by thousands of sources
- Typically simultaneously
- And in small size (kilobytes)

What is streaming data - examples

- Examples:
  - Stock trading platforms - track minute changes in the stock market in near-real time and use that info to make a profit
  - Credit card transactions (imagine how much data CC companies must be dealing with!)
  - Traffic analysis - where is traffic backed up? Where is there an accident/etc..(think Waze, Google Maps)
  - Gaming
  - Transportation vehicles often have tracking devices - like those used by Walmart/Target - real time data allows stores to know when to expect deliveries so drivers can spend more time driving trucks instead of waiting at stores
    - Imagine thousands of trucks sending this data simultaneously

Streaming data - more

- Streaming data often referred to as “unbounded data”
  - Whereas **batch** data is often called “bound” data
- Streaming data is constant - always flowing, never “finishes”
  - Think: credit card transactions, Walmart/Target trucks, etc..

Streaming data and telemetry

- Streaming data is commonly used for telemetry
- Where telemetry is the act of collecting data from a large number of geo-dispersed devices as its generated
- So the result of the act of telemetry == streaming data

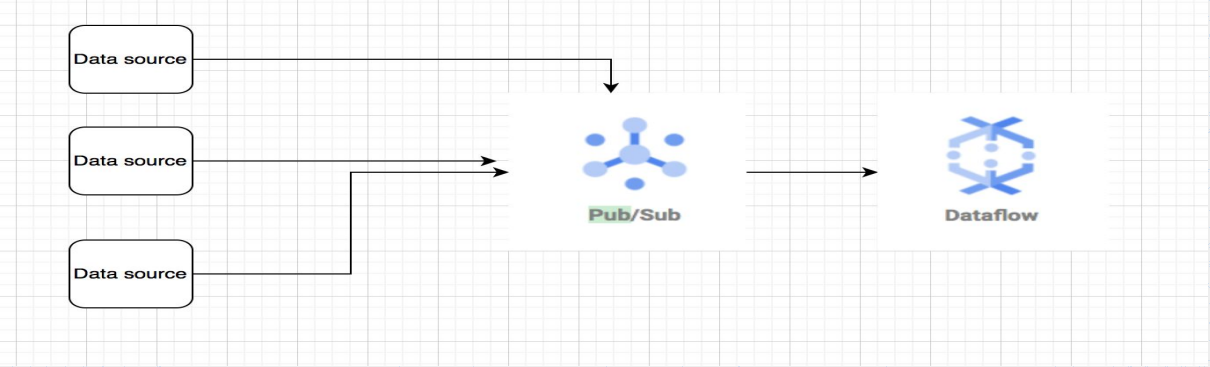
Latency in Streaming data

- When working with streaming data we require a low latency
- Low latency == minimal time delay
- So, we want things to happen fast!

How does streaming data get ingested?

- When working with GCP, how would we ingest streaming data - where does that data go?
  - Ie - what services within GCP would we typically use? And why?
  - Data ingest: pub/sub (SQS/SNS =~ AWS equivalents), data processing: dataflow/dataproc, data analysis: BigQuery
- For instance how would actually collect and analyze the data collected by a number of devices (ie trucks for Walmart or credit card transactions in different places)
- We would typically see the data collected by a number of devices (ie trucks for Walmart or credit card transactions in different places)

How does streaming data get ingested?



- Pub/Sub would first do the massive real time data ingest
- Then the data would be forwarded to Dataflow/Dataproc for the real time processing of that data
- After Dataflow it may be forwarded to a storage (like GCS/Cloud SQL) or analytics service in GCP (BigQuery)

Challenges of streaming data?

- Data that is generated continuously by thousands of sources
- Typically simultaneously
- And in small size (kilobytes)
- Examples:
  - Stock trading platforms
  - Credit card transactions (imagine how much data CC companies must be dealing with)
  - Mobile gaming

Streaming data - fast action

- When working with streaming data, we often need to respond quickly and take action - why?
- Imagine a credit card company (Amex, Visa, etc...)
  - The sooner they detect fraud the better! Prevents even more fraudulent transactions from occurring
  - ***Clearly, doing analysis just once a day for fraud is not sufficient***
- Imagine a mobile maps app (Waze, Google Maps)
  - The sooner they detect traffic the better!
  - The more accurate they are the more users they get
  - Hence the ability to generate more revenue

Streaming data - fast action

- Sometime storage would be skipped to save time
  - This way the data can be sent straight for analysis
- Of course that data must be stored somewhere
  - So, often the service that provides analysis also acts as storage
    - So it may do both **analysis** and **storage**
- Contrast that with pulling the data in Cloud Storage and then moving it to BigQuery
  - That would add additional latency which we really don't want
  - Time is of the essence with streaming data!

Streaming data challenges

- Why would we use pub/sub?
- First, let's look at challenges of a tightly coupled system
- Then, contrast that with a loosely coupled system

Tightly coupled system challenges

- Clearly, our streaming data needs to be sent to some sort of receiver
- Suppose our sender of data is something like a traffic sensor, a sensor in a Walmart truck, or even a credit card reader
- So we could have many senders and a single receiver perhaps
  - If synchronous, what is the problem? - blocking
  - So that's why we do Asynchronous data ingestion
- Then, if there's a direct link between the sender and receiver we would call that a ***tight coupling***

Tightly coupled system challenges

- A tight coupling means that there is no buffer between the sender and receiver
- What we want is sender -> buffer -> receiver
- Problem with sender -> receiver
- While this may sound OK, the problem is this:
- Directly coupling a sender and receiver can be more failure prone - how?
    - ***The receiver could become overloaded resulting in either lost messages or (best-case) a delay in processing of those messages***
        - ***Delays mean that those decisions are no longer real-time***

Tightly coupled system challenges

- Also - if either a sender or a receiver goes down that can result in lost data as well
- Now that we've discussed tightly coupled systems let's compare them to ***loosely*** coupled systems

Loosely coupled systems - pub/sub

- Loosely coupled systems by comparison have an intermediate buffer
- Note the difference - we no longer use the term sender and receiver, but rather publisher/subscriber

- Note that we now have a buffer in between the publisher and subscriber
- Having a buffer gives us **additional fault tolerance**

Loosely coupled systems - buffer

- The buffer can hold messages (CC transaction data) so that communications are not lost!
- With a buffer, if a publisher/subscriber ***goes down or is overloaded***, then we are still in good shape
- Imagine if a subscriber goes down - with a buffer it can still receive the message when it comes back up
- As long as your buffer can also scale automatically to hold more messages you need not worry about losing messages...

Loosely coupled systems - buffer

- Cloud pub/sub provides a message bus/buffer for us
  - Which allows us to build loosely coupled systems
- That is the fault tolerant intermediary between your sender and receivers
- It also ensure that messages are not lost in the process

What is Pub/Sub

- Pub == Publisher, Sub == Subscriber
- Cloud pub/sub provides a message bus/buffer for us
- It plays a big role in streaming data ingest
- Service that provides large scale ingestion of streaming data from anywhere in the world
- Provides a fault tolerant message buffer for loosely coupled systems
    - Those systems can be anything! As we've talked about: from credit card readers, to tracking devices on Target/Walmart trucks...
- It's a no-ops/serverless service : no node/clusters to provision

What is Pub/Sub

- Pub/sub has global availability - so you don't have to choose a region
- You choose a topic and it's automatically globally available
- Allows you to decouple senders and receivers
- Senders are never attached to receivers
- In fact, they don't even know that each other exists
- They both just plug into pub/sub

What is pub/sub

- Like many services on google cloud, there is an open source equivalent
- Kafka == the open source equivalent
- Guarantees as part of it's delivery model at least once delivery of any message that it receives
- AWS equivalent of Pub/Sub is actually 2 separate services (roughly): SNS and SQS

What is pub/sub - asynchronous messaging

- Pub/sub provides asynchronous messaging
  - Async messaging == any number of senders/subscribers combinations
  - You can have one to many, many to one, etc
  - Can have 1,2, ...even millions!

What is pub/sub - terminology

- Pub/sub has comes with its own terminology that we need to understand:
  - We have: Publishers, Subscribers, Topics, Messages, and Message Store
- Publisher == Sender
- Step by step what happens?
- Step 1: You (programmer) create a topic in pub/sub service
  - Then publisher sends messages to that topic
  - Message would have a payload and optional attributes

What is pub/sub - steps

- Step 2: once messages are sent to a topic they are held in a store until they are delivered to a receiver
- If subscriber/receiver is not able to retrieve a message right away the message store can hold message up to a max of 7 days
  - This is built in fault tolerance
- Step 3: pub/sub will forward messages from a topic to all subscribers individually
  - We will talk about push/pull of messages shortly

What is pub/sub - steps

- 4. The subscriber will take the messages from the subscription and once it sees it, sends an acknowledgement
  - If not acknowledged, within a configurable deadline, Pub/Sub will attempt to redeliver it
- 5. After message is acknowledged by subscriber, it's removed from the subscription's queue of messages
  - So message is no longer in pub/sub and lives within the subscriber itself instead

Push and pull in Pub/Sub

- There are 2 ways pub/sub delivers messages to subscribers
- 1. Can be **pushed** by pub/sub to subscribers
  - Pub/Sub -> Subscribers == push
  - As soon as Pub/Sub receives message sends it to subscriber
  - **Push** gives **lower latency** b/c no waiting by subscriber
- 2. Can be **pulled** by subscribers from Pub/Sub
  - Subscribers <- Pub/Sub == pull
  - Subscriber can initiate checking for messages and pull it at some set interval
  - Pull is the **default** option

Push model advantages

- Immediate message delivery: Messages are delivered in real-time or near real-time, ensuring timely updates to subscribers.
- ***Reduced latency***: Subscribers receive messages as soon as they are available, minimizing any delay.
- ***Simplicity for subscribers***: Subscribers only need to establish a connection and wait for incoming messages.

Push model challenges

- Scalability: Publishers need to handle a potentially large number of concurrent connections, which can be resource-intensive.
- Network dependency: Subscribers require a stable network connection to receive messages.
- Load balancing: Distributing messages evenly across multiple subscribers can be challenging.

Pull model

- In the pull model, subscribers actively request messages from publishers.
- Subscribers periodically poll or request messages based on their own availability and pace.
- Publishers respond with available messages when requested.
- This model is suitable for scenarios where subscribers have intermittent or varying availability.

Pull model advantages

- Subscriber control: Subscribers can regulate the rate at which they receive messages, matching their processing capabilities.
- Offline availability: Subscribers can fetch messages even if they were temporarily unavailable.
- Load balancing: Publishers can distribute messages across multiple subscribers based on their pull requests
- Pull is also ideal for large volume of messages - batch delivery

Pull model challenges

- Increased latency: Subscribers may experience delays in receiving messages, depending on their polling frequency.
- Complex coordination: Subscribers need to manage and orchestrate their pull requests efficiently.
- Resource waste: Frequent polling can result in unnecessary network traffic and resource consumption.

To push or to pull? Choosing the right model

- Consider the nature of your application, its requirements, and the trade-offs between real-time delivery and subscriber control.
- Push model is suitable for immediate updates and low-latency scenarios.
- Pull model is suitable for intermittent availability, load balancing, and managing subscriber rate.

Out of order messaging

- One challenge is that sometimes messages may arrive out of order for subscribers
- Why is this a problem? Let's consider an example...

Out of order messaging example

- Let's consider an online shopping system that uses Pub/Sub.
- When you place an order, the system publishes an order event to a Pub/Sub topic.
- Subscribers are responsible for updating inventory and generating invoices based on these order events.
- However, due to delays in message delivery, a subscriber ***might receive an older order event after a newer event***.
- This can cause issues, such as updating inventory incorrectly or generating invoices for orders that haven't been processed yet.
- To handle this, the system can use techniques like timestamps and unique identifiers to order the messages correctly and prevent inconsistencies.

Handling Out of order messaging

- How can we fix this problem of out of order messaging?
- Well, the system can use techniques like **timestamps** and **unique identifiers** to order the messages correctly and prevent inconsistencies.
- Pub/Sub itself recently (late 2020) also added the ability to handle message ordering through the use of a flag
  - This is not enabled by default!

Batch data aka bulk data

- Batch data == opposite of streaming data
- Large sets of data that pool up over time
- As opposed to streaming data, typically data comes from a ***small number*** of sources (usually 1)

Examples of batch data

- Payroll and billing systems that have to be processed weekly or monthly
- Importing a large dataset for Machine Learning analysis
- Importing legacy data into Google Cloud Storage

Batch data - latency and storage

- ***Low*** latency is not as important as it is with streaming data for batch data
- **Storage -** Typically batch data is stored in Cloud Storage, BigQuery, Cloud SQL, etc..

Cloud Dataflow

Cloud Dataflow

- Dataflow == managed Apache Beam
- What problem does it solve?
- Let's talk about a common problem - processing both batch and streaming data
- When and why would we want to do that? Let's consider an example...

Cloud Dataflow - batch + streaming example

- Think about credit card fraud detection
- We would need streaming data which would provide the current transaction - is it fraudulent?
- Imagine: you live in San Francisco, you recently used your CC in SF but 1 hour later there is a charge on your card in NYC
    - Clearly it's fraudulent - you can't be in NYC just 1 hour after SF
- Batch data would be needed for context - the recent transactions that would be compared to the latest/most recent (the one in NYC)
- CC company needs both types of data (batch + streaming) to conclusively say this transaction is most likely fraudulent!

Cloud Dataflow - batch + streaming example

- Well, with Dataflow, we can process both batch and streaming data ***in a single pipeline***
- Prior to Dataflow, 2 separate pipelines were required for this

Cloud Dataflow - data processing challenges

- Suppose an **element** represents a single data input - like a single row in a table
- If we are processing data ***one row at a time*** (one element), from ***a single source*** - that is quite **easy**
  - One row, single source
- If we were to ***combine elements that each have a different format*** - that's a much harder form of data processing
- Finally, if we were to process out of order data from different sources, with streaming data - that's **very hard**

Cloud Dataflow - data processing challenges

- From a GCP perspective how do we solve this challenge:
  - process out of order data from different sources, with streaming data - that's **very hard**
- That is really where Apache Beam + Cloud Dataflow comes
- Apache Beam - helps us **create** unified data processing pipelines
- Cloud Dataflow - helps us **execute** those pipelines
  - Cloud Dataflow is basically managed Apache Beam
  - Gives us auto-scaling

Cloud Dataflow - characteristics

- Auto-scaling - no need to manually provision extra compute + storage
  - it auto-scales **both compute and storage**
- No-Ops/serverless - no clusters or nodes to provision
- Processes both stream and batch workloads

Cloud Dataflow and Apache Beam

- Apache Beam is open source supported by multiple communities beyond GCP
- Beam provides the programming model and SDK for writing portable data processing pipelines
- Dataflow is a managed service that executes those pipelines at scale
- You'll notice that if you go to Dataflow docs in GCP there are links to Apache Beam documentation - https://cloud.google.com/dataflow/docs

Apache Beam

- Apache Beam is 100% code based
- You can define your batch and streaming pipelines using Java, Python, or Go

Beam currently supports the following language-specific SDKs:

- Apache Beam Java SDK 

- Apache Beam Python SDK 
- Apache Beam Go SDK 

A Scala  interface is also available as Scio.

Cloud Dataflow GCP details

- Cloud Dataflow can natively connect to Pub/Sub, BigQuery, Cloud ML Engine
- You can use something called **connectors** to connect to other Google Cloud services and external services as well such as Bigtable, and Apache Kafka
- Dataflow pipelines are regional based which you can choose

Cloud Dataflow and IAM

- Cloud Dataflow is unique in that it gives only gives project-wide level access
  - It's either
- Cloud Dataflow has 4 main IAM roles:
  - Dataflow Admin - anytime you see admin, that's typically the most access you can get. Admin gets full pipeline access + can list the machine type/storage bucket config access
  - Datflow Developer - you get full pipeline access but no machine type/storage bucket config access
  - Dataflow Viewer- view permissions only (read only)
  - Dataflow Worker - just for service accounts
    - If you have a service account for your Dataflow pipeline (which is highly recommended) you would use this role

Apache Beam - Combine vs GroupBy

- Apache Beam offers two primary options for performing aggregations: ***Combine and GroupBy***
- Combine functions allow you to efficiently perform parallel aggregations over a collection of elements.
    - For example, you can calculate the sum, average, maximum, or any other custom aggregation over a large dataset in a distributed manner.

- GroupBy allows you to group elements by key and process them together.
    - It's useful for operations such as grouping events by user, partitioning data by timestamp, or performing per-key aggregations.
    - GroupBy can be particularly beneficial when you need to transform your data based on key-value pairs.

Apache Beam - Combine vs GroupBy Example

- Consider a scenario where you have a dataset of online sales transactions, and you want to calculate the total revenue for each product category (like toys, clothes, etc.).

- You can use the GroupBy operation to group the sales data by category and then perform a sum aggregation on the revenue field within each category.

- This is quite similar to how it would work in SQLL

Cloud Dataproc

Cloud Dataproc - data processing

- Cloud Dataproc is a managed Apache Hadoop and Apache Spark service offered by Google Cloud Platform (GCP).
- It simplifies the deployment and management of big data processing and analytics workflows.
- With Cloud Dataproc, users can leverage the power of Hadoop and Spark to process large datasets efficiently.

Key Features of Cloud Dataproc

- Fully Managed: Google handles the provisioning, configuration, and maintenance of the underlying infrastructure, allowing users to focus on data processing tasks.

- Compatibility: Supports Apache Hadoop and Apache Spark, allowing users to run existing Hadoop and Spark jobs without modification.

Key Features of Cloud Dataproc cont..

- Autoscaling: Automatically adjusts the cluster size based on workload demands, optimizing resource allocation and reducing costs.
- Integration: Seamlessly integrates with other GCP services, such as BigQuery, Cloud Storage, and Dataflow, enabling end-to-end data processing pipelines

Cloud Dataproc vs Dataflow

- Cloud Dataproc and Dataflow are **both managed services** provided by Google Cloud Platform (GCP) for ***processing big data workloads.***
  - ***So at a high level - they are both services for data processing***
- While they have **overlapping capabilities,** they **differ in terms of use cases** and architectural **approach**.
- Let's examine the differences

Cloud Dataproc

- Batch Processing: Ideal for processing large volumes of data using batch-oriented frameworks like Apache Hadoop and Apache Spark.
- User-Managed Clusters: Users have direct control over cluster configurations and can optimize for specific use cases.
- Pre-existing Code: Supports running existing Hadoop and Spark jobs, making it easy to migrate on-premises workloads to the cloud.
- Performance: Offers high-performance computing capabilities suitable for complex data processing and machine learning tasks.

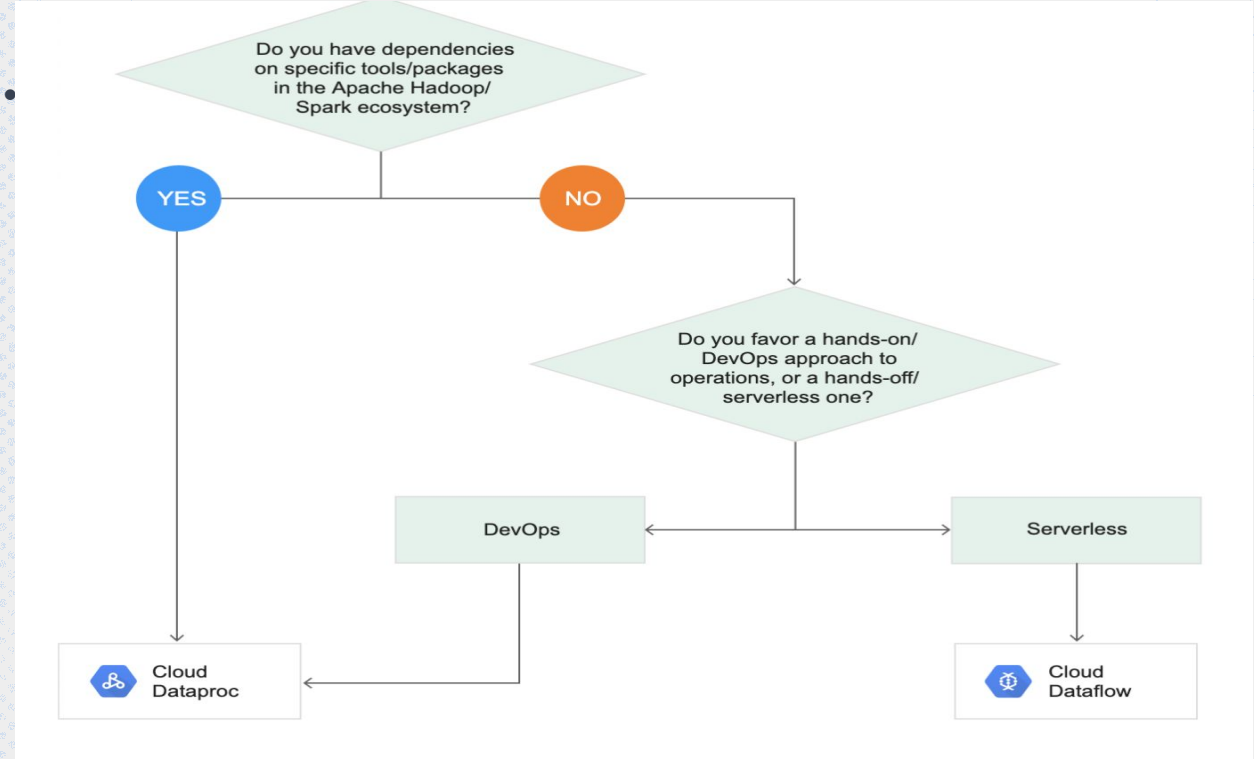
Back to Cloud Dataflow (brief comparison)

- Stream **and** Batch Processing: Designed for both stream and batch processing of data using a unified programming model.
- **Serverless** (opposed to user-managed clusters with Dataproc): Automatically manages the infrastructure, allowing developers to focus solely on writing data processing logic.
- Fully Managed: (**by GCP)** Provides automatic scaling, fault tolerance, and support for event time-based windowing.
- Integration with Pub/Sub: Integrates seamlessly with Pub/Sub for real-time data ingestion and processing.

Dataflow or Dataproc?

- Batch vs. Stream Processing: If the workload primarily involves batch processing and utilizes Hadoop or Spark, Cloud Dataproc is a suitable choice. For real-time stream processing, Cloud Dataflow is more appropriate.
- User Control vs. Serverless Experience: Cloud Dataproc offers fine-grained control over cluster configuration, while Cloud Dataflow provides a serverless experience with infrastructure management abstracted.
- Existing Code vs. Unified Model: If you have existing Hadoop or Spark jobs, Cloud Dataproc allows you to run them without modification. Cloud Dataflow provides a unified model for both batch and stream processing.

Cloud Dataproc vs Dataflow



Cloud Dataproc vs Dataflow

Workload	Cloud Dataproc	Cloud Dataflow
Stream processing (ETL)	No	Yes
Batch processing (ETL)	Yes	Yes
Iterative processing and notebooks	Yes	No
Machine learning with Spark ML	Yes	No
Preprocessing for machine learning	NO	Yes (with Cloud ML Engine)

Dataflow/Dataproc use cases

- Cloud Dataproc: ETL pipelines, log analysis, machine learning model training, data warehousing, and analytics.
  a. Cloud Dataproc is well-suited for processing massive datasets using Hadoop and Spark.
- Cloud Dataflow: Real-time data processing, data streaming, event-driven architectures, and data transformation.

Dataflow/Dataproc + other GCP services

- Both Cloud Dataproc and Dataflow integrate with various GCP services, such as BigQuery, Cloud Storage, Pub/Sub, AI Platform, and more, enabling end-to-end data processing pipelines.

Dataproc, Spark, Hadoop terminology

- Cloud Dataproc, Spark, and Hadoop each utilize specific terminology that is important to understand when working with these technologies. Let's explore terminology for each in further slides

Cloud Dataproc - terminology

- Cluster: A group of virtual machines (VMs) used for distributed data processing.
- Main Node: The central coordinator in a cluster responsible for managing and monitoring tasks.
- Worker Nodes: The compute instances that perform data processing tasks in a cluster.
- Job: A unit of work submitted to the cluster for execution.
- Cluster Mode: Specifies how the cluster resources are allocated between the master and worker nodes.

Apache Spark - terminology - RDD

- Apache Spark is a distributed data processing framework commonly used with Cloud Dataproc.
- **RDD** (Resilient Distributed Dataset):
  a. An immutable distributed collection of objects used for data processing.
  b. RDDs provide fault tolerance and efficient parallel processing in Spark.
  c. Example: Creating an RDD from a text file to perform word count analysis.

Apache Spark - terminology - Dataframe

- Dataframe: A distributed collection of data organized into named columns.
- DataFrames offer a higher-level abstraction and support structured and semi-structured data processing.
- Example: Loading a CSV file into a DataFrame for data exploration and manipulation.

Apache Spark - terminology - Spark Streaming

- Spark Streaming - An extension of Spark for real-time stream processing and data integration.
- Spark Streaming ingests data in small batches and processes them in near real-time.
- Example: Processing and analyzing streaming data from a message queue.

Dataproc + Apache Spark lab

- https://github.com/varoonsahgal/sp-gcp-data/wiki/Dataproc-and-Pub-Sub

Machine Learning + API's in GCP

What is Machine Learning?

- It's often thought that you input some data to a machine and out comes some magical insight - and that is how machine learning works
  a. But that's not true!
- Machine Learning == process of combining inputs to produce useful predictions on ***never before seen data***
- A machine learning **model** is the program which gets trained and can then make those decisions/predictions on data not seen before

Training machine models

- When we train machine models, it's a bit like training a 3 year old human to learn things for the first time
- Feeding the model is what's known as training
- For example, if we want to train a machine on how to recognize that a picture of a dog is indeed a dog, we would first have to train that machine by showing it many, many pictures of dogs beforehand
    a. We would also have to label the data so that it understands what it's looking at
    b. The more data you have to feed into a model the better
        i. It can then see dogs from different angles, different shades and then make better decisions on never before seen data accordingly!

Testing machine models

- Once you train a machine model you would then of course test it out with a different dataset as well

How do we create machine models

- Can use something like Tensorflow framework to start creating models
  a. See here: https://www.tensorflow.org/overview
- ChatGPT is built on the OpenAI GPT-3.5 model, which is developed using the TensorFlow deep learning framework.

GCP - Machine Learning API's

- Google Cloud Platform (GCP) provides a suite of powerful Machine Learning (ML) APIs that enable developers to incorporate pre-trained ML models into their applications.
- These APIs offer ready-to-use functionality for tasks such as image analysis, natural language processing, translation, and more.
- Let's explore some of the key ML APIs in GCP and how they can be leveraged in real-world scenarios.
- Note that some offerings are **pre-trained** whereas others are models that **you can train yourself (Auto-ML)**
  a. **Auto-ML see here: https://cloud.google.com/automl**
  b. **See everything here: https://cloud.google.com/products/ai**

Cloud Vision API

- The Cloud Vision API allows you to analyze and understand the content of images.
- Example Use Cases:
  - Content moderation: Detect inappropriate or unsafe images in user-generated content.
  - Object recognition: Identify objects and their locations within an image.
  - Optical character recognition (OCR): Extract text from images, including handwritten text.
  - Facial recognition: Recognize faces and perform facial analysis.

Natural Language API

- The Natural Language API provides insights into the structure and meaning of text.
- Example Use Cases:
  - Sentiment analysis: Determine the sentiment (positive, negative, or neutral) of a text.
  - Entity recognition: Identify and classify entities (people, places, organizations) mentioned in the text.
  - Content classification: Categorize text into predefined or custom categories.
  - Syntax analysis: Parse the grammatical structure of sentences.

Translation API

- The Translation API enables you to automatically translate text between languages.
- Example Use Cases:
  - Multilingual support: Translate web content, chat messages, or product descriptions into different languages.
  - Localization: Localize software applications or user interfaces for international audiences.
  - Content enrichment: Translate user-generated content for better accessibility and reach.

Video Intelligence API

- The Video Intelligence API allows you to extract insights from videos.
- Example Use Cases:
  - Content moderation: Detect and filter inappropriate or unsafe content in user-uploaded videos.
  - Object tracking: Track and identify specific objects or subjects throughout a video.
  - Scene detection: Automatically identify different scenes or segments within a video.
  - Speech transcription: Extract spoken words and generate text captions for videos.

Machine Learning + Vision API Lab

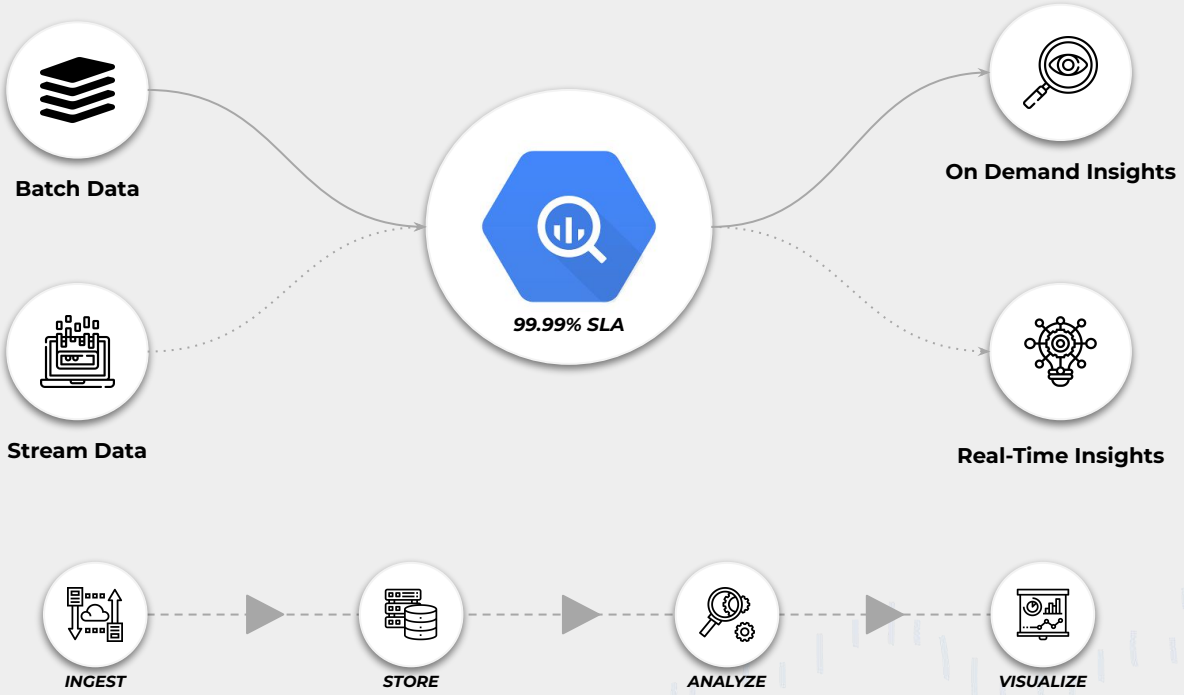
- https://github.com/varoonsahgal/sp-gcp-data/wiki/Vision-ML-APIs-lab

BigQuery

Agenda

- Serverless data warehousing
- Columnar vs. Row-based Storage
- Normalization vs. Denormalization with Columnar Storage
- Projects, Datasets, Tables
- Batch Data Import/Export
- Semi-Structured Data Analysis with SQL Arrays and Structs
- Partitions and Performance Optimizations

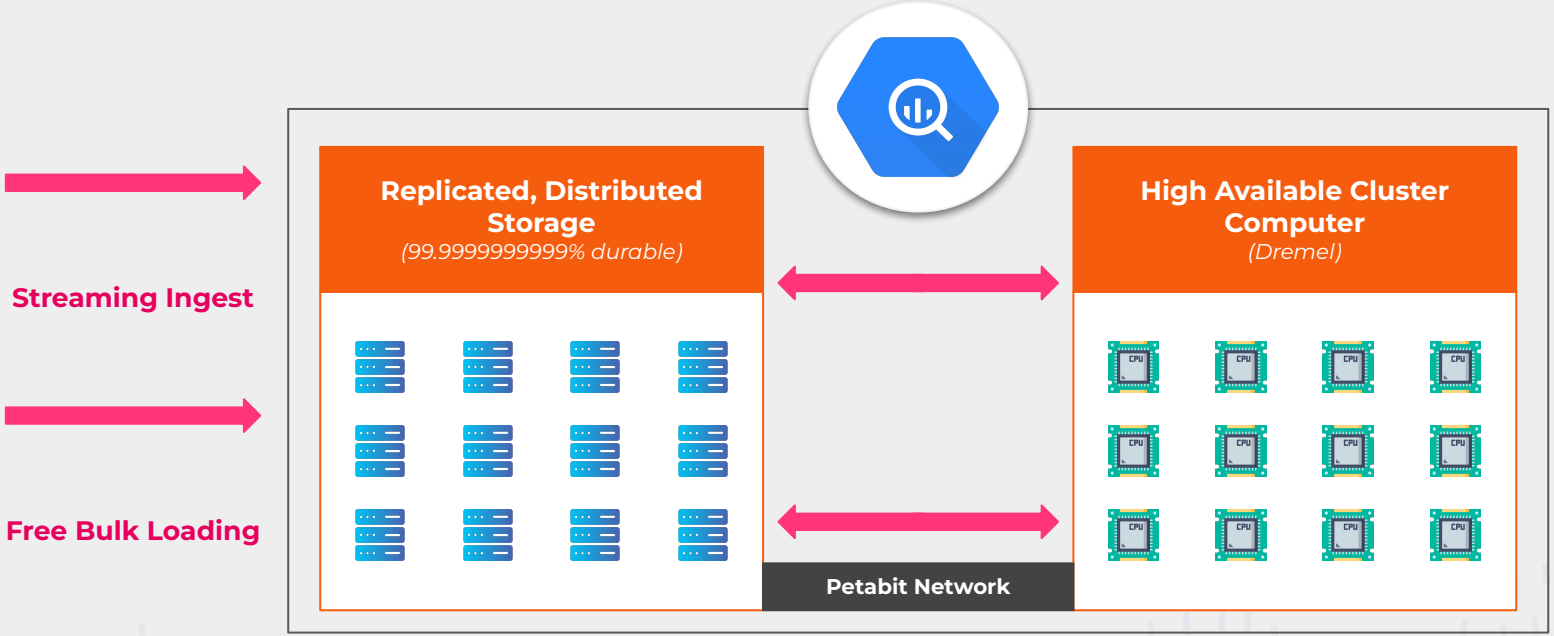
What is BigQuery?



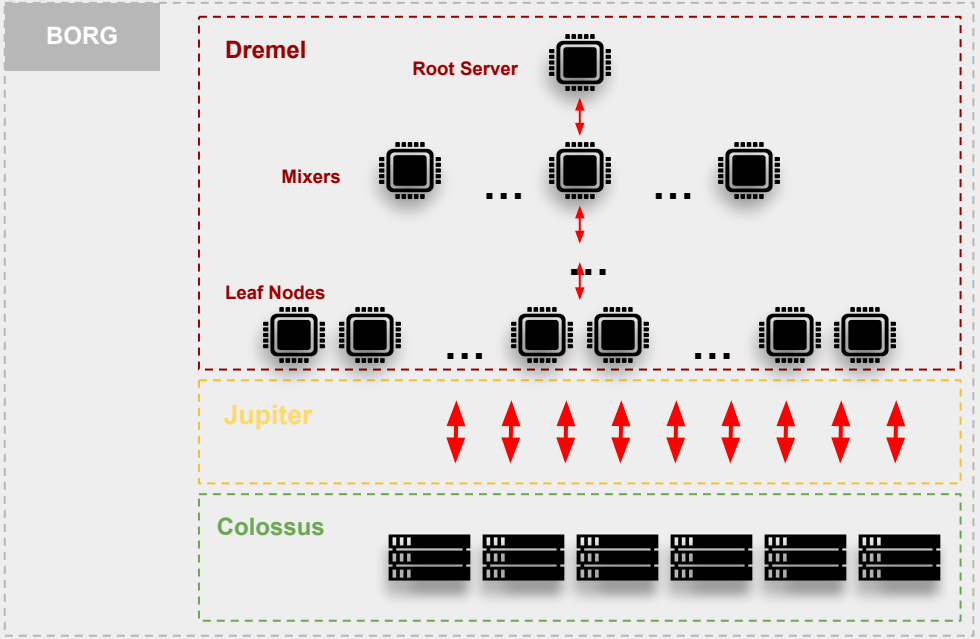
What is BigQuery?

- Google BigQuery is a fully-managed enterprise data warehouse
- It is a cloud-based big data analytics web service for processing very large read-only data sets
- It runs on serverless/no-ops (no clusters/nodes) architecture and uses SQL-like syntax to query and analyze data in the order of billions of rows
- It was developed by Google ,released for public in 2011 and runs on Google cloud storage infrastructure & supports REST APIs
- Because of its scalable and distributed analysis engine, it can query terabytes of data in seconds and petabytes of data in minutes

How does BigQuery work?



How does BigQuery work?



Dremel - *Massively parallel real-time query engine with SQL + REST API*

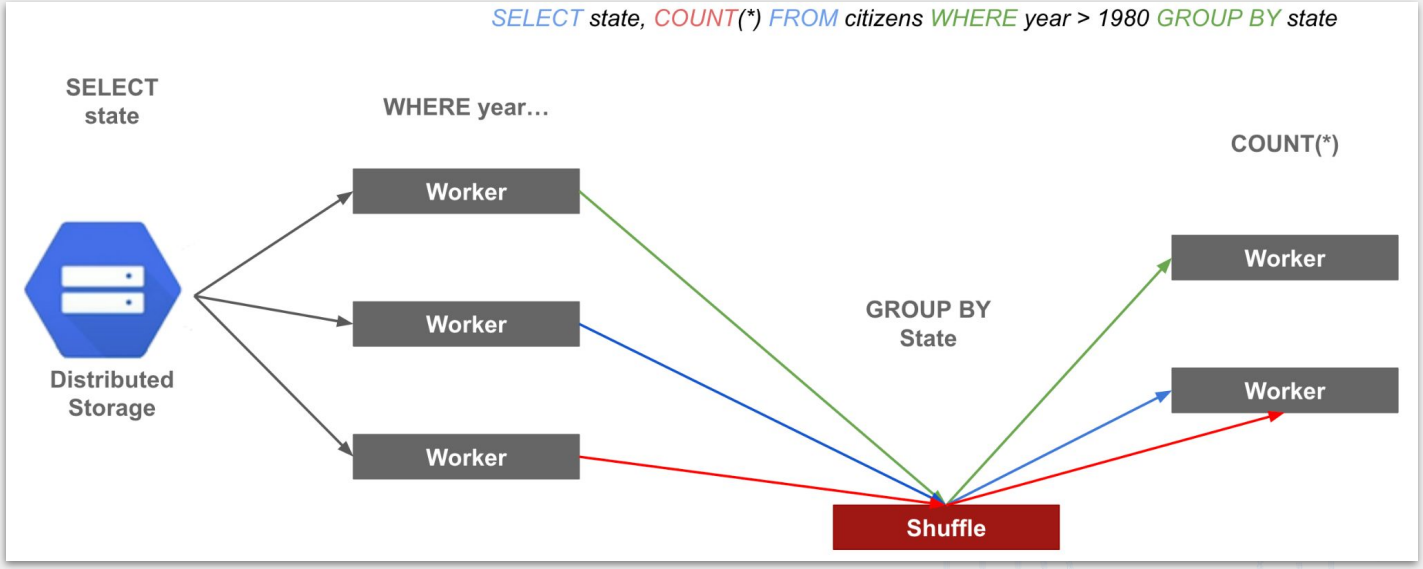
Colossus - *Distributed Storage layer for Dremel, successor to GFS/HDFS*

Capacitor - *Columnar nested + compressed file format for Dremel, inspiration for Parquet etc*

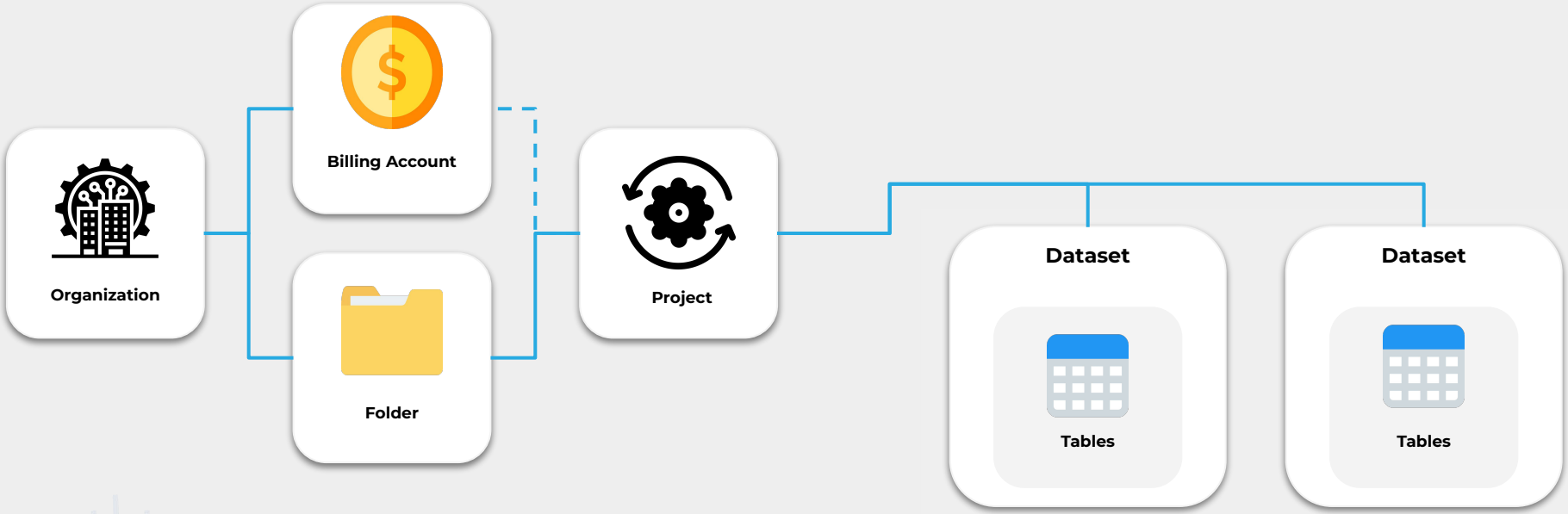
Borg - *Large-Scale cluster management, run Dremel jobs on 10000+ servers*

Jupiter - *Google's high-capacity low-latency internal network*

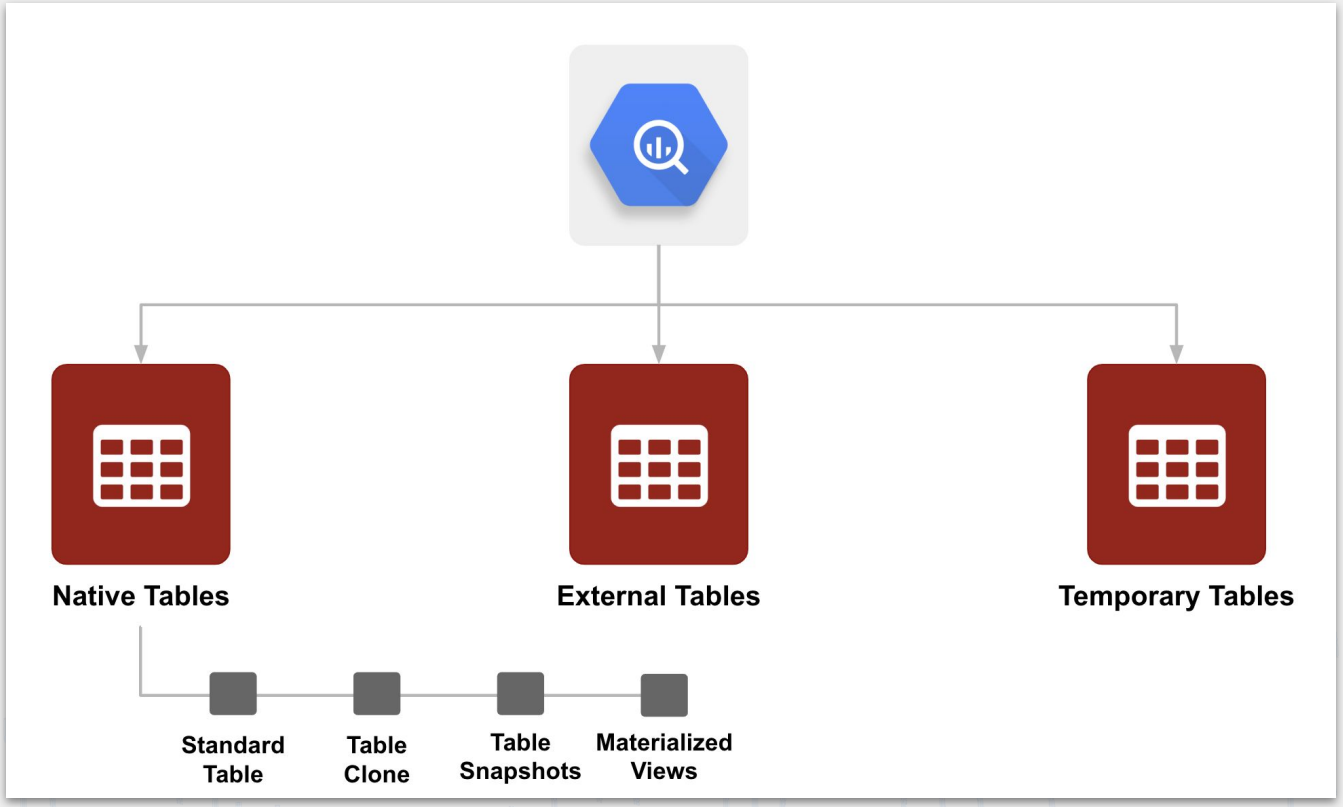
How does BigQuery Query Execution?



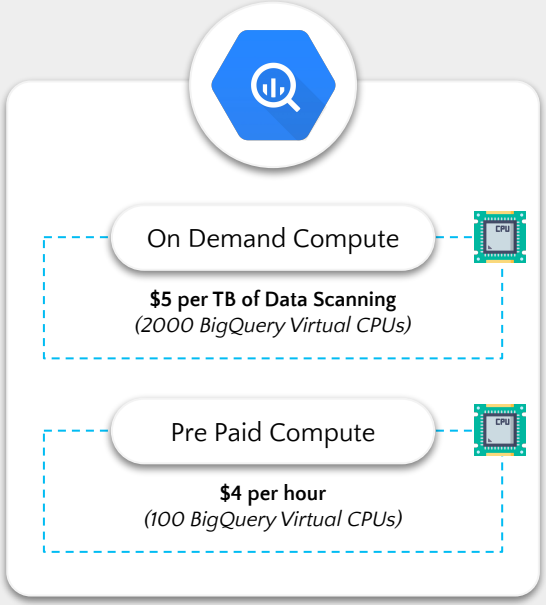
Project Hierarchy

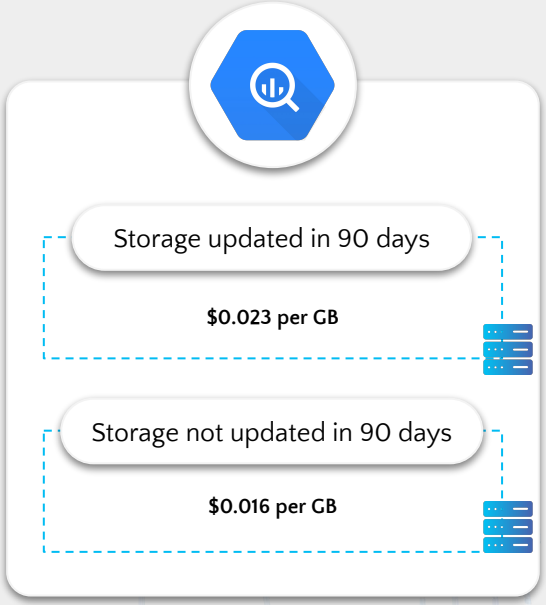


Types of BigQuery Tables



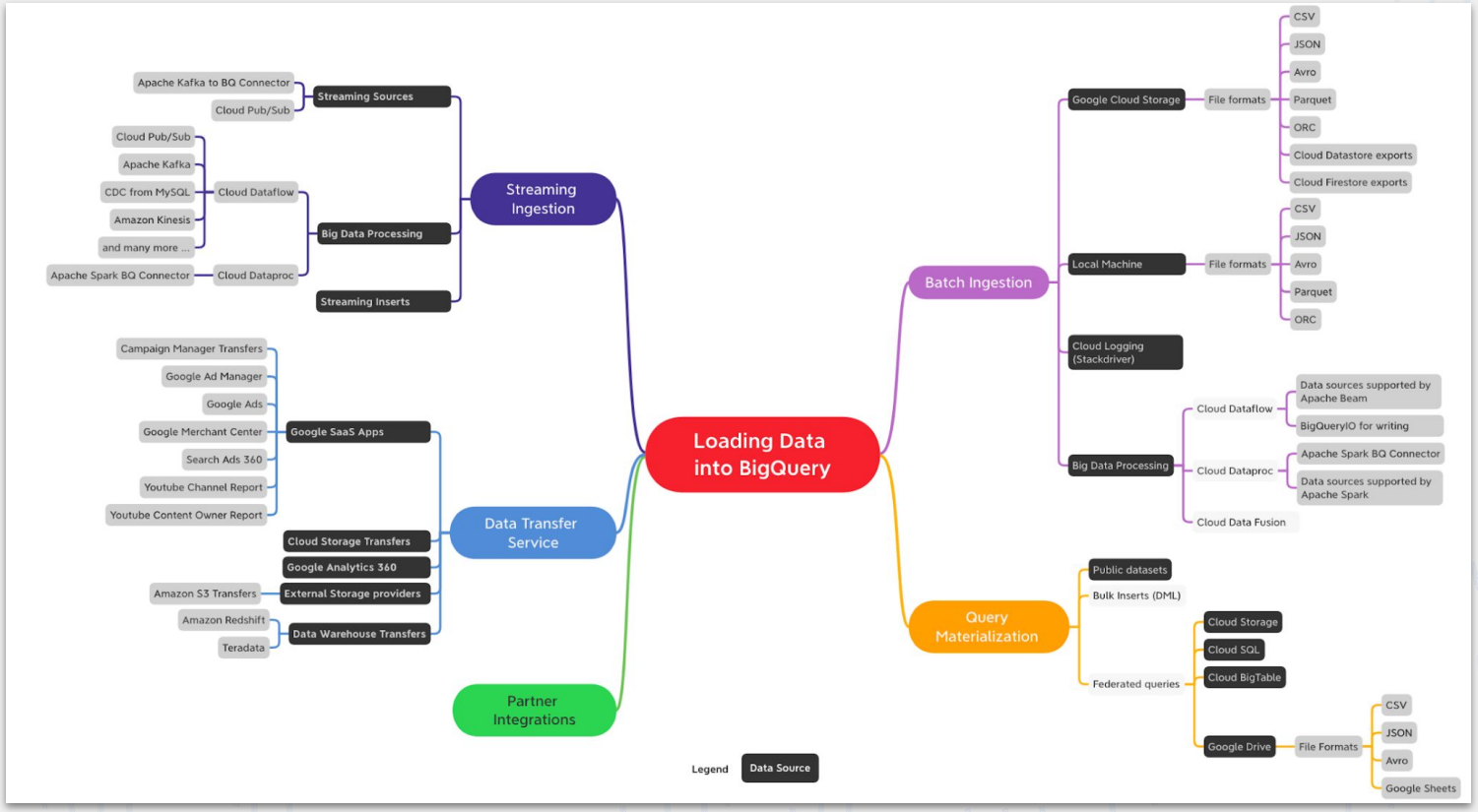
BigQuery Pricing





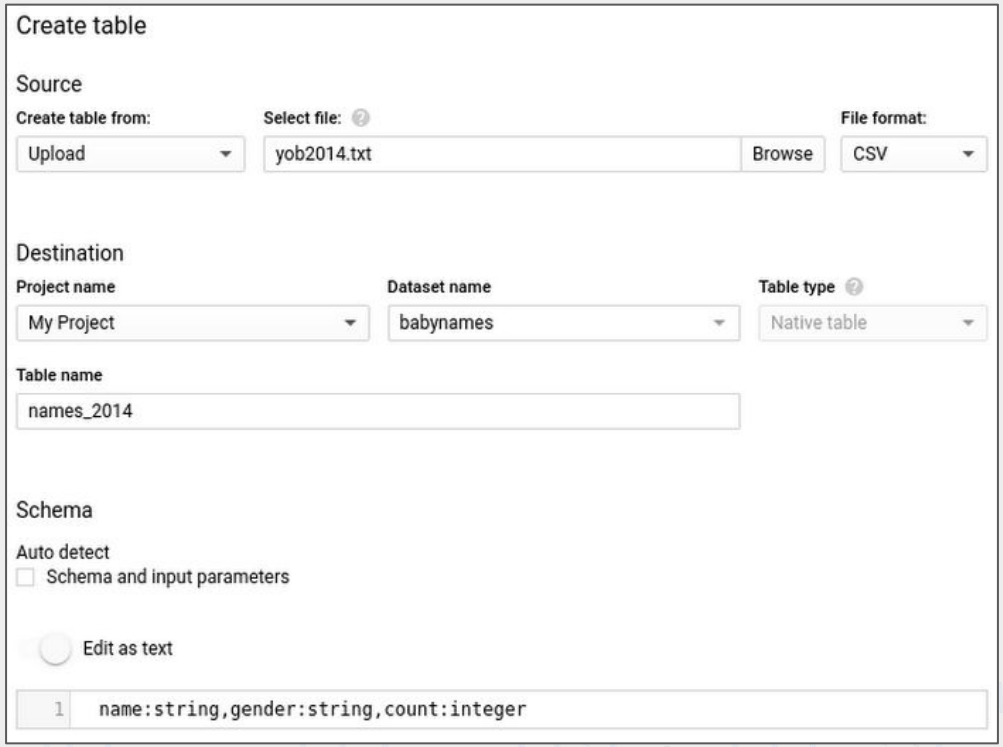
When to choose BigQuery?

How to import the data in BigQuery

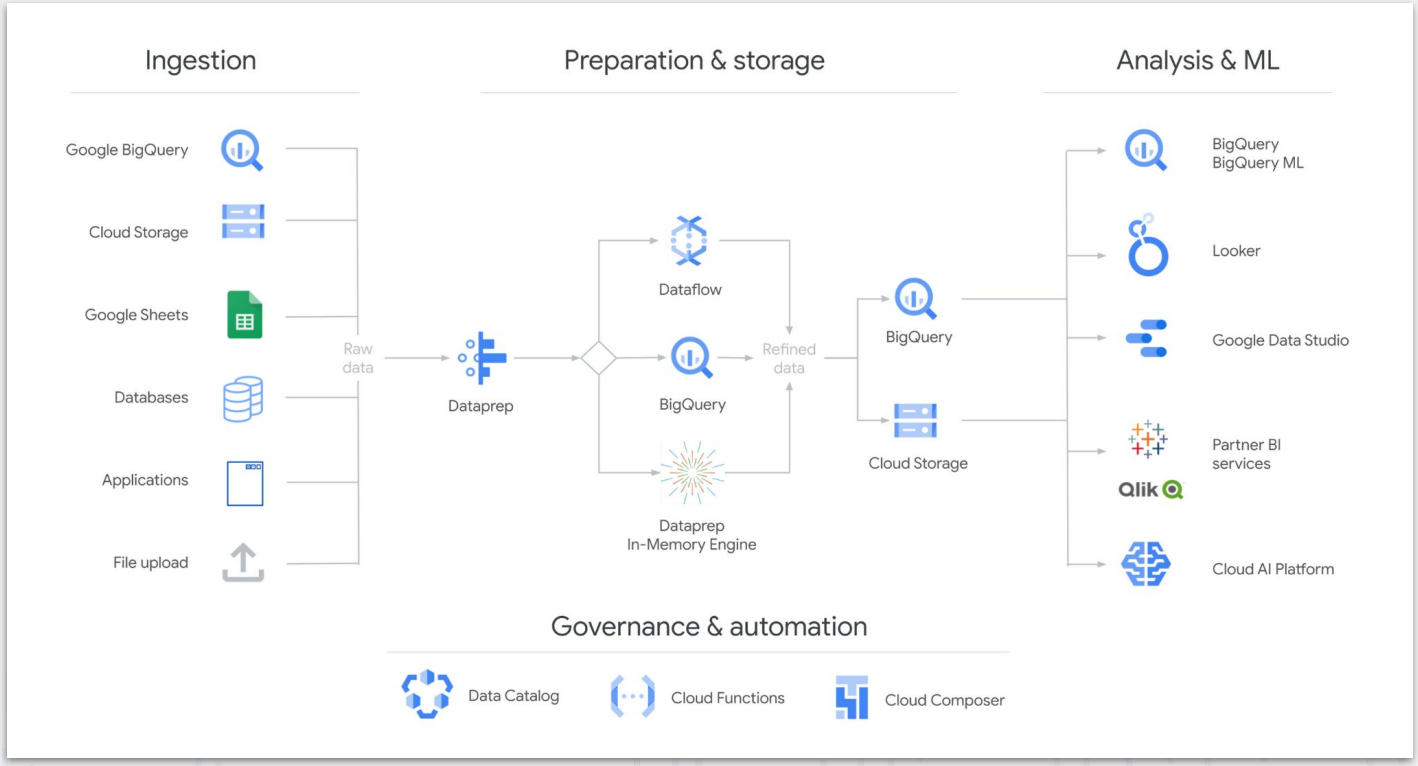


How to import the data in BigQuery

Load Using BigQuery UI



Preparing and Transforming Data



Load data through Cloud Storage

- Go to BigQuery Cloud Console
- In the navigation panel, in the **Resources** section, expand your Google Cloud project and select a dataset
- On the right side of the window, in the details panel, click **Create table**. The process for loading data is the same as the process for creating an empty table

Load data through Cloud Storage

- On the Create table page, in the Source section:
- For Create table from, select Google Cloud Storage

- Now click on Browse, for browsing the data in GCS
- After that select the File Format accordingly

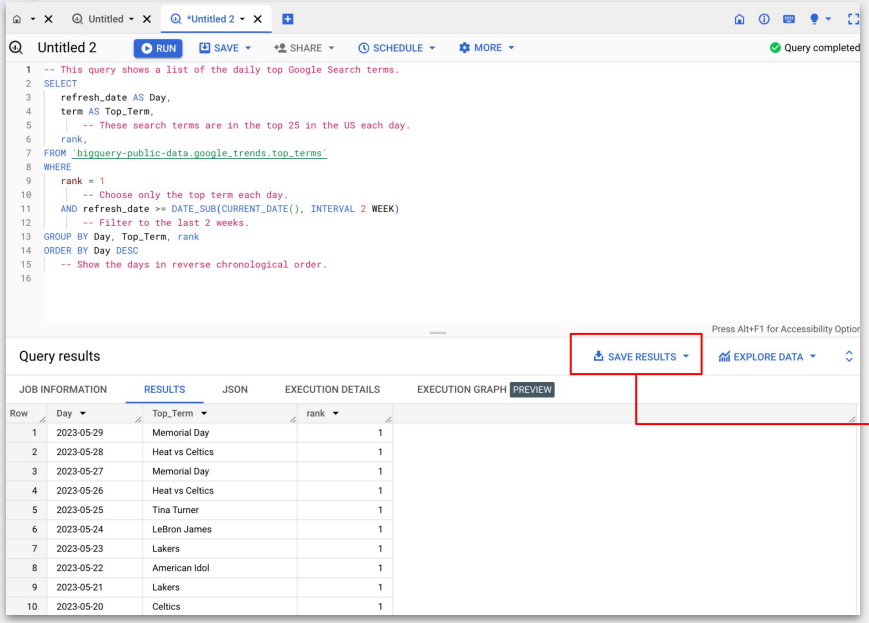
Load data through Cloud Storage

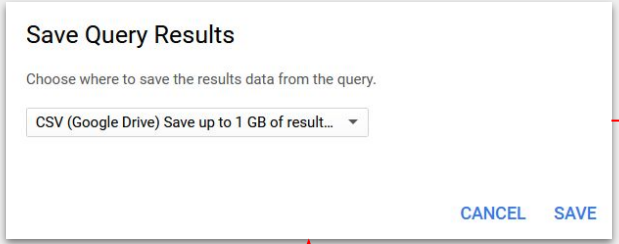
- Now provide the Table name and click on Create

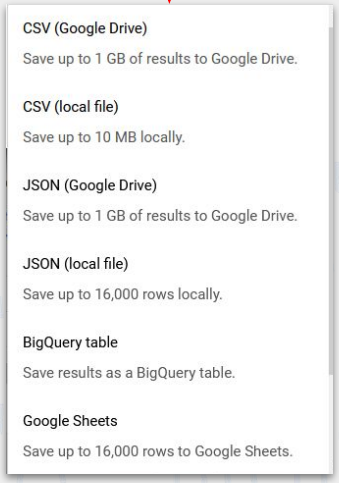
How to Export the data in BigQuery

- Run any query within the BigQuery Editor
- After that click on Save Result button on bottom
- Then select one option for saving the file out of many (local, GCS, BigQuery Table, etc)
- Click on Save button

How to Export the data in BigQuery







Semi-structured Data Processing using BigQuery

- JSON, AVRO, Parquet are completely supported
- Ingest real-time data at scale
- Load & Analyze data in seconds
- Transform data with SQL
- Combine data with any other business data

Working with JSON Data

- BigQuery natively supports JSON data using the JSON data type
- JSON is a widely used format that allows for semi-structured data, because it does not require a schema
- Applications can use a "schema-on-read" approach, where the application ingests the data and then queries are executed based on the assumptions about the schema of that data
- This approach differs from the STRUCT type in BigQuery, which requires a fixed schema that is enforced for all values stored in a column of STRUCT type
- By using the JSON data type, you can ingest semi-structured JSON into BigQuery without providing a schema for the JSON data upfront
- This lets you store and query data that doesn't always adhere to fixed schemas and data types
- By ingesting JSON data as a JSON data type, BigQuery can encode and process each JSON field individually

Working with JSON Data - Creating Tables

- Go to BigQuery Web UI
- Run the following query:

```
CREATE TABLE mydataset.table1(
  id INT64,
  cart JSON
);
```

Note: You can't partition or cluster a table on JSON columns, because the equality and comparison operators are not defined on the JSON type

Working with JSON Data - Create JSON Values

You can create JSON values in the following ways:

- Use SQL to create a JSON literal
- Use the **PARSE_JSON** function to convert a string to a JSON type
- Use the **TO_JSON** function to convert a SQL type to a JSON type

We can create JSON Literal as well:

```
INSERT INTO mydataset.table1

VALUES(1, JSON '{"name": "Alice", "age": 30}');
```

Working with JSON Data - Create JSON Values

- We can converter String to JSON by using PARSE_JSON function
- Let's see an example which converts a column from an existing table to a JSON type and stores the results to a new table

```
CREATE OR REPLACE TABLE mydataset.table_new
AS (
  SELECT
    id, SAFE.PARSE_JSON(cart) AS cart_json
  FROM
    mydataset.old_table
);
```

Note: The SAFE prefix used in this example ensures that any conversion errors are returned as NULL values

Using Nested & Repeated Fields

- BigQuery doesn't require a completely flat denormalization. You can use nested and repeated fields to maintain relationships as follows:

BigQuery Best Practices

Controlling Costs

- Avoid SELECT * (full scan), select only essential columns
  - LIMIT doesn't limit costs (still a full scan)
- Use SELECT * EXCEPT
- Sample data using the free Preview option
- Preview (dryrun) queries to estimate costs
- Use ***max bytes*** billed to limit query costs
- Monitor costs using dashboards and audit log
- Partition data by date — breaks query results into stages
- Delete data no longer needed using default dataset/table/view expiration
- Apply hard limit on bytes processed per day

BigQuery Best Practices

SQL Anti-Patterns

- Avoid self-joins – Use window functions to perform calculations across many rows related to current row in same table
- Skewed Partitions – Avoid unequally sized partitions, for e.g., when one value occurs more often than any other value
- Avoid Cross-Joins – These joins generate more output than input. Use pre-aggregated data or window function
- Updating/Inserting Single Row/Column – Avoid point-specific DML, instead prefer batch updates and inserts

BigQuery Best Practices

Optimizing Query Computation

- Avoid repeatedly transforming data via SQL queries
- Use SQL instead of avoid JavaScript for user-defined functions
- Use approximate aggregation functions (e.g. APPROX_COUNT_DISTINCT) if precision is not needed
- Optimize query performance by ordering its operations
    - Use ORDER BY only in the outermost query
    - Push complex operations to the end of the query
    - Wherever joining data from multiple tables, optimize join patterns – start with the largest table

BigQuery Best Practices

Managing Query Outputs

- Avoid repeated joins and using the same subqueries
- Writing out large result sets has performance/cost impacts
    - Use filters or LIMIT clause
    - Use cached results as much as possible – cache size is 10GBs in compressed form
- Use LIMIT clause for sorting large tables to avoid Resources Exceeded error

Data engineering with GCP

How to choose a Storage Service

- When choosing a storage service in Google Cloud Platform (GCP), it's important to consider various factors to make an informed decision.
- This decision model helps evaluate different storage services based on specific requirements and use cases.
- Let's explore the decision model with examples to understand the process better.

Factors to consider

- Performance: Consider the required throughput, latency, and IOPS (input/output per second) for your application.
- Scalability: Evaluate the ability of the storage service to handle growing data volumes.
- Durability: Assess the data protection mechanisms and backup options provided by the storage service.
- Cost: Analyze the pricing models and associated costs for the storage service.
- Access Control: Evaluate the security features and granular access controls available.

Example 1: High Throughput Analytics

- Requirement: A data analytics application that processes large amounts of data with high throughput.
- Recommendation: BigQuery
- Explanation: BigQuery provides fast query processing for large datasets, optimized for analytics workloads. It offers high scalability and built-in data warehousing capabilities.

Example 2: Real Time Data Processing

- Requirement: A real-time streaming application that processes and analyzes data in near real-time.
- Recommendation: Cloud Pub/Sub and Cloud Dataflow
- Explanation: Cloud Pub/Sub offers scalable and reliable messaging, while Cloud Dataflow provides powerful data processing capabilities for real-time streaming applications.

.

Thank you!

If you have any additional questions, please ask!If