Google Professional Data Engineer Certification Exam

Last updated November 13, 2020

This reviewer is based on the [modules and exam guide provided via the Google Partner Kickstart program]. I have taken the liberty of combing through the whole exam guide and collating all the important points in this reviewer.

It is my hope that you find this reviewer helpful in your own chase for the Google Pro DE certification.

Kiana Alessandra V. Villaera
kvvillaera@up.edu.ph

Table of Contents

Part 1: Designing Data Processing Systems **1**

Part 2: Building and Operationalizing Data Processing Systems **20**

Part 3: Operationalizing Machine Learning Models **30**

Part 4: Reliability, Policy, and Security to ensure solution quality **49**

Part 1: Designing Data Processing Systems

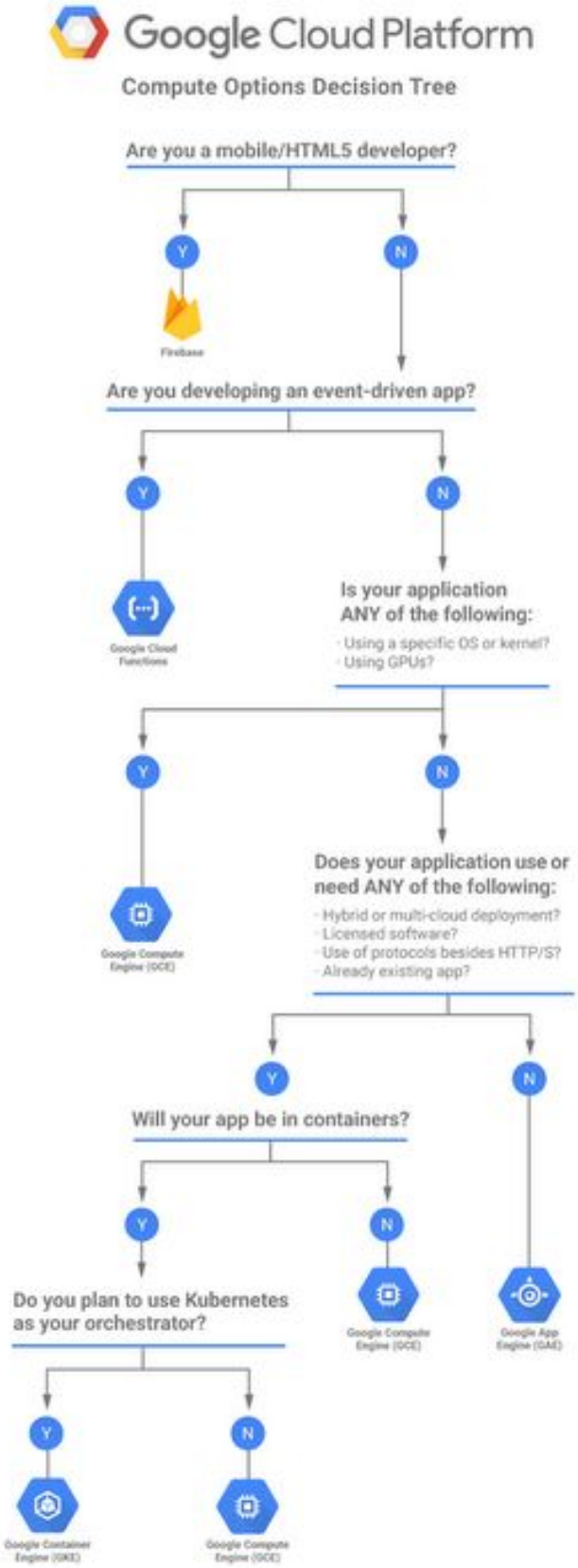
- Data Storage
  - Solutions include:
    - **Managed:**
      - **Cloud SQL**
      - **Cloud Bigtable**
    - **Serverless:**
      - **Cloud Storage**
      - **Cloud Datastore**
      - **Cloud Firestore**
      - **Cloud Spanner**
  - Know the difference between Cloud Spanner and Cloud SQL
    - Cloud Spanner is used when you:
      - **need to handle massive amounts of data** with an elevated consistency with a
      - **big amount of data handling** (+100,000 reads/writes per second)
      - Spanner gives much **better scalability and better SLOs**

        - Use Cloud SQL if you need **cheap storage without server configuration**
        - More on Cloud Spanner
            - **Fully managed relational database with UNLIMITED scale**
            - strong **CONSISTENCY**
            - Up to **99.999% availability**
            - Handles a**utomatic sharding, transaction processing, replicas**, etc.
    - Know the difference between Cloud Datastore and Cloud Firestore
        - Cloud Datastore is basically a **giant JSON tree**
            - Objects are saved as **entities** which can be classified under a **kind** and appended with **attributes**
        - Cloud Firestore is **more structured**
            - More powerful querying capabilities versus Cloud Datastore
        - More robust SLA than the realtime database
        - All Cloud Firestore queries scale to the size of your result set. The search for the top 10 restaurants in Chicago for a restaurant review app will take the same amount of time whether your database has 300 restaurants, 300 thousands, or 30 million
        - Firestore has multi-region support. Realtime Database is only hosted in NA
        - Different pricing model. Realtime Database focuses on the amount of data that is downloaded and stored. Firestore is primarily driven by the number of reads or writes that you perform
        - More in this blog
    - Considerations for Representation, Pipeline, and Infrastructure may be different
    - Service differentiators include access methods, cost/speed of specific actions, size of data, how data is organized or stored
    - Know how to identify technologies backwards from their properties
    - Managed Services still have IT overhead but it is minimal compared to on-prem solutions
    - Serverless services remove IT responsibility and server management is not part of the overhead. Individual instances are not visible.

Choose your solutions based on access pattern

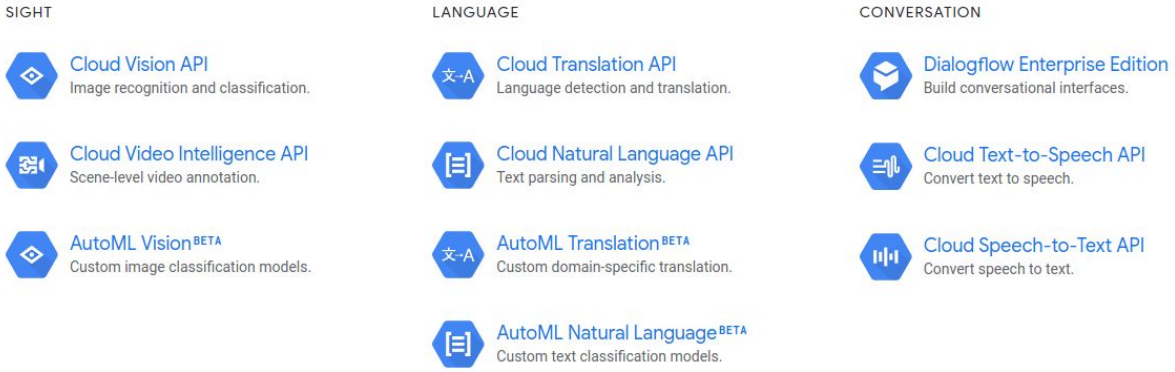
	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Data warehouse
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

- Data Processing
    - Solutions include:
        - **Compute Engine**
        - **App Engine**
        - **Kubernetes Engine**
        - **Cloud Functions**
        - **Managed:**
            - **Cloud Dataproc**
        - **Serverless:**
            - **BigQuery**
            - **Cloud Dataflow**
    - Data Processing combines storage and compute and abstracts these processes. Example: **DAG (Directed Acyclic Graph)** for Dataproc, **Tables and queries** for BigQuery, and **Pcollections and pipelines** for Dataflow
    - The difference between Google App Engine and Google Compute Engine is that
        - App Engine is a **Platform-as-a-Service**. You **simply deploy your code and the platform does everything else for you**
        - **Compute Engine** is an **Infrastructure-as-a-Service**. You have to create and configure your own VM instances. **Costs less than App Engine but you have to manage everything yourself**
        - **Cloud Functions** is the next level up from the App Engine in terms of abstraction because there are no instances. This is **ideal for rare calls and rapidly changing loads**
        - **Google Container Engine** is for the management and orchestration of Docker container clusters that run within Google's public cloud services. It is based on Kubernetes
        - More [in this video]


Google Cloud Platform
Compute Options Decision Tree
Are you a mobile/HTML5 developer?
Y
N
Firebase
Are you developing an event-driven app?
Y
N
Google Cloud Functions
Is your application ANY of the following:
· Using a specific OS or kernel?
· Using GPUs?
Y
N
Google Compute Engine (GCE)
Does your application use or need ANY of the following:
· Hybrid or multi-cloud deployment?
· Licensed software?
· Use of protocols besides HTTP/S?
· Already existing app?
Y
N
Will your app be in containers?
Y
N
Do you plan to use Kubernetes as your orchestrator?
Google Compute Engine (GCE)
Google App Engine (GAE)
Y
N
Google Container Engine (GKE)
Google Compute Engine (GCE)

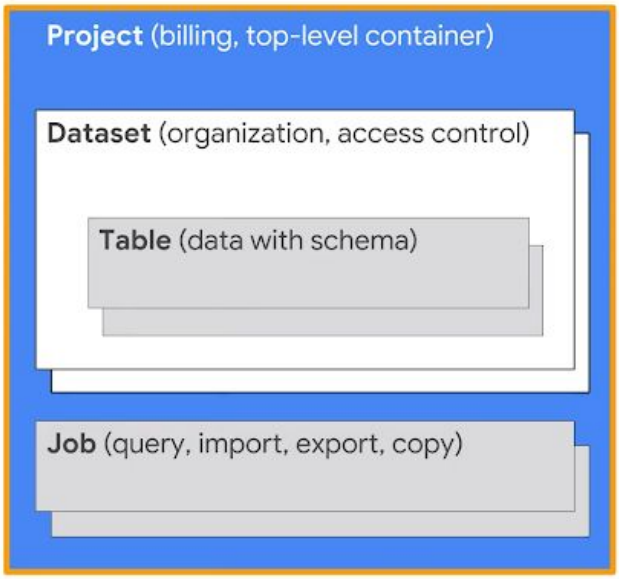
- Artificial Intelligence
    - Solutions include:
        - **Customized**
            - **Cloud ML**
            - **Cloud TPU**
            - **Cloud AutoML**
        - **Pre-built**
            - **Cloud Vision API**
            - **Cloud Speech-to-Text API**
            - **Cloud Video Intelligence API**
            - **Cloud Natural Language API**
            - **Cloud Translation API**
    - Makes unstructured data useful
    - Understand when you might want to use each AI solution on GCP
- Pre and Post processing services
    - Solutions Include:
        - **Transfer Appliance**
        - **Cloud Dataprep**
        - **Cloud Datalab**
        - **Cloud Data Studio**
        - **Dialog Flow**
- Infrastructure Services
    - Solutions Include:
        - **Messaging**
            - **Cloud Pub/Sub**
        - **Automation**
            - **Cloud Deployment Manager**
            - **Cloud Composer**
            - **Cloud APIs**
        - **Operations**
            - **Stackdriver**
            - **GCP Console and Cloud Shell**
            - **Monitoring**
            - **Logging**
            - **Error Reporting**
            - **Trace**
            - **Debugger**
        - **Networking**
            - **Cloud VPN**
            - **Partner Interconnect**
            - **Dedicated Interconnect**
            - **Cloud IoT Core**
            - **Cloud Endpoints**
        - **Security**

            - **Cloud Armor**
            - **Cloud Firewall Rules**
            - **Cloud IAM**
            - **Key Management Service**
    - Don't memorize unimportant details like actual prices



- Designing Flexible Data Representations
    - It is good to know how data is stored and what purpose or use case it is optimized for
        - Cloud Storage is an **OBJECT stored in a BUCKET**
        - Cloud Datastore is a **PROPERTY contained in an ENTITY and is in a KIND category**
        - Cloud SQL consists of **VALUES stored in ROWS and COLUMNS in a TABLE in a DATABASE**
        - Cloud Spanner consists of **VALUES stored in ROWS and COLUMNS in a TABLE in a DATABASE**
    - Data in Files and Data in Transit
        - File Formats
            - CSV
            - XML
            - JSON
            - AVRO - Serialization and deserialization
        - BigQuery

BigQuery datasets, tables, and jobs



A project contains users and datasets
Use a project to:

- Limit access to datasets and jobs
- Manage billing

A dataset contains tables and views
Access Control Lists for Reader/Writer/Owner
Applied to all tables/views in dataset

A table is a collection of columns
Columnar storage
Views are virtual tables defined by SQL query
Tables can be external (e.g., on Cloud Storage)

A job is a potentially long-running action
Can be canceled

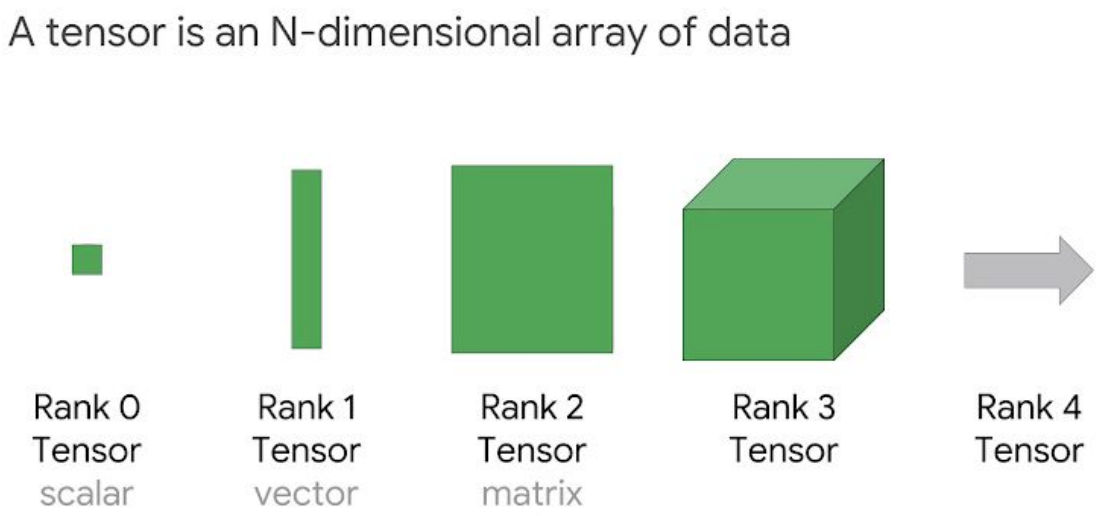
BigQuery storage is columnar

Record-oriented storage
Supports transactional updates

Each column in a separate, compressed, encrypted file that is replicated 3+ times
No indexes, keys or partitions required
For immutable, massive datasets

- RDD (Resilient Distributed Datasets)
    - Spark hides data complexity with abstraction
- Dataflow

        - Each step is a **transform**
        - Together they form a **pipeline**
        - The pipeline is executed on the cloud by a **Runner**
        - Each step is **elastically scaled**
        - Each transform is applied to a **PCollection**
        - The results of an apply() is another **PCollection**
        - Does ingest, transform, and load on **Batch or Stream**
        - For streaming data, define **windows** to make mini batches
    - Tensorflow
        - A tensor is an **N-dimensional array of data**



- Design Data Pipelines
    - Cloud Dataproc

Cloud Dataproc

Cluster node options
Single node (for experimentation)

Standard (1 master only)

High availability (3 masters)

Benefits
Hadoop—familiar

Automated cluster mgmt

Fast cluster resize

Flexible VM configurations

HDFS
Use Cloud Storage for a stateless solution

HBASE
Use Cloud Bigtable for a stateless solution

Objectives:
Shut down the cluster when it is not actually running jobs

Start a cluster per job or for a particular kind of work

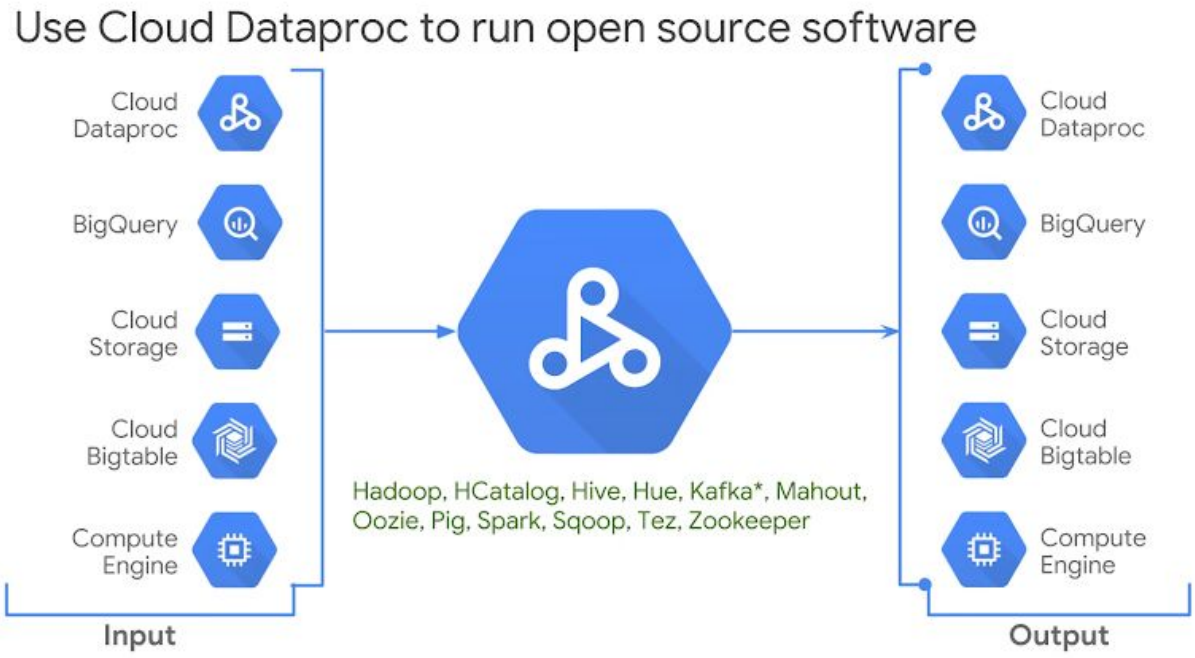
Data storage
Don't use hdfs to store input/output data; use it for temporary working storage

Disk performance SCALES with size!

Cloud storage
Match your data location with your compute location; zone matters

Machine types and preemptible VMs

- Hadoop is **hard to configure**
- Keep it **stateless**. Storage should be **separate**
- Spark can wait until all the requests are in before applying resources
- Transformations are **lazy** (it waits)
    - For transformations, Spark adds them to a DAG of computation and only when the driver requests some data, does this DAG actually get executed
- **Anonymous Functions** are like lambda functions (Eager execution)
    - Execute ASAP
    - Scala
- Dataproc can augment BigQuery
- Extract data in BQ, pull the data into the Spark cluster for further analysis
- Use Cloud Dataproc to run **open source software**



        - Use initialization actions to install custom software and cluster properties to configure Hadoop
- Cloud Dataflow
    - Overview

Cloud Dataflow

Write Java or Python code, deploy it to Dataflow, which then executes the pipeline.

Open-source API (Apache Beam) can also be executed on Flink, Spark, etc.

Parallel tasks are autoscaled by execution framework.

Same code does real-time and batch.

You can get input from any of several sources, and you can write output to any of several sinks. The pipeline code remains the same.

You can put code inside a servlet, deploy it to App Engine, and schedule a cron task queue in App Engine to execute the pipeline periodically.

Dataflow supports side inputs, which allows different transformations on data in the same pipeline.

TIP

For Cloud Dataflow users, use roles to limit access to only Dataflow Resources, not just the Project.

- The Pipeline

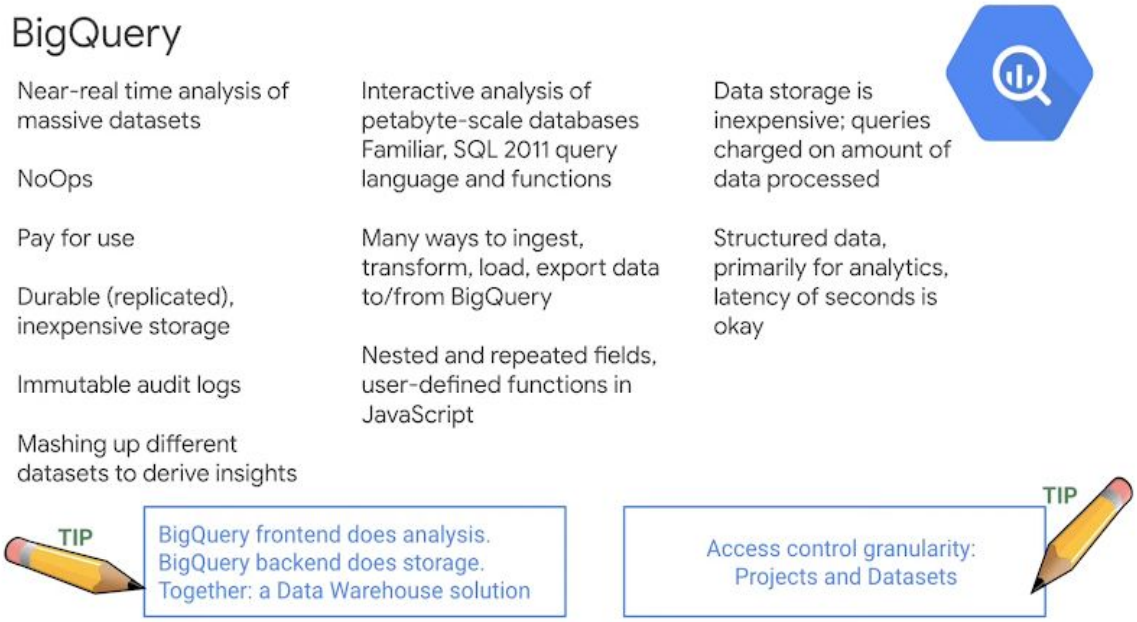
Dataflow pipeline - documented - easier to maintain

```
Pipeline p = Pipeline.create();
p
  .apply(TextIO.read().from("gs://...”))
  .apply(ParDo.of(new Filter1()))
  .apply(new Group1())
  .apply(ParDo.of(new Filter2())
  .apply(new Transform1())
  .apply(TextIO.write().to("gs://...”));
p.run();
```

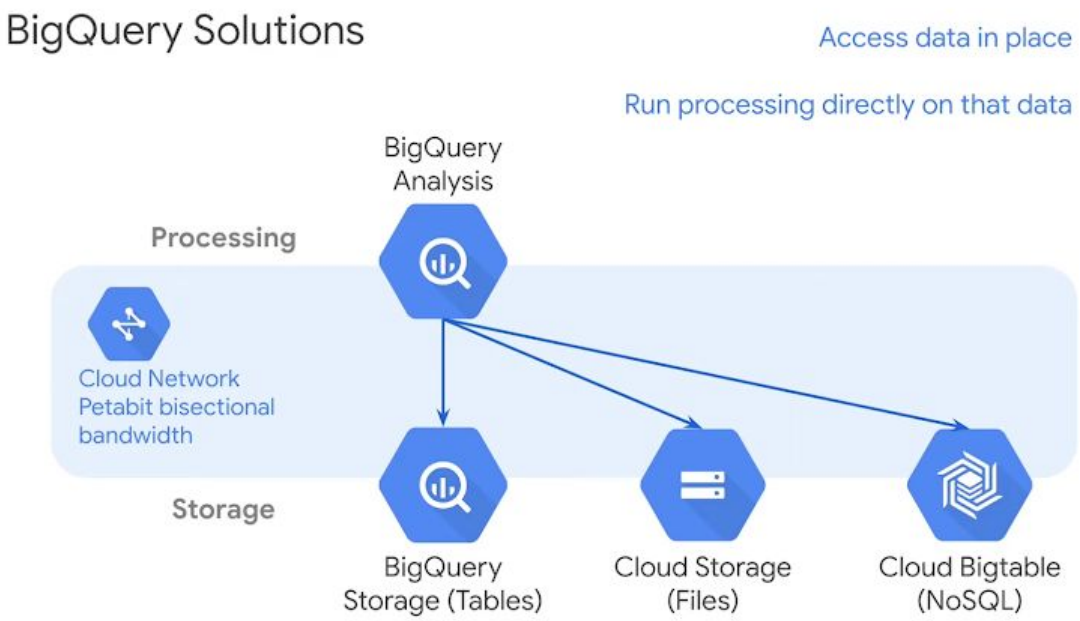
```
class Filter1 extends DoFn<...> {
  public void
  processElement(ProcessContext c)
{
      ... = c.element();
      ...
      c.output(...);
  }
```

- Operations
  - **ParDo** - allows for parallel processing
  - **Map** - 1:1 relationship between input and output in Python

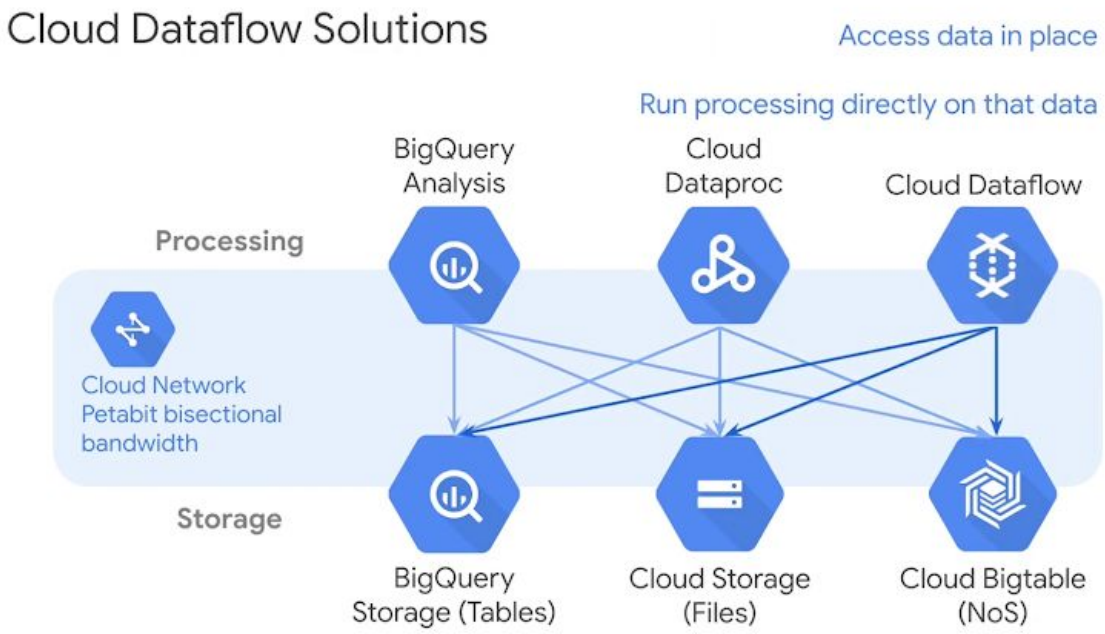
        - **FlatMap** - For non 1:1 relationships, usually with a **GENERATOR** in Python
        - **.apply(ParDo)** - Java for Map and FlatMap
        - **GroupBy** - Shuffle
        - **GroupByKey** - Explicitly shuffle
        - **Combine** - aggregate values
    - Pipelines are often organized into **Map and Reduce** sequences
    - **Side inputs** - parallel paths of execution for making different transformations on the same source data
    - A **pipeline** is a more maintainable and less error-prone way to organize data processing code
    - Template workflow supports non developer users
        - Developer **executes pipeline** on Dataflow
        - Dataflow stores template in **cloud storage**
        - Users submit templates to run **jobs**
        - Opens up new options for **separation of work** for better security and resource accountability
- BigQuery and Cloud Dataflow Solutions
    - Overview



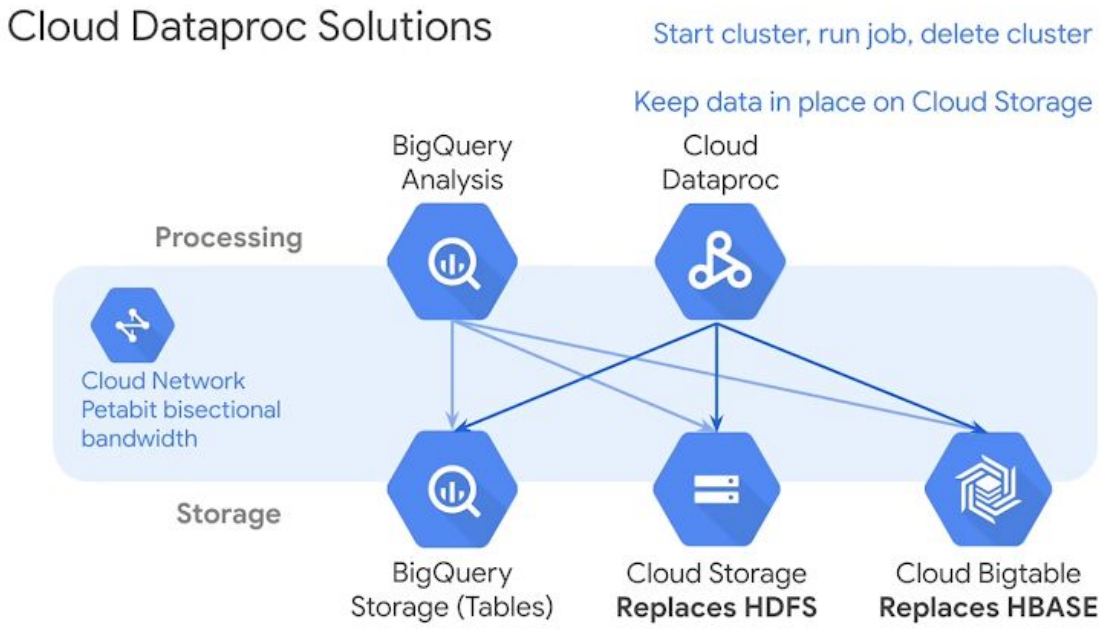
        - **Structured data** takes a few seconds to query
        - **Data Warehouse** solution
        - Access control granularity is at the **project and dataset** level
        - BQ frontend does **analysis**
        - BQ backend does **storage**



    - Keep data **in place** on Cloud Storage
        - I.e. the data is immutable
- Cloud Dataflow Solutions

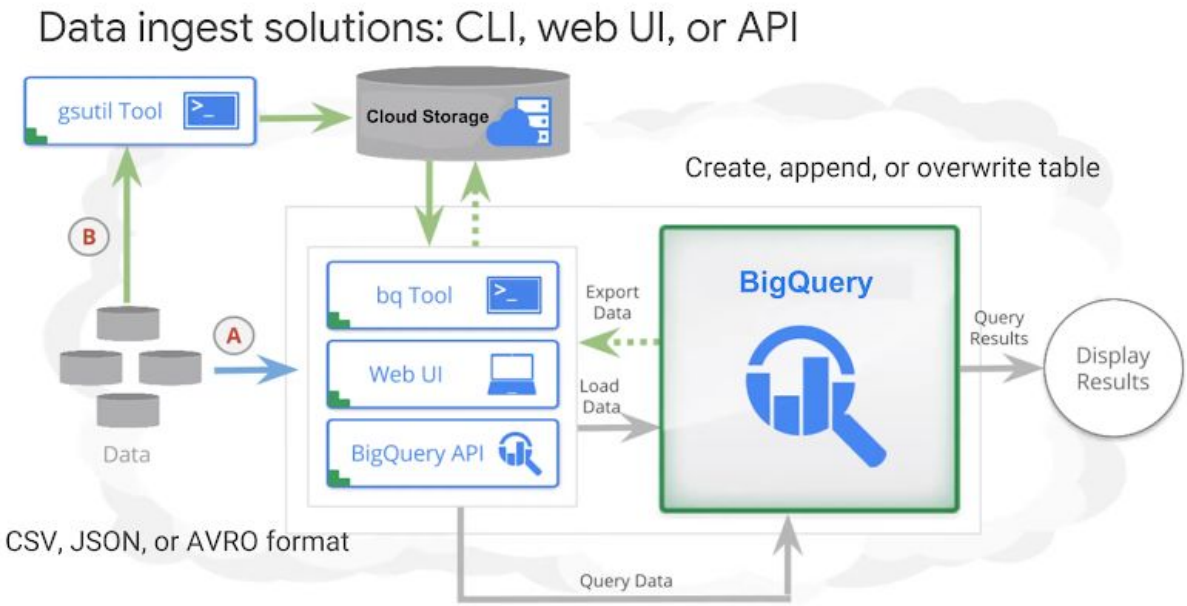


- Cloud Dataproc Solutions



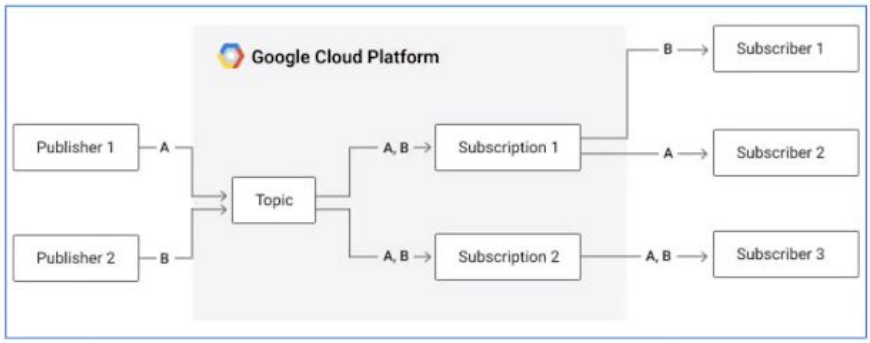
- X. Design Data Processing Infrastructure

- Data ingest solutions
  - Three ways of **loading data** into BigQuery
    - **Files on disk or Cloud Storage**
    - **Stream data**
    - **Federated Data source**
  - Options for transferring storage:
    - **Gsutil** - use this when transferring data from an on-premise location
    - **Storage Transfer Service** - use when transferring data from another cloud storage provider
    - **Transfer Appliance** - use when data is too big to transfer electronically



- XI. Cloud Pub/Sub solutions
- Overview

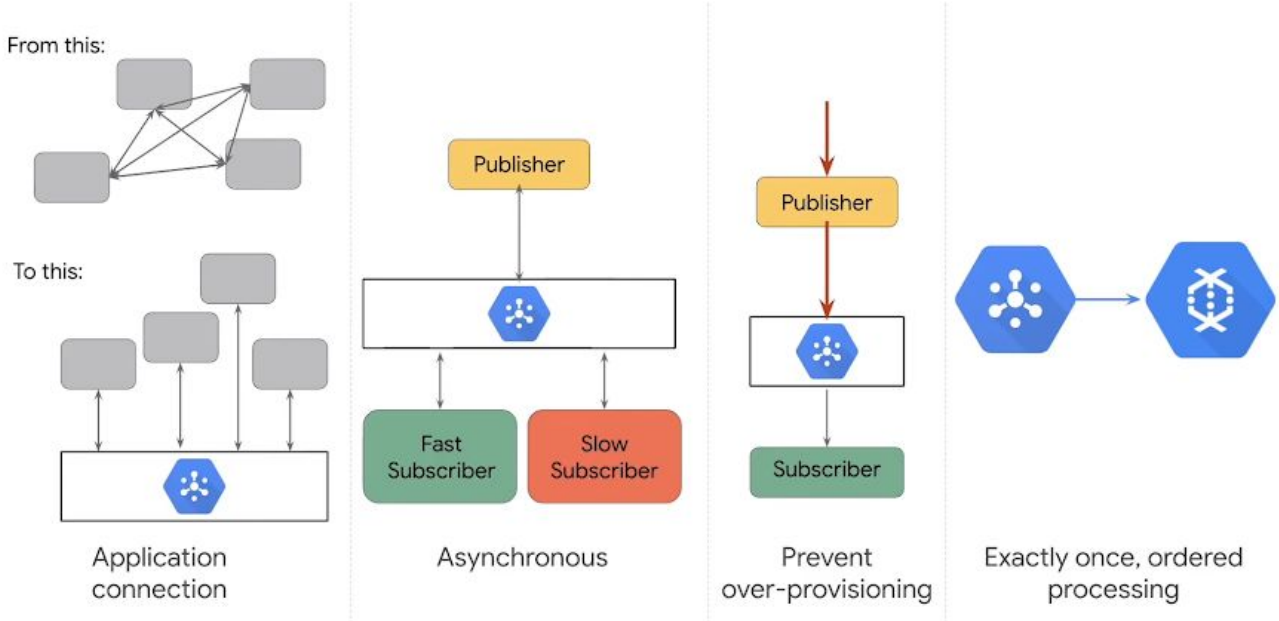
Cloud Pub/Sub



Cloud Pub/Sub provides a NoOps, serverless global message queue
Asynchronous; publisher never waits, a subscriber can get the message now or anytime within 7 days
At least once delivery guarantee
Push subscription and pull subscription delivery flows
100s of milliseconds — fast

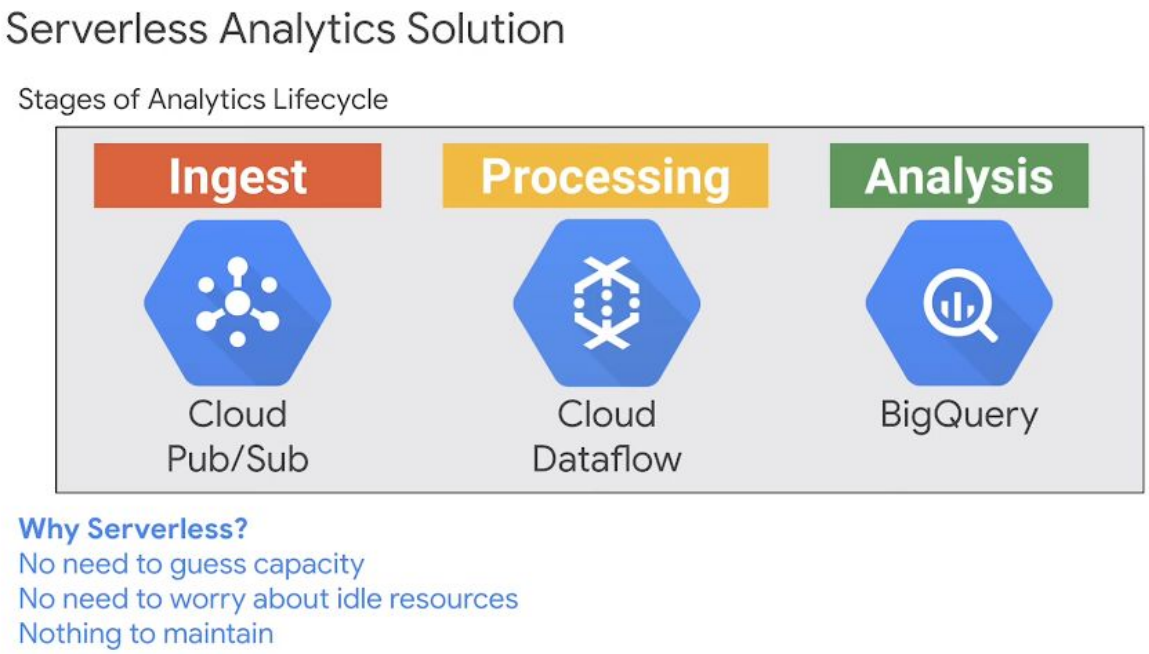
- PubSub holds messages up to **7 days**
- Common Applications of Cloud Pub/Sub

Common applications of Cloud Pub/Sub



- PubSub handles **exactly once delivery** while Cloud Dataflow handles de-duplication, ordering, and windowing

- Separation of duties enables scalable solutions
- Serverless Analytics Solution



- Example use case: mobile gaming platform
- XII. Designing Data Processing Systems Tips
    - Be familiar with the common use cases and qualities of the different storage options
    - Cloud Datalab, Google Data Studio, and BigQuery all have interactive interfaces. When should you use each?
        - Cloud Data Studio focuses on **visualization**. You **do not need to code**.
        - Cloud Datalab focuses on **data exploration**, analysis, creating ML models, and visualization. You **need to code**.
            - It is based on the Jupyter framework
    - Be familiar with the common assemblies of services and how they are often used together: Cloud Dataflow, Cloud Dataproc, BigQuery, Cloud Storage, and Cloud Pub/Sub
    - Technologically, Cloud Dataproc is superior to open-source Hadoop, and Cloud Dataflow is superior to Cloud Dataproc. **However, this does not mean that the most advanced technology is always the best solution; you need to consider the business requirements**. The client might want to first migrate from the data center to the cloud. Make sure everything is working (validate it) . And only after they are confident with that solution, consider improving or modernizing.
- XIII. Practice Exam Questions

- Storage for CSV files. Analysts will run ANSI SQL queries. Support complex aggregate queries and reuse existing I/O-intensive custom Apache Spark transform. How should you transform the input data>
    - Use BigQuery for Storage. Use Cloud Dataflow to run the transformations
    - Use BigQuery for Storage. Use Cloud Dataproc to run the transformations
    - Use Cloud Storage for storage. Use Cloud Dataflow to run the transformations
    - Use Cloud Storage for storage. Use Cloud Dataprof to run the transformations
- Streaming log messages will be stored in a searchable repository. How should you set up the input messages? Requirements are final result message ordering, stream input for 5 days, and querying of the most recent message value
    - Use Cloud Pub/Sub for input. Attach a timestamp to every message in the publisher
    - Use Cloud Pub/Sub for input. Attach a unique identifier to every message in the publisher
    - Use Apache Kafka on Compute Engine for input. Attach a timestamp to every message in the publisher
    - Use Apache Kafka on Compute Engine for input. Attach a unique identifier to every message in the publisher
- Apache Kafka is overly complex as compared to Cloud Pub/Sub

- XIV. Data Representations, Pipelines, and Processing Infrastructure
    - Storage
        - Tip 1: Ask yourself if the data is **useful as is or if it needs to be transformed or cleaned**
        - Tip 2: Ask yourself if this is going to be an **ongoing process or one time activity**
        - Attempting a **schema design on unstructured data** could be useful to identify which parts of the data have some kind of order

Topic	Tips
Future advances in data technology	Tradeoffs between common formats and efficient formats.
Changes to business requirements	Notice business changes that imply possible solution changes.
Awareness of current state and how to migrate the design to a future state	What state/format is the data in and for what purposes, and where does it need to go, for what new purpose, and in what new state/format?
Data modeling	How do data items relate to one another?
Tradeoffs	JSON, XML, AVRO, CSV, Databases: SQL, NOSQL, ...
Distributed systems	Main points: Can the application deal with out-of-order data? Eventual consistency? Can the application handle the overhead and delay of ordering or synchronizing?
Schema design	If the data is structured: "organized for a purpose."

- B. Pipelines
- Tip 3: **Revisit data representation** to make sure that the pipeline is efficient

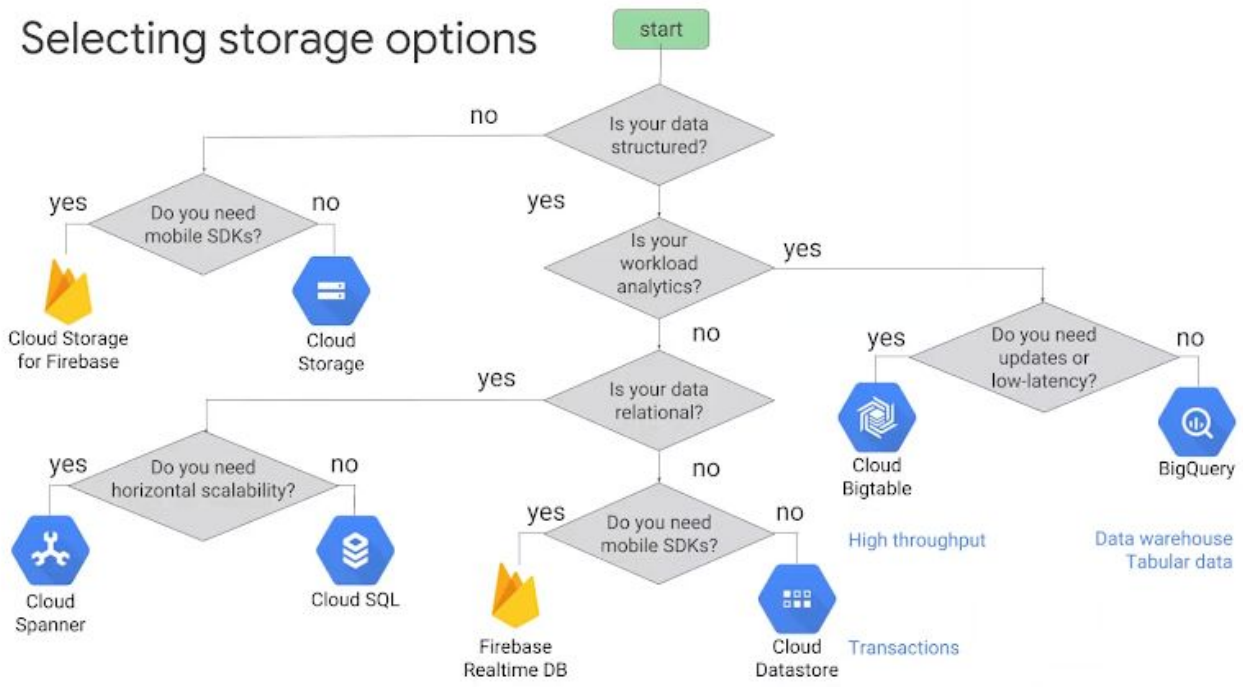
Topic	Tips
Future advances in data technology	Proprietary versus open? Proprietary may mean faster advances in production innovation.
Changes to business requirements	Can the pipeline be modified to fit the need? How extensive is the change required?
Awareness of current state and how to migrate the design to a future state	Can you push the change earlier in the flow? For example, using ETL? Or does the solution need to be dynamic?
Data modeling	Diagram the data flow and transitions.
Tradeoffs	There might be multiple valid methods to arrive at the result, but not all scale, perform, adapt the same. What qualities are desired/valued?
System availability	Is there an SLA? If not, what is your SLA and how will you support that?
Distributed systems	Better scale but complex coordination. Is it worth it? Migration to a new scalable platform versus simplicity?
Schema design	More... pipeline processing design... beware of inefficient equivalents.
Common sources of error (e.g., removing selection bias)	Such as... recency bias... loads data on one server and hurts performance.

- C. Processing Infrastructure
  - Tip 4: There might be **more than one way** to get the results

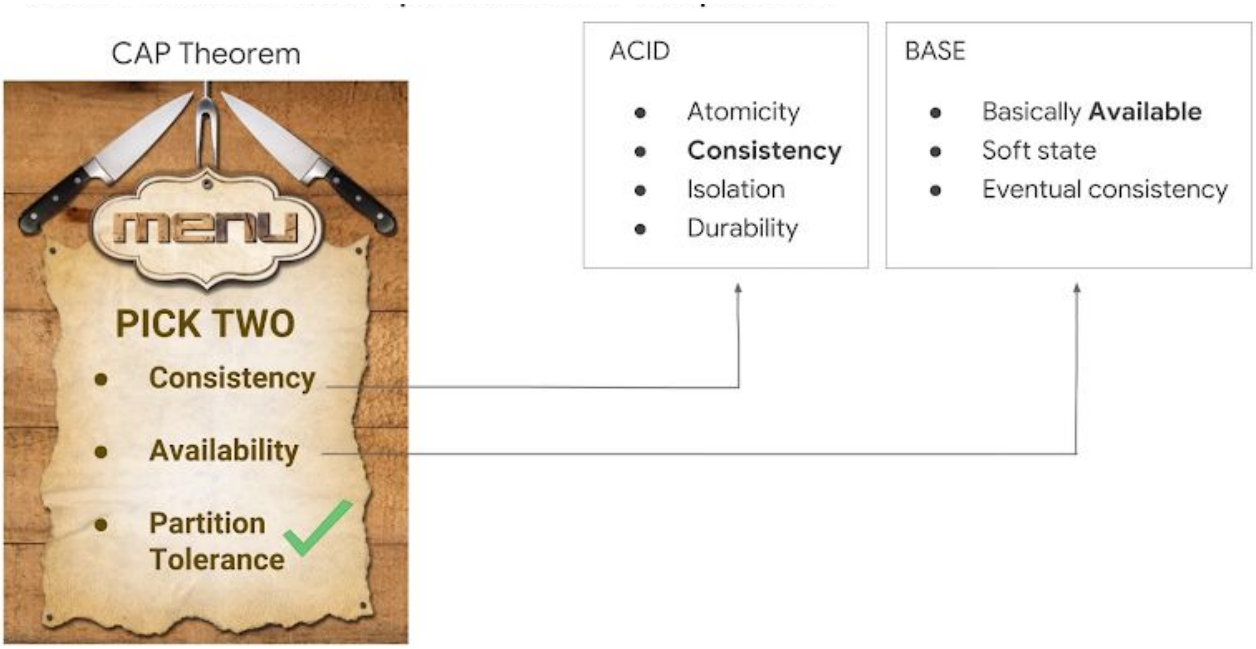
Topic	Tips
Future advances in data technology	How important are present qualities versus future flexibility to the business?
Changes to business requirements	If the requirements change, will the service support it?
Awareness of current state and how to migrate the design to a future state	What future state might be required, and how can you keep that path/option open?
Data modeling	Can the pipeline use platform-specific methods for better results?
Tradeoffs	Proprietary versus open? Support versus lock-in?
System availability	What if the service is unavailable? Time issues? Alternative plans?
Distributed systems	Performance, delay, complexity?
Schema design	Is the schema optimal for the platform? Ex: unstructured or structured ?
Capacity planning	An ongoing process. Initial and adapting up/down scaling? When to reconsider service?
Different types of architectures: message brokers, message queues, middleware, service-oriented	Especially important for IoT, streaming, and coordination of more complex systems/solutions

Part 2: Building and Operationalizing Data Processing Systems

- Building Data Processing Systems
  - Building databases



- Building and maintaining flexible data representations
  - **Databases** are systems for the storage and retrieval of information
    - One approach is to previously define a complex structure that is tailored to the intended operations. This approach may be **faster and more efficient for queries and reports**
    - Another approach is to retrieve information from documents geared towards the business logic. This approach may be **easier for data entry and maintenance**
  - **Too much data division = too much additional work**
  - What transaction qualities are required?



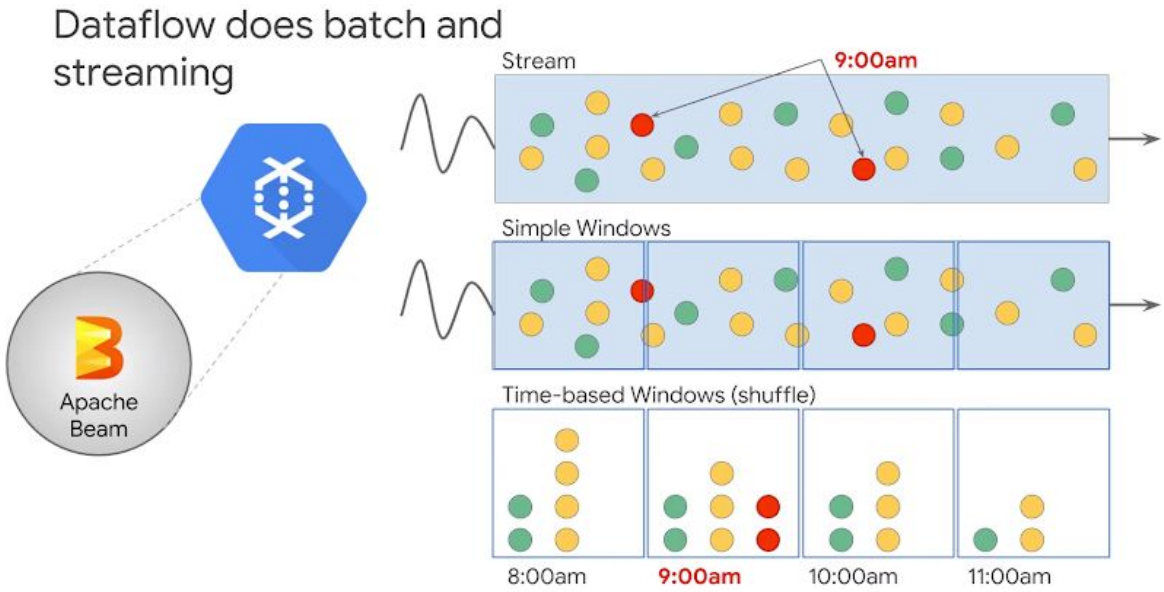
- ACID
  - Stands for
    - **Atomic** - everything in a transaction succeeds or the entire transaction is rolled back
    - **Consistent** - a transaction cannot leave the database in an inconsistent state
    - **Isolated** - transactions cannot interfere with each other
    - **Durable** - completed transactions persists, even when servers restart
  - Incompatible with **availability and performance** in very large systems
  - The **CAP theorem** states that if you want consistency, availability, and partition tolerance, you have to settle for two out of three
- BASE
  - Stands for
    - **B**asic **A**vailability
    - **S**oft-state
    - **E**ventual consistency
  - Permits **stale data**
  - Permits **approximate answers**
  - **Consistent state**
- Decide which method best suits your needs
- Can also use a hybrid method
- More [in this blog post]

            - More in this video. It compares SQL Server and Hadoop according to the ACID vs BASE paradigm
        - In cloud datastore there are only 2 api's that offers strong consistency
            - Lookup by key method
            - Ancestor query
- Cloud Storage, Cloud SQL, and Cloud BigTable
    - Cloud Storage
        - **Cluster node options**
            - Persistent Storage
            - Staging area for other services
            - Storage classes
        - **Access**
            - Granular access control
                - Project
                - Bucket
                - Object
            - IAM roles
            - ACLs
            - Signed URLs
        - **Features**
            - Versioning
            - Encryption options
                - Default
                - CMEK
                - CSEK
            - Lifecycles
            - Change storage class
            - Streaming
            - Data
            - Transfer/synchronization
            - Storage Transfer Service
            - JSONa and XML APIs
        - **Best Practices**
            - Traffic estimation
    - Cloud SQL
        - **Familiar structure**
            - Similar to MySQL
        - **Not supported**
            - User Defined Functions (UDF)
            - MySQL-esque replication, statements, and functions related to files and plugins
        - **Flexible Pricing**
            - Per use
            - Per hour

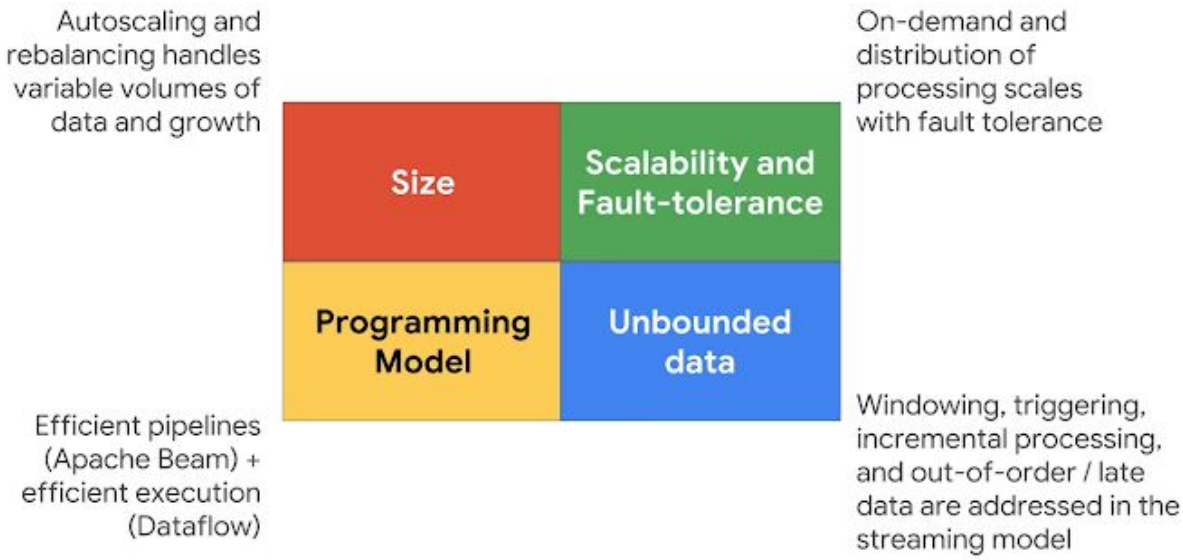
        - **Connect from anywhere**
            - Use the static IP address
            - SQL connector libraries
        - **Fast**
        - **Google Security**
    - Cloud BigTable
        - Meant for **high throughput data**
        - **Millisecond latency**
        - **NoSQL**
        - Access is optimized for **Row Key prefixes**
        - Schema design and time series support
        - Access control
        - Performance design
        - Choose between SSD and HDD
        - Tables should be **TALL and NARROW**
    - **Cloud Spanner**
        - Global
        - Fully managed RDB with transactional consistency
        - Strongly typed
            - Must define a schema
        - Schema design
        - Data model
        - Updates secondary indexes
        - Data types
        - Transactions
        - Timestamp bounds and commit timestamps
        - Use cloud spanner if you need globally consistent data or more than 1 Cloud SQL instance
    - **Cloud Datastore**
        - NoSQL
        - Atomic transactions
            - ACID support
        - High availability of reads and writes
        - Massive scalability with high performance
        - Flexible storage and querying of data
        - Balance of strong and eventual consistency
        - Encryption at rest
        - Fully managed with no planned downtime
        - IAM
        - Storage Size calculations
        - Multitenancy
        - Encryption
    - Comparison

	Cloud Datastore	Cloud Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Best for	Getting started, App Engine applications	"Flat" data, Heavy read/write, events, analytical data	Structured and unstructured binary or object data	Web frameworks, existing applications	Large-scale database applications (> ~2 TB)	Interactive querying, offline analytics
Use cases	Getting started, App Engine applications	AdTech, Financial and IoT data	Images, large media files, backups	User credentials, customer orders	Whenever high I/O, global consistency is needed	Data warehousing

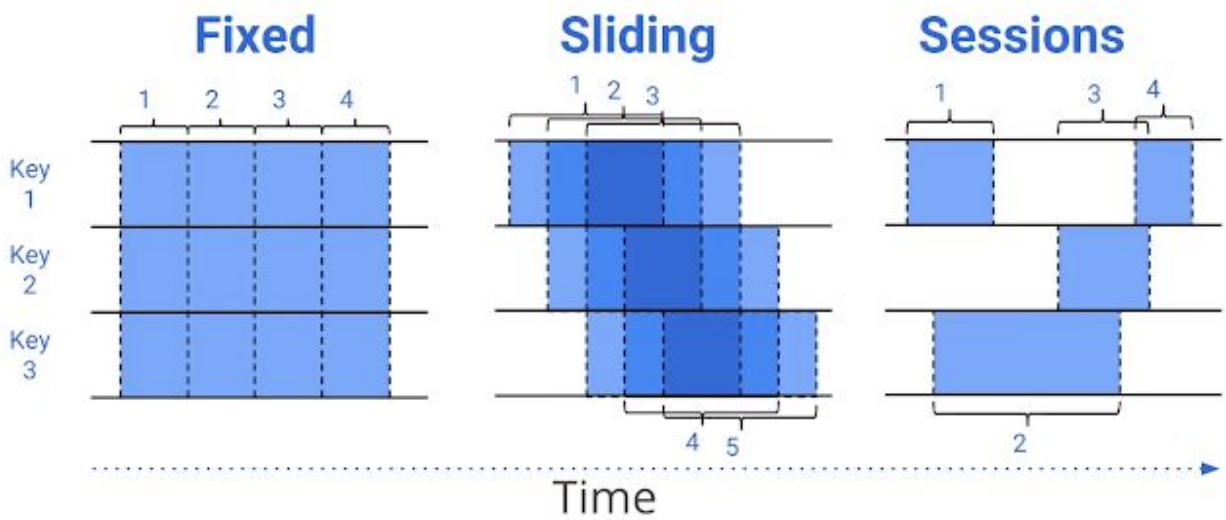
- Building and Operationalizing Pipelines
  - Dataflow does batch and streaming
    - It is **apache beam as a service**



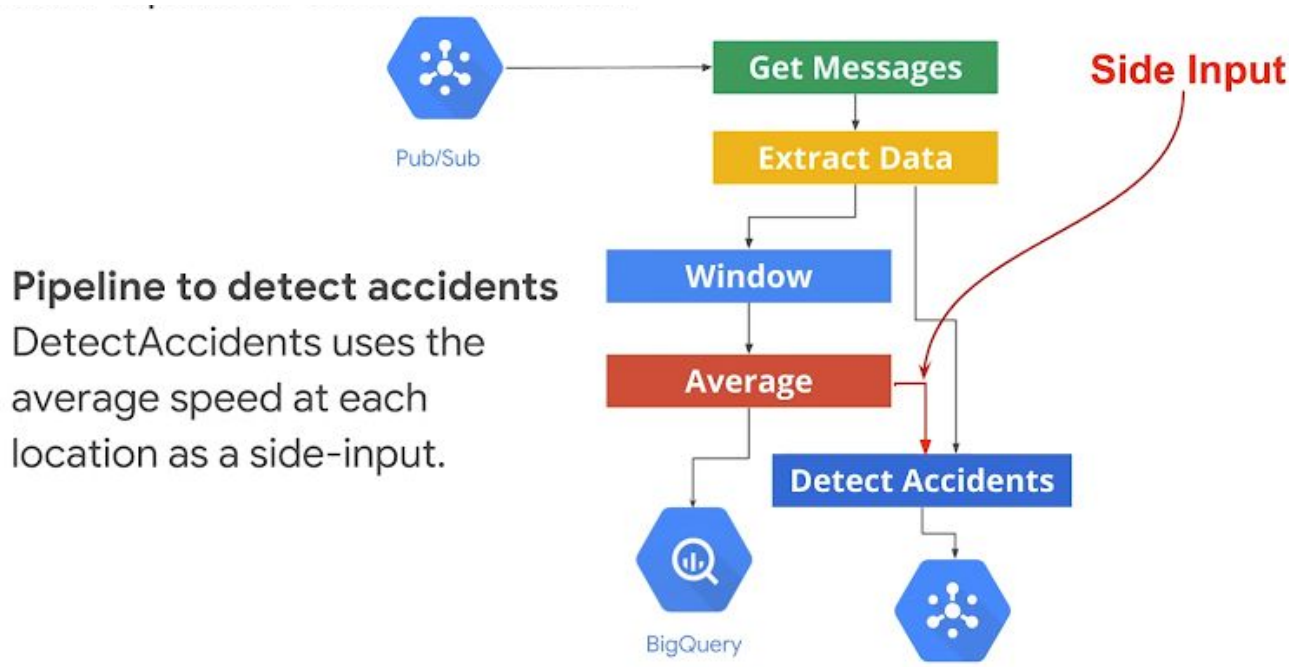
    - **Solves many stream processing issues**



- To compute averages on streaming data, we need to b**ound the computation within time-windows**
- **Triggering** controls how results are delivered to the next transforms in the pipeline
- **Watermark** is a heuristic that tracks how far behind the system is in processing data from the event time
- **Window types**
    - Fixed
    - Sliding
    - Session-based
- Updated results (late)
- Speculative results (early)
- Windows are the answer to **"where in event time?"**
    - Divides data into event-time-based finite chunks
    - Often required when doing **aggregations over unbounded data**



- Side inputs
  - **Additional inputs** to ParDo apart from the main input Pcollection
  - When you specify a side input, you create a view of some other data that can be read from within the ParDo transform's DoFn while processing each element
  - Useful if your ParDo needs to **inject additional data** when processing each element in the input PCollection



  - More [information from the official Apache Beam Programming Guide]

- Practice Exam Questions 2
  - An application that relies on Cloud SQL to read infrequently changing data is predicted to grow dramatically. How can you increase capacity for more read-only clients?

        - Configure high availability on the Master node
        - Establish an external replica in the customer's data center
        - Use backups, so you can restore if there is an outage
        - Configure read replicas
    - A BigQuery dataset was located near Tokyo. For efficiency reasons, the company wants the dataset duplicated in Germany
        - Change the dataset from a regional location to multi-region location, specifying the regions to be included
        - Export the data from BigQuery into a bucket in the new location and import it into a new dataset at the new location
        - Copy the dataset in source region to the dataset in the target region using BigQuery commands
        - Export the data from BigQuery into a nearby bucket in Cloud Storage. Copy to a new regional bucket in Cloud Storage. Import into the new dataset in the new location
            - Important points
                - A is wrong because datasets are immutable, so the location can't be updated
                - B is wrong because BigQuery writes and reads from nearby buckets, so the new location can't read the old location data
                - C is wrong because BigQuery doesn't provide a location-to-location move or copy command
    - A transactionally consistent global relational repository where you can monitor and adjust node count for unpredictable traffic spikes
        - Use Cloud Spanner. Monitor storage usage and increase node count if more than 70% utilized
        - Use Cloud Spanner. Monitor CPU utilization and increase node count if more than 70% utilized for your time span
        - Use Cloud Bigtable. Monitor data stored and increase node count if more than 70% utilized
        - Use Cloud Bigtable. Monitor CPU Utilization and increase node count if more than 70% utilized for your time span
            - Important points
                - B is correct because of the requirement to globally scalable transactions--Use cloud spanner. CPU utilization is the recommended metric for scaling, per Google best practices, linked below.
                - A is not correct because you should not use storage utilization as a scaling metric
                - C and D are not correct because you should not use Cloud BigTable for this scenario
- Building and Operationalizing Pipelines and Processing Infrastructure

Topic	Tips
Data cleansing	What kinds of quality controls are there on the source data?
Batch and streaming	What frequency of results is needed?
Transformation	Is the data in a usable structure, or does it need to be transformed for practicality or efficiency?
Acquire and import data	How will data be acquired or imported?
Testing and quality control	How will the quality of the data and results be validated? Metrics?
Connecting to new data sources	Are there new data sources that will be required, and how will that data be made available to the solution? Frequency? Validation? Quantity? Connection method? Security?

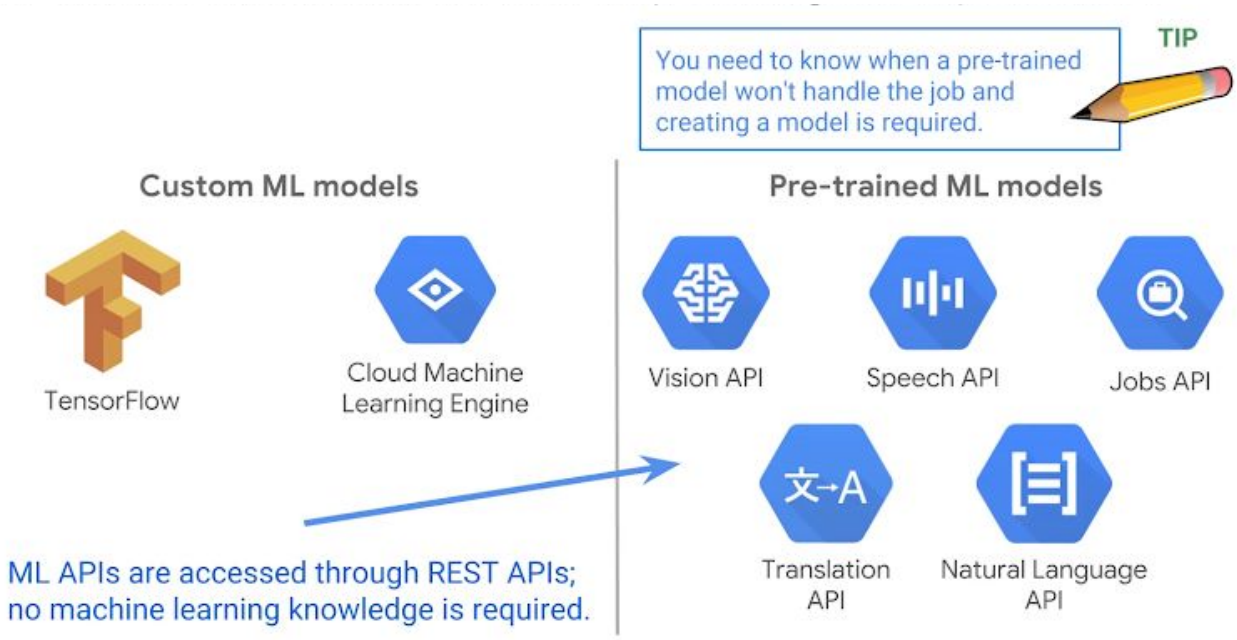
Topic	Tips
Provisioning resources	Fixed or dynamic? How will resources be provisioned initially, and how will they be adjusted? Frequency of adjustment? Method?
Monitoring pipelines	How will workflow be monitored? For example, in Cloud Dataflow, the pipeline is visible in the console. How do capabilities connect with procedures?
Adjusting pipelines	What happens when the source data changes or the business circumstance changes? Who will make changes of what kind?
Testing and quality control	How do you know the pipeline is continuing to do what it is supposed to do and operating within parameters? What happens if there is a change in the flow of data?

- Data management is **often influenced by business requirements**. After the data has been used for the "live" application, is it collected for reporting, for backup and recovery, for audits, or for legal compliance? What are the changing business purposes of the data in different time frames?
- "Effective use of managed services": Choose the right service and the correct settings/features for specific use cases. Consider **costs, performance, and effective use cases** (key features) for these:
  - Cloud BigTable
  - Cloud Spanner
  - Cloud SQL
  - BigQuery
  - Cloud Storage
  - Cloud Datastore
  - Cloud Memorystore

    - What is Data cleansing? **Data cleansin**g is improving the data quality through consistency. You could use **Cloud Dataprep** to Extract, Transform, or Load (ETL). You could run a data transformation job on Cloud **Dataproc**.
    - Batch and streaming together? You should already be thinking "**Cloud Dataflow**."
    - Integrating with new data sources? You should be familiar with the connectors available between services in the cloud and common import/acquisition configurations.
    - Testing and quality control and monitoring: You should be familiar with the common approaches to testing that are used in production environments, such as A/B testing, and other rollout scenarios. Similarly, there are operational and administrative monitoring elements for most services in the GCP Console, and statistical and log monitoring in Stackdriver. Do you know how to enable and use Stackdriver with common services?
        - **A/B Testing**
            - Statistical way of comparing two or more versions (A and B)
            - Figure out the **complexity of customer behavior**
        - **Stackdriver**
            - Now referred to as **Operations**
            - Umbrella term that encompasses Monitoring, Logging, etc.
            - See this video of a short demo
            - For collecting metrics, logs, and traces across Google Cloud and your applications
            - Set up alerts and notifications
- Quiz
    - A common assembly of technologies for Date Engineering is:
        - Cloud Dataproc, Cloud SQL, and Cloud Datastore
        - Cloud Pub/Sub, Cloud Dataflow, and BigQuery
    - When you prepare using tables that compare different technologies
        - Read from the table-up so that if you see a keyword in a question, you will recognize which associated technology in the heading row is a candidate for the solution
        - You don't really need to know the characteristics of each technology. So it is safe to ignore tables. They have too much information anyway
    - What are the three kinds of streaming windows discussed?
        - Fixed, Sliding, and Sessions
        - Elastic, Average, and Wide-column

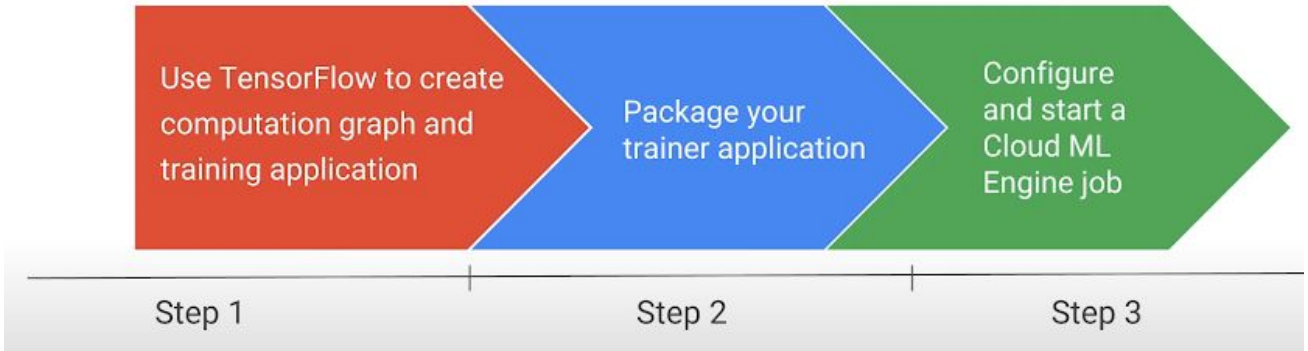
Part 3: Operationalizing Machine Learning Models

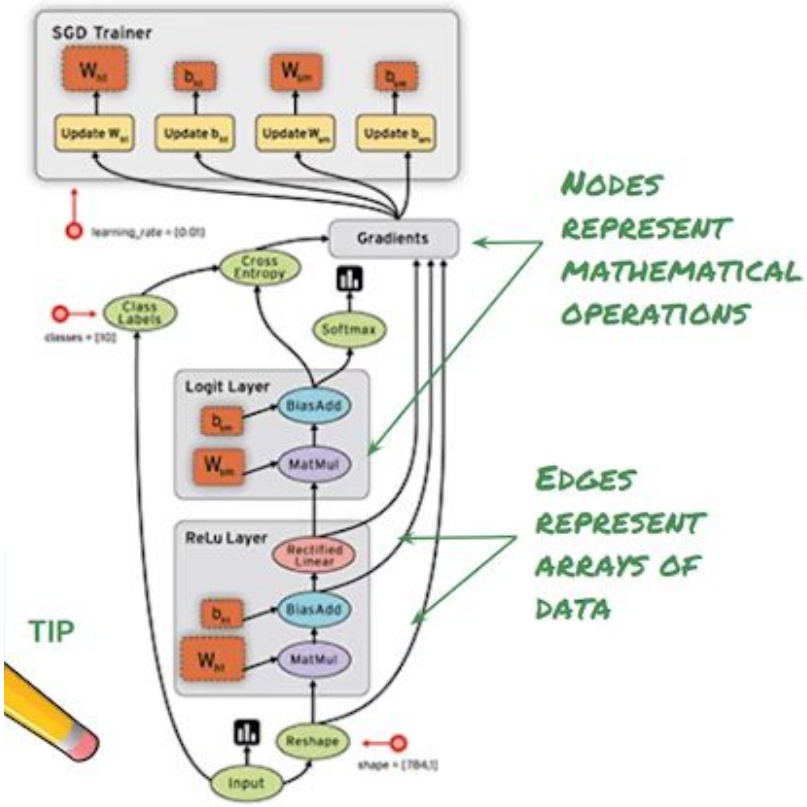
- Analyzing and Modeling
    - Pretrained models are a fast way to magical experiences



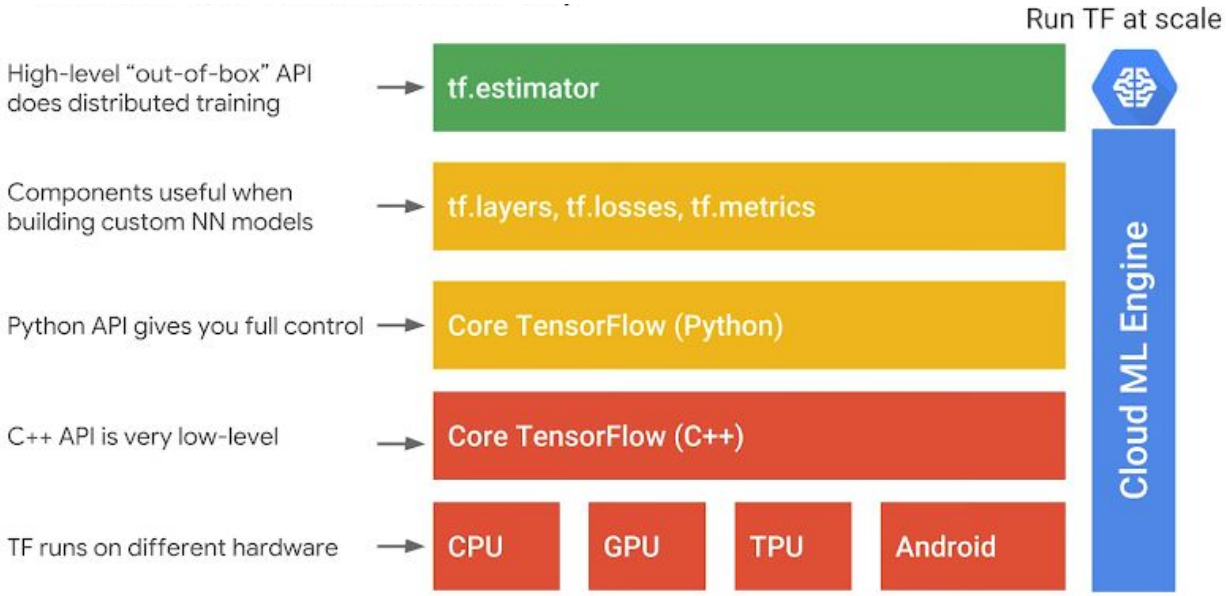
- Deploying an ML Pipeline
  - Data analysis and ML are carried out in self-descriptive, shareable, executable notebooks
    - Typically contains code, charts, and explanations
  - In Cloud Datalab, start locally on sampled dataset
  - Then scale it out to GCP using serverless technology

  - Training your model with Cloud ML Engine

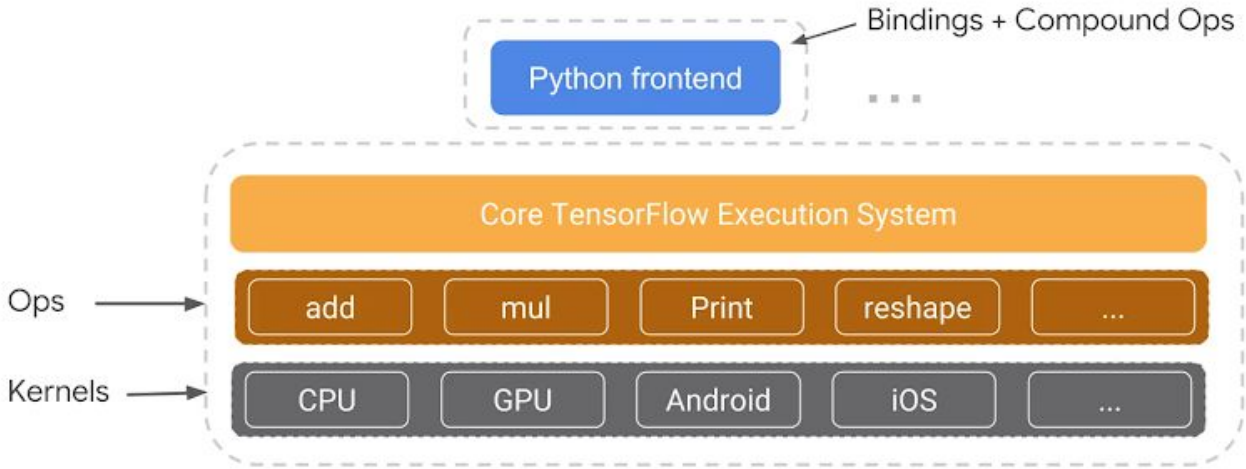
- 

- Machine Learning
  - Steps:
    - Collect data
    - Organize data
    - Create model
    - Use machines to flesh out the model from data
    - Deploy fleshed out model
  - Tensorflow is an open-source high-performance library for numerical computation that uses directed graphs
    - 

  - Tensorflow toolkit hierarchy



- The Python API lets you build and run directed graphs
- Tensorflow does lazy evaluation
- You can use it in eager mode but it is typically not used in production environments
- Graphs can be processed, compiled, remotely executed, and assigned to devices
- Tensorflow can distribute computation

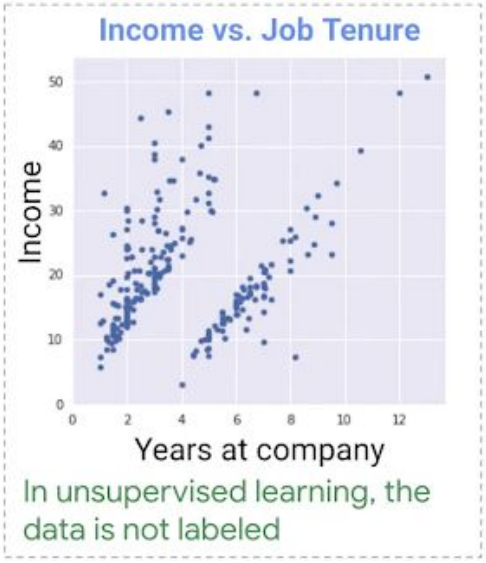


- ML and Unstructured Data
  - What qualifies as unstructured data?
    - If the schema is not useful for the purposes of the analysis or query
  - Is the problem suited for big data, machine learning, or human decision making?

Human Reasoning	Easy counting problems = big data	Harder counting problems = ML
Real-time insight into supply chain operations. Which partner is causing issues?	Did error rates decrease after the bug fix was applied?	Are programmers checking in low-quality code?
Drive product decisions. How do people really use feature X?	Which stores are experiencing long delays in payment processing?	Which stores are experiencing a lack of parking space?

- Data Engineers must focus on both the training and inference stages of ML
  - Is the question about TRAINING or INFERENCE?
  - Technically, ML is not predicting but is instead using inference
- In supervised learning, you have labels
  - Regression and classification are supervised ML model types

Unsupervised Learning



In unsupervised learning, the data is not labeled

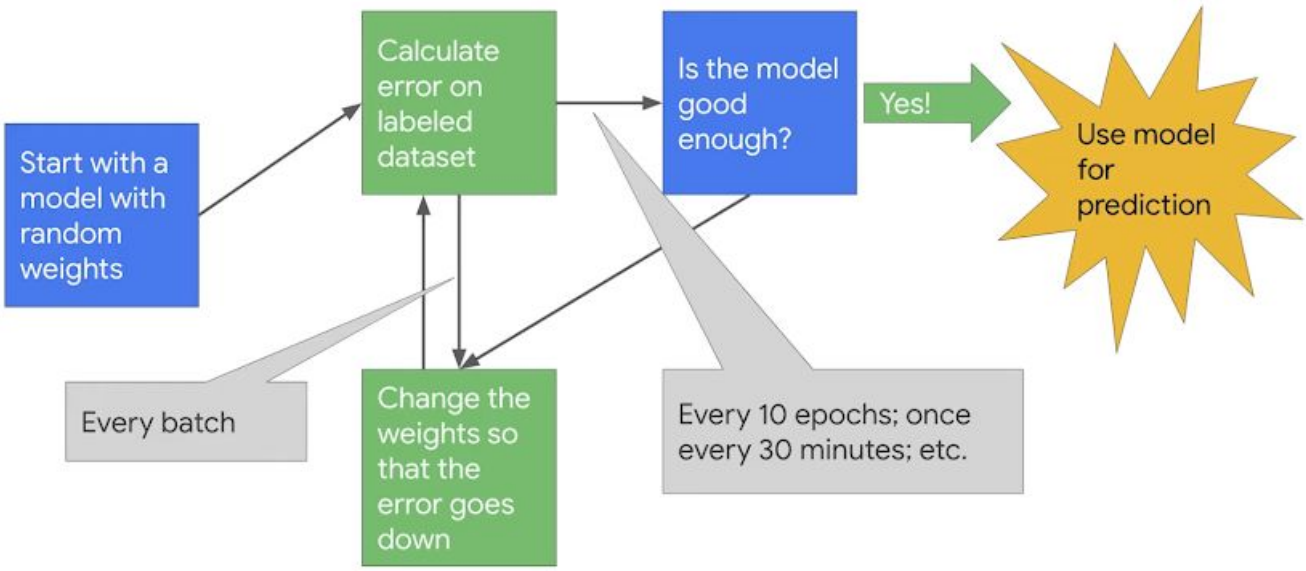
Supervised Learning

In supervised learning, we are learning from past examples to predict future values

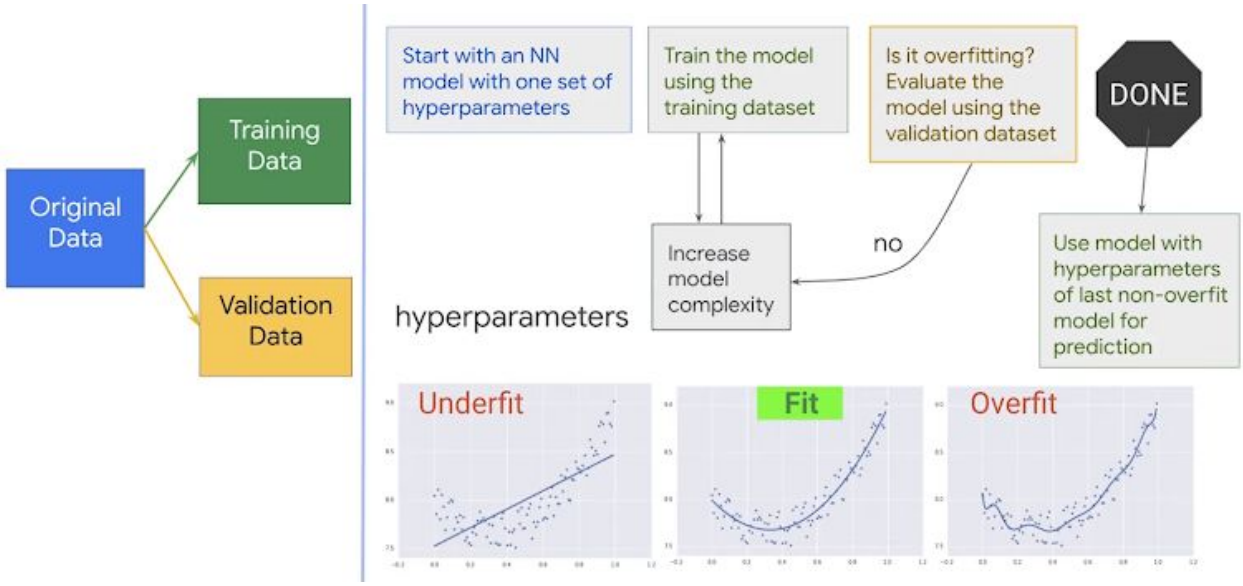
- A data warehouse can be a source of structured data training examples for your ML model
- Mean Square Error is a measure of loss
- Gradient descent is used to find the best parameters
  - How an ML problem is turned to a search problem
- Recompute error after each batch of examples (not full dataset)
  - RMSE
- Use cross entropy for classification problems
  - It is differentiable

- Training and Validating

- Occasionally, evaluate the model to decide whether to stop



- Split data, experiment with models



- Use independent test data or cross-validate with the data is scarce
- Working with Estimator API
  - Set up ML model
    - Regression or classification?
    - What is the label?
    - What are the features?
  - Carry out ML steps
    - Train the model
    - Evaluate the model
    - Predict with the mode

- Practice Exam Questions 3

- Quickly and inexpensively develop an application that sorts product reviews by most favorable to least favorable
    - Build an application that performs sentiment analysis using the Natural Language API. Use the score and magnitude to sort the reviews
        - Notes:
            - Use a pretrained model whenever possible
- Maximize speed and minimize cost of deploying a TensorFlow machine-learning model on GCP
    - Export the trained model to a SavedModel format. Deploy and run your model on Cloud ML Engine
        - Notes
            - Best practice: “just deploy it”

- Machine Learning and Analysis
    - The goal is **practical, not perfect**

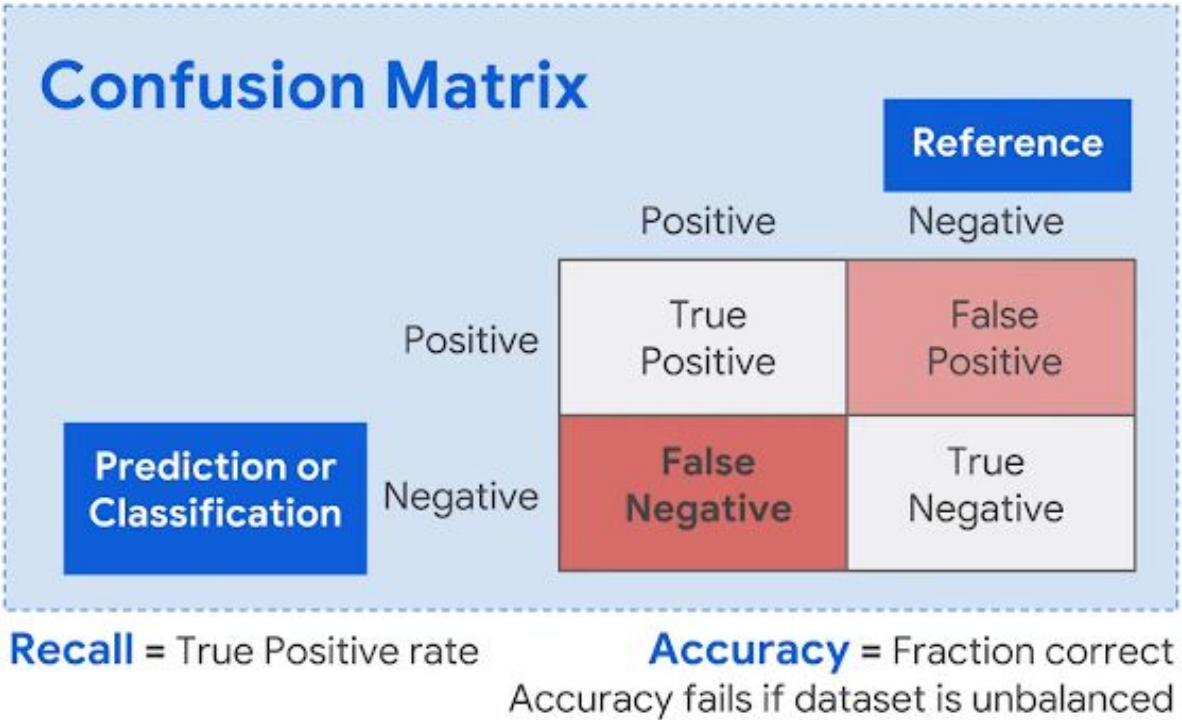
Topic	Tips
Feature selection/engineering	The process of selecting relevant predictors for building a model.
Algorithm selection	Google ML uses general algorithms that are applicable to a wide variety of datasets. Categories: Linear and DNN, Regressor and Classifier.
Debugging a model	Code, data, features, errors. Look for outliers and verify features.

Topic	Tips
Data collection and labeling	What labels are required for your application and/or for training ML? Can the data be labeled as it is being collected? If not, what strategy will you use for labeling data?
Data visualization	Make complex data easier to understand by surfacing trends, outliers, and patterns.
Dimensionality reduction	Reduce random variables being considered, focus on meaningful variables. Feature selection and feature reduction.
Data cleaning/normalization	Normalizing/de-normalizing is a tradeoff between efficiency and performance.
Defining success metrics	Assumed perfection, 100%, often is unrealistic. How will you know that the results are acceptable unless you define a measure or condition of success?

Topic	Tips
Planning (e.g. fault-tolerance)	Persist datasets / pcollections
Executing (e.g., rerunning failed jobs, performing retrospective re-analysis)	Persist
Stress testing data recovery plans and processes	Measure / manage "error budget". Make recovery part of regular activities so you know they are working.

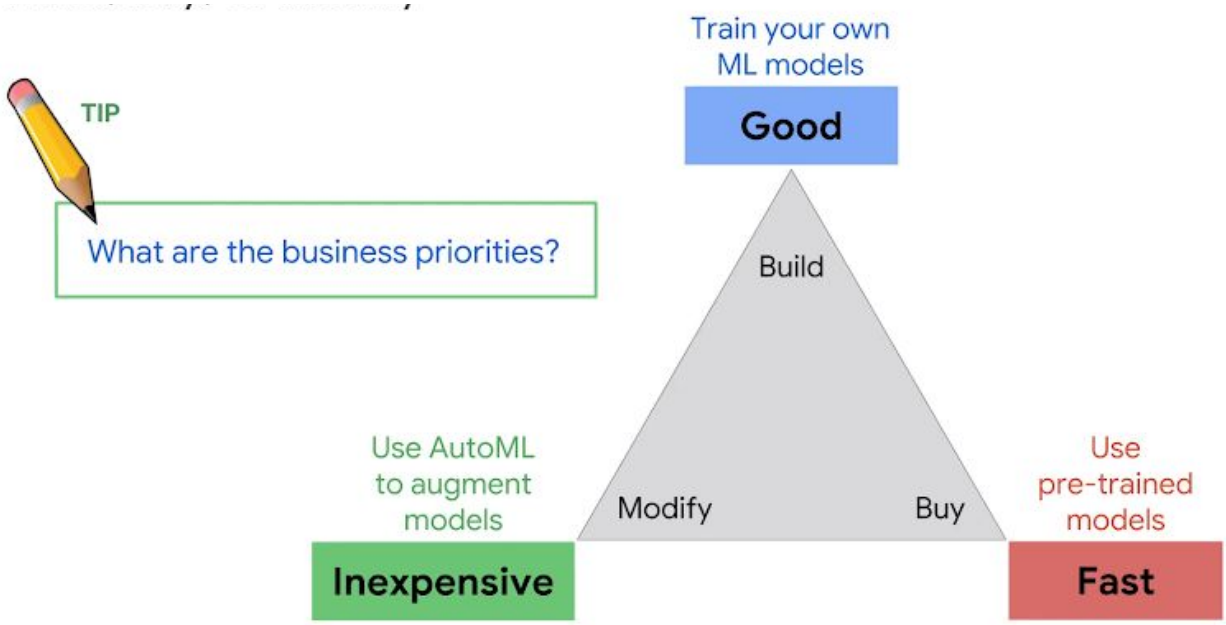
- Exam Guide
    - Leveraging pre-built ML as a service
        - Study these:
            - Cloud Vision API
                - **Pre-trained** machine learning models through REST and RPC APIs
                - Assign labels to images and quickly classify them into millions of predefined categories
            - Cloud Text-to-speech API
                - Convert text into natural sounding speech
            - Cloud Speech-to-text API
                - Accurately convert speech into text
                - Transcribe your content in real time or from stored files
            - Cloud AutoML Vision
                - Automate the training of your own **custom** machine learning models
                - Simply upload images and train custom image models
            - Cloud AutoML Natural Language
                - Reveal the structure and meaning of text
                - Extract information about people, places, and events
                - Train your own high-quality ML custom models to classify, extract, and detect sentiment
            - Natural Language API
                - Pre-trained model for sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis
            - Cloud AutoML Translation
                - Upload translated language pairs and AutoML Translation will train a custom model
            - Translation API
                - Instantly translates text into more than one hundred languages
            - Dialogflow

    - Deploying an ML Pipeline
        - You need to know how to deploy existing models to Cloud ML Engine and to maintain them which might involve training
        - Continuous evaluation is setting up continuous evaluation of the ML model so that steps can be taken to improve it
        - Study these:
            - Kubeflow
            - Cloud ML Engine
            - Spark ML
            - BigQuery ML
    - Choose the appropriate training and serving infrastructure
        - Edge computing is the design of distributing processing in a strategic way so that model processing is pushed closer to the inputs for example in IoT doing Machine Learning processing closer to the IoT sensors by performing work in nearby data centers or regions is edge computing'
        - Study these:
            - GPU
            - TPU
    - Measuring, monitoring, and troubleshooting ML models
        - One common source of error is accidental inclusion of biased data in the data being used for model trainings or validation
        - Define these:
            - Features
            - Labels
            - Models
            - Regression
            - Classification
            - Recommendation
            - Supervised and unsupervised learning
            - Evaluation
            - Metrics
            - Assumptions about data
- Modeling Business Processes for Analysis and Optimization
    - Confusion Matrix classifies kinds of errors



- Use confusion matrix to describe the performance of classification models

- Build, buy, or modify



- Ways to build effective ML
  - Nig Data
  - Feature Engineering
  - Model Architectures
- Consider using data in place of ETL
- Repeat the data and send it along in chunks

        - Grouping the work can be efficient and give additional control over the processing of the data
    - We need to make our ML pipeline more robust

Estimator examples:

1. Ran the training_op for num_steps or num_epochs iterations.
2. Saved checkpoints during training.
3. Used final checkpoint as model.

Realistic, real-world ML models need:

1. Use a fault-tolerant distributed training framework.
2. Choose model based on validation dataset.
3. Monitor training, especially if it will take days.
4. Resume training if necessary.

- Feature Engineering and Performance
    - Overview

1. Related to what is being predicted?
2. Value should be known for prediction
3. Numeric with meaningful magnitude?
4. Enough examples

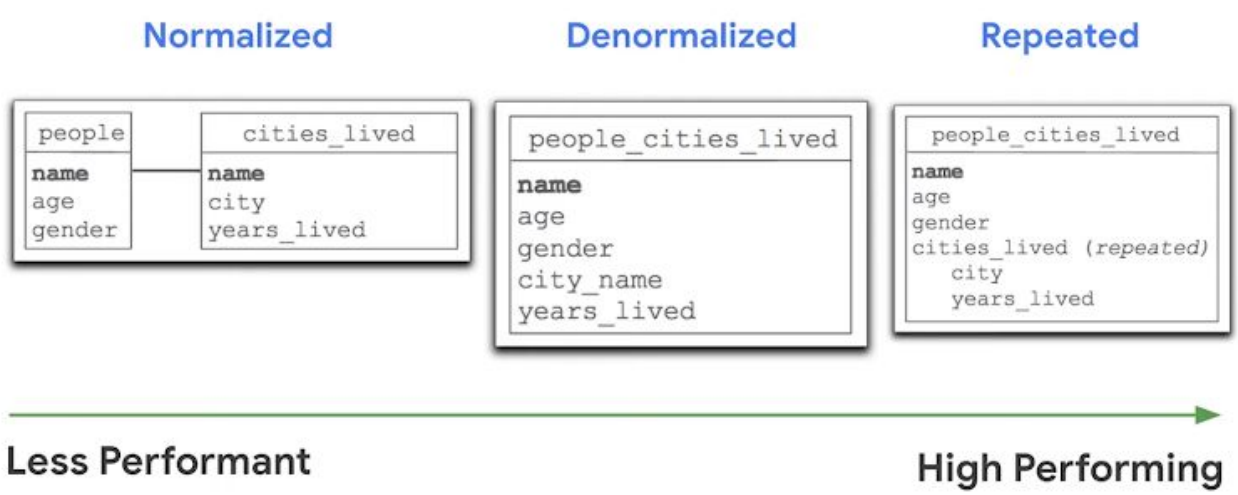
Feature engineering
    Pre-processing
    Feature creation
    Hyperparameter tuning

Good features bring human insight to a problem

Feature crosses
Discretize floats that are not meaningful
Bucketize features
Dense and sparse features
    DNNs for dense, highly correlated
    Linear for sparse, independent

    - Learning rate is an important hyperparameter
        - Small step sizes can take a very long time to converge
            - Though you are guaranteed to find the minimum
        - Larger step sizes may never converge to the true minimum
        - A correct and constant step size can be difficult to find
    - BigQuery Performance
        - Performance is critical to practical solutions

        - Factors:
            - Input data and data sources (I/O)
                - How many bytes does your query read
            - Communication between nodes (shuffling)
                - How many bytes does your query pass to the next stage?
                - How many bytes does your query pass to teach slot?
            - Computation
                - How much CPU work does your query require
            - Outputs (materialization)
                - How many bytes does your query write
            - Query anti-patterns
                - Are your queries following SQL best practices?
- Schema and Performance
    - Wide tables are not very scalable
    - Normalized data
        - Break tables apart according to relation
        - Efficient. Increases orderliness of data
        - Use this when
    - Denormalized data
        - Represents data in a flat format
        - Performant
        - Use repeated fields for higher performance and in parallel using columnar processing
        - Use this when
    - When should you apply each?



Original data

ID	Name	Transaction		
1	Bob			
2	Doug			
		OrderID	Date	Quantity
		11537	10/29/2018	11
		78244	11/05/2018	14
		32367	09/09/2018	18
3	Tom			
		OrderID	Date	Quantity
		13323	04/09/2018	6
		45452	11/10/2018	13
		17671	08/20/2018	15

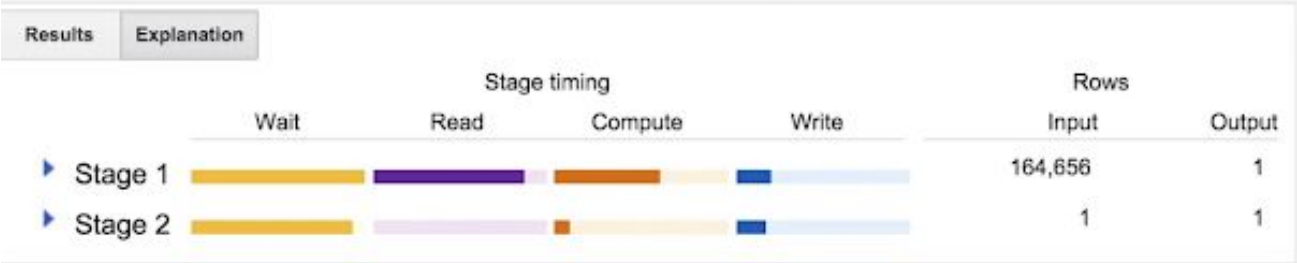
Normalized data

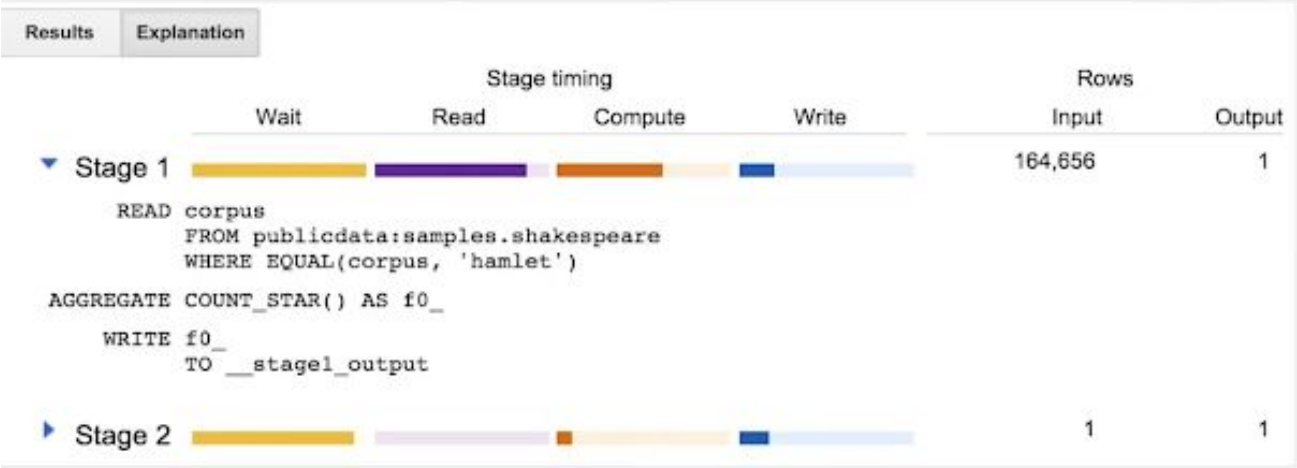
Customer Table	
ID	Name
1	Bob
2	Doug
3	Tom

Transaction Table			
ID	OrderID	Date	Quantity
2	11537	10/29/2018	11
2	78244	11/05/2018	14
2	32367	09/09/2018	18
3	13323	04/09/2018	6
3	45452	11/10/2018	13
3	17671	08/20/2018	15

- BigQuery can use nested schemas for highly scalable queries
- Pipeline and Performance
  - 4 Key Elements of Work
    - I/O
      - How many bytes did your ead?
    - Shuffle
      - How many bytes did you pass to the next stage?
        - Grouping
          - How many bytes do you pass to teach group
    - Materialization
      - How many bytes did you write to storage?
    - CPU work
      - User-defined functions (UDFs), functions
  - Avoid I/O Wastefulness
    - Do not SELECT *, use only the columns you need
    - Filter using WHERE as early as possible in your queries
  - Shuffle Wisely: Be aware of data skew in your dataset
    - Skewed Data creates an imbalance between BQ worker slots (uneven partition sizes)
    - Filter your dataset as early as possible to avoid overloading workers on JOINS
    - Use Query Explanation map and compare the Max vs the Avg times to highligh skew
    - BQ will automatically attempt to reshuffle workers that are overloaded with data
  - Careful use of GROUP BY
    - Best when number of distinct groups is small

        - Grouping by with a high-cardinality unique ID is a BAD idea
    - Joins and Unions
        - Know your join jets and if they are unique. NO accidental cross joins
        - LIMIT wildcard UNIONS with _TABLE_SUFFIX filter
    - Limit UDFs to reduce computational load
        - Use native SQL whenever possible
    - Diagnose performance issues with query explanation map



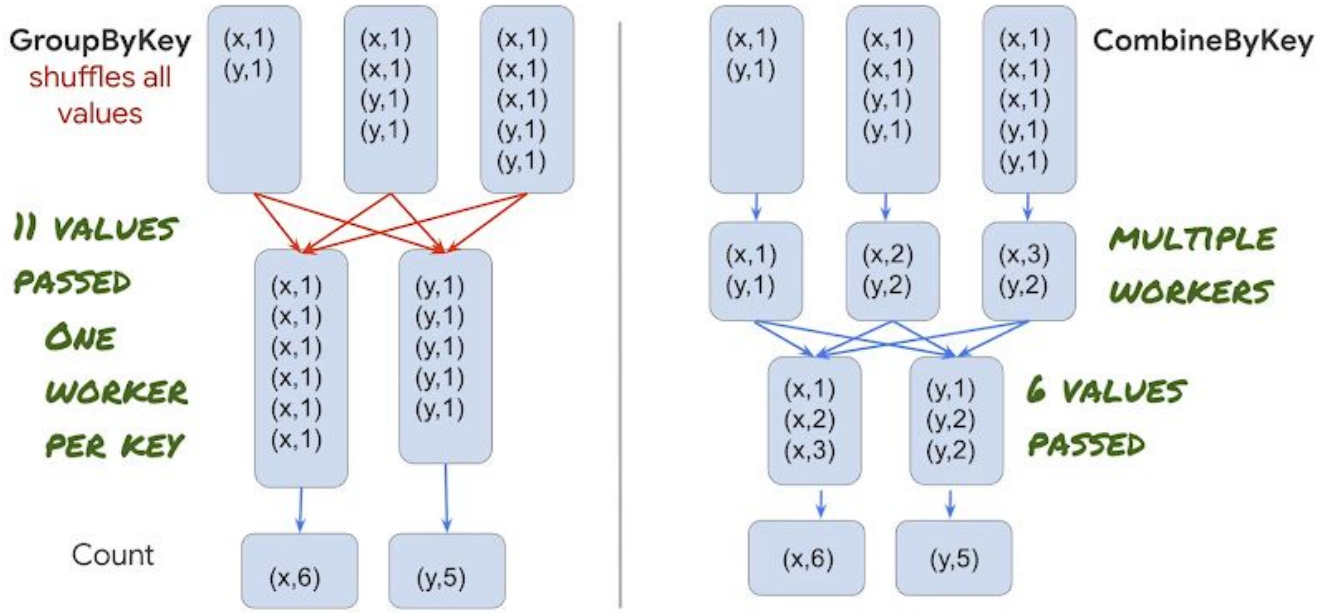


API JSON Name	Web UI*	Ratio Numerator **
`waitRatioAvg`	AVG	Time the average worker spent waiting to be scheduled.
`waitRatioMax`	MAX	Time the slowest worker spent waiting to be scheduled.
`readRatioAvg`	AVG	Time the average worker spent reading input data.
`readRatioMax`	MAX	Time the slowest worker spent reading input data.
`computeRatioAvg`	AVG	Time the average worker spent CPU-bound.
`computeRatioMax`	MAX	Time the slowest worker spent CPU-bound.
`writeRatioAvg`	AVG	Time the average worker spent writing output data.
`writeRatioMax`	MAX	Time the slowest worker spent writing output data.

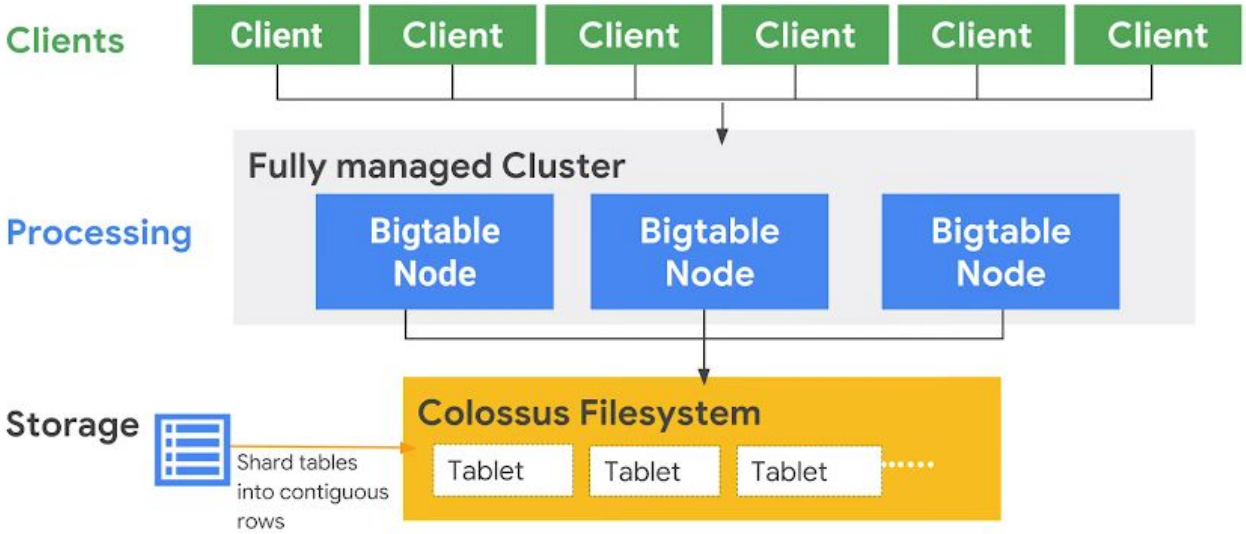
* The labels 'AVG' and 'MAX' are for illustration only and do not appear in the web UI.

** All of the ratios share a common denominator that represents the longest time spent by any worker in any segment.

- Dividing work
  - Table Partitioning
    - Time partitioned tables are a cost-effective way to manage data
    - Queries spanning time periods are straightforward to write
    - When you create tables with time-based partitions, BQ automatically loads data in the correct partition
      - -time_partitioning_type
      - -time_partitioning_expiration
  - Order of operations can influence shuffling overhead



        - Subsequent groups, aggregations, etc. are computed only within the time window
- BigTable performance
    - BigTable separates processing and storage



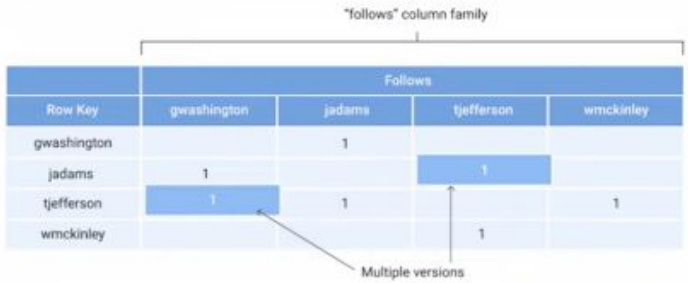
    - A table can only have one index (the row key)

	user_information				
Row Key	followers	birthdate	age	gender	messageCount
andrew	34	06_04_1986	34	F	1
brianna	31	07_24_1993	23	F	12
caitlyn	55	03_22_1952	51	F	15

ROWS ARE STORED IN ASCENDING ORDER OF THE ROW KEY

ENTRIES IN THE BIRTHDATE COLUMN CANNOT BE INDEXED

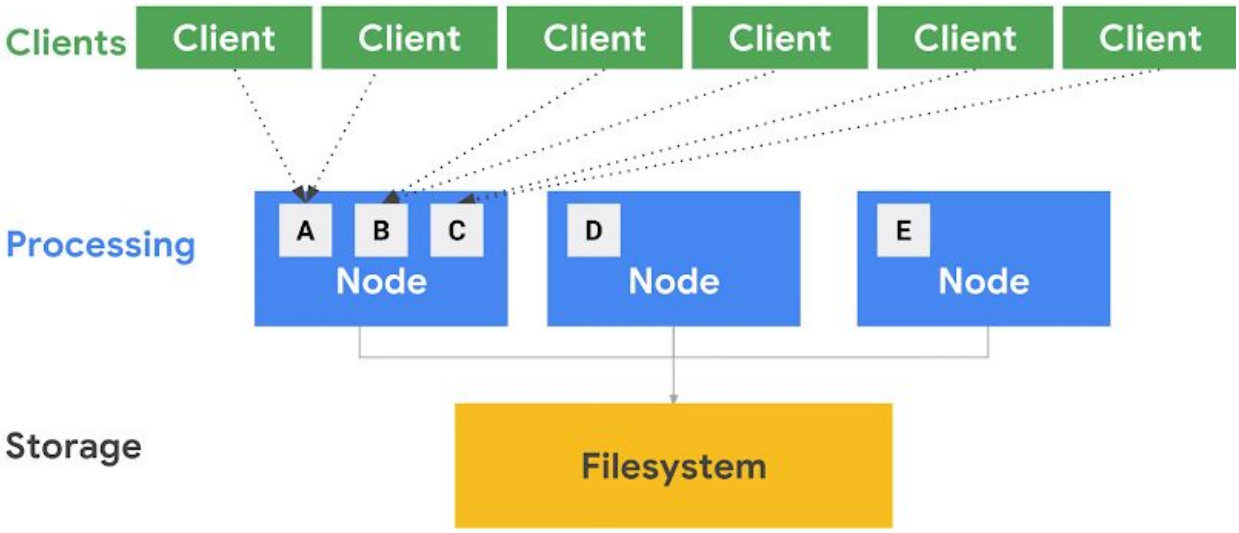
    - Wide Table (Cloud Spanner) versus Sparse Table (Cloud BigTable)

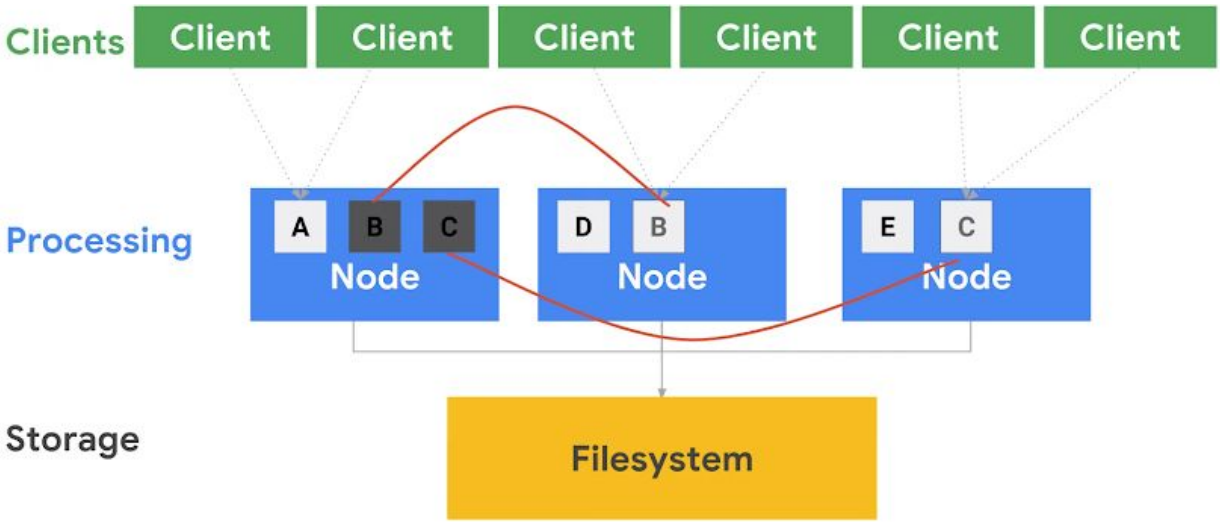


Sparse Table

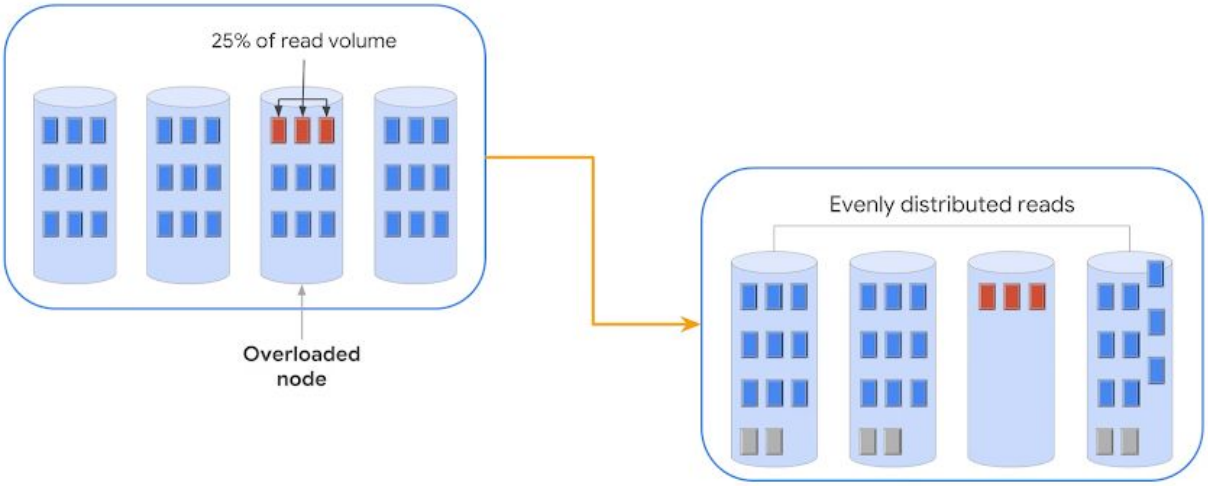
Row Key	Column Data
gwashington	Follows:jadams:1
jadams	Follows:gwashington:1, Follows:tjefferson:1
tjefferson	Follows:gwashington:1, Follows:jadams:1, Follows:wmckinley:1
wmckinley	Follows:tjefferson:1

- Row Key Design
  - What is the most common query you need to support?
  - Will you often need to retrieve records within a certain time range?
  - Can you have both distributed writes and block reads?
  - Choose an index that randomizes access instead of encouraging hotspots
- BigTable learns access patterns



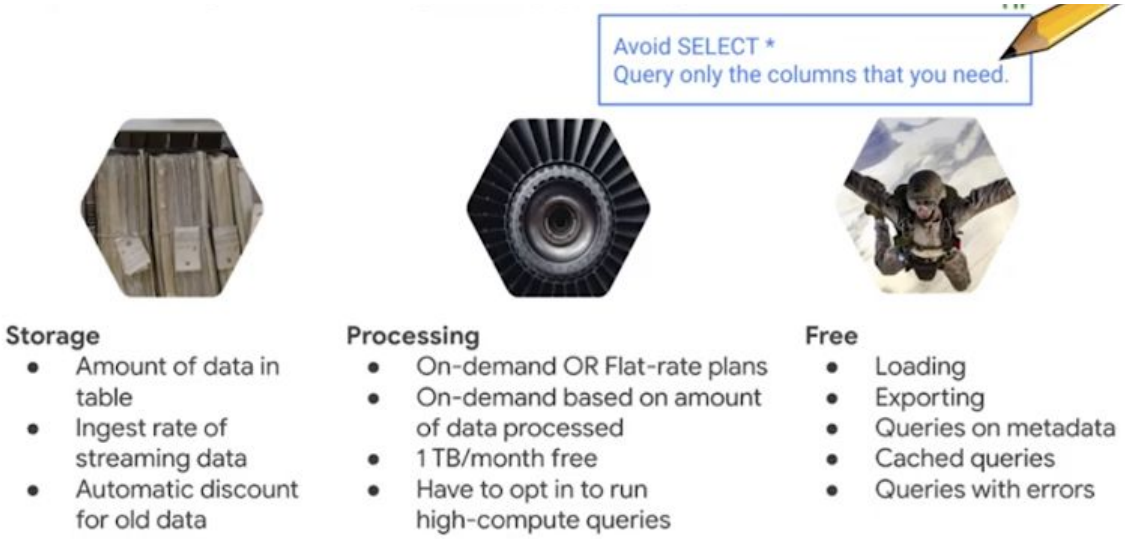


- Isolate performance from “noisy neighbors”
- React automatically to new patterns, splitting and balancing



- Growing a Bigtable cluster
  - Make sure clients & Bigtable are in the same zone
  - Change schema to minimize data skew
  - Performance increases linearly with number of nodes
  - SSD is faster than HDD
  - Takes a while after scaling up nodes for performance to be seen

- Price Estimation
  - 3 Categories of BQ Pricing
    - Storage
    - Processing
    - Free

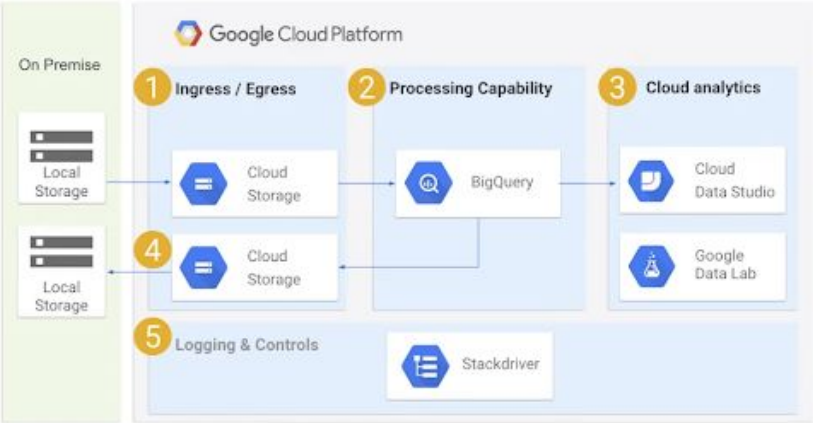


    - Use pricing calculator to estimate cost of a query
    - Use query validator to estimate how much data will be used by the query
    - Price your queries before running them
- Business Requirements and Optimization
    - Mapping business requirements to data representations
        - Working with business users
            - Listen first
        - Gathering business requirements
            - Watch for keywords that indicate requirements or part of solutions
    - Optimizing data representations, data infrastructure performance, and cost

Topic	Tips
Resizing and scaling resources	Serverless services eliminate the need to resize. Otherwise, you will need one or more metrics and a scaling process.
Data cleansing, distributed systems	Filter/clean/transform data first, and simpler algorithms later. Distributed systems, which can require re-balancing of data, are usually automatic in serverless services.
High performance algorithms	Consider alternative approaches. Dividing work up differently can enable higher performance algorithms. Order of operations can make a huge difference in performance.
Common sources of error (e.g., removing selection bias)	Data ingest, data selection. Sample data, keep track or distribution of data, and adjust if circumstances change.

- Case Study 3
    - Port the data from Hadoop to GCS

1. Source data is ingested in GCS (no changes to sourcing)
2. financial reports are generated entirely in BigQuery
3. Cloud Analytics allow users to review the reports
4. Data is egressed on-premise (no changes to target structures -> no changes downstream)
5. Logs and controls out of the box in Stackdriver

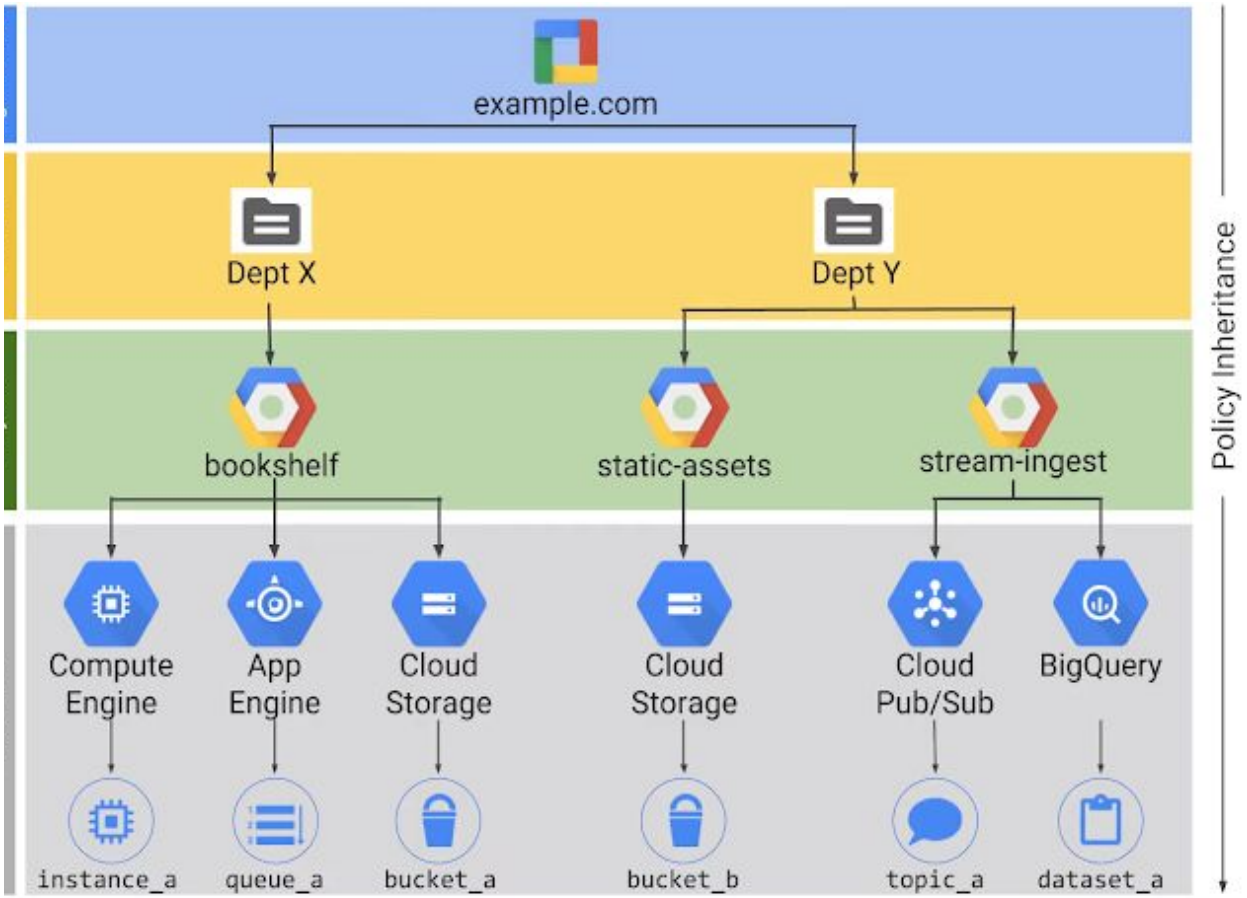


- Operationalizing ML Models Quiz
  - ML steps include:
    - Collect data, organize data, and create a model
  - What distinguishes problems that are appropriate for ML?
    - Hard problems that reduce to complex counting

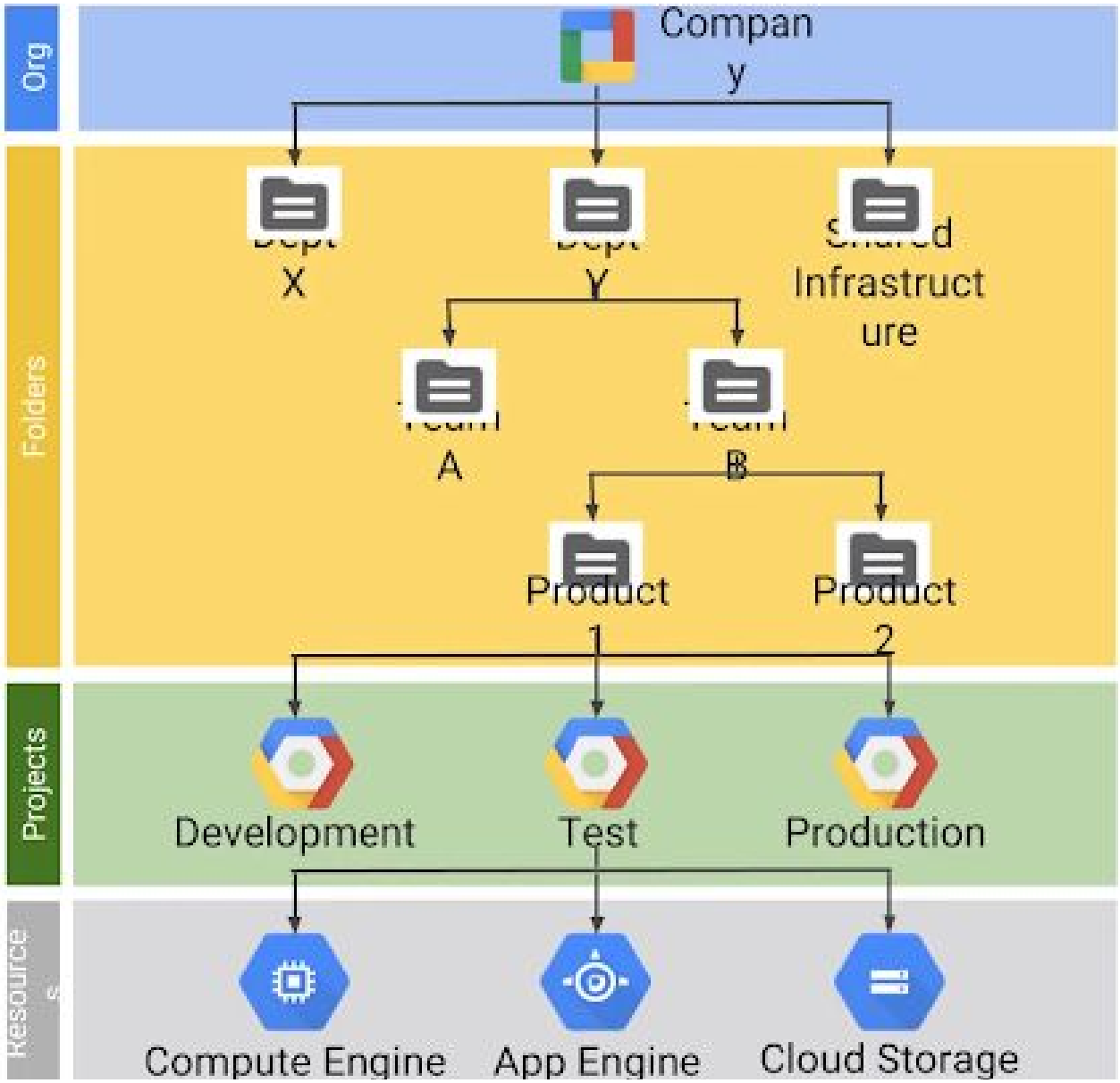
Part 4: Reliability, Policy, and Security to ensure solution quality

- Reliability, Policy, and Security
  - Reliability
    - Service produces consistent outputs and operates as expected
    - Availability
      - Service is accessible on demand. A measure of the amount of time that the item is in an operable state
    - Durability
      - Pertains to data loss
      - Means that data does not disappear and information is not lost overtime
      - Measure of the rate at which data is lost
    - Alternate service, failover, back up, disaster recovery
    - What are the business requirements and how much time is needed for acceptable disaster recovery?
    - Distributing for scale may improve reliability
      - Reduces the impact of a single loss, increasing reliability
    - Performing Quality Control
      - Monitor BigQuery with Stackdriver
      - Use Tensorboard to monitor training
        - Tailored to tensorflow
    - Estimator comes with a method that handles distributed training and evaluation

            - In Tensorflow, data is often divided into training and evaluation sets, defining apath for measuring effectiveness or improvement
        - Exam tip: check for built in quality control mechanisms in certain services as demonstrated by TensorBoard/TensorFlow
- Practice Exam Questions 4
    - Use Data Studio to visualize YouTube titles and aggregated view counts summarized over 30 days and segmented by Country Code in the fewest steps
        - Set up a YouTube data source for your channel for Data Studio. Set Views as the metric and set Video Title and Country Code as report dimensions
- Data Visualization and Reporting Tools
    - DataStudio lets you build dashboards and reports
        - Easy to read, share, and fully customizable
        - Handles authentication, access rights, and structuring of data
    - Troubleshooting and improving data quality and processing performance is distributed through all the technologies
    - Look for these tools in each technology area
    - Advocating policies and publishing data and reports
        - Not currently covered in Google Cloud training
- Designing for Security and Compliance
    - Cloud IAM Resource Hierarchy
        - Policy is set on a resource and each policy contains a set of roles and role members
        - Resources inherit policies from parents
        - Project policy overrules resource policy if the project is less restrictive



- Folders
  - Additional grouping mechanism and isolation between projects
    - Different legal entities
    - Departments
    - Teams
  - Folders allow delegation of administration rights



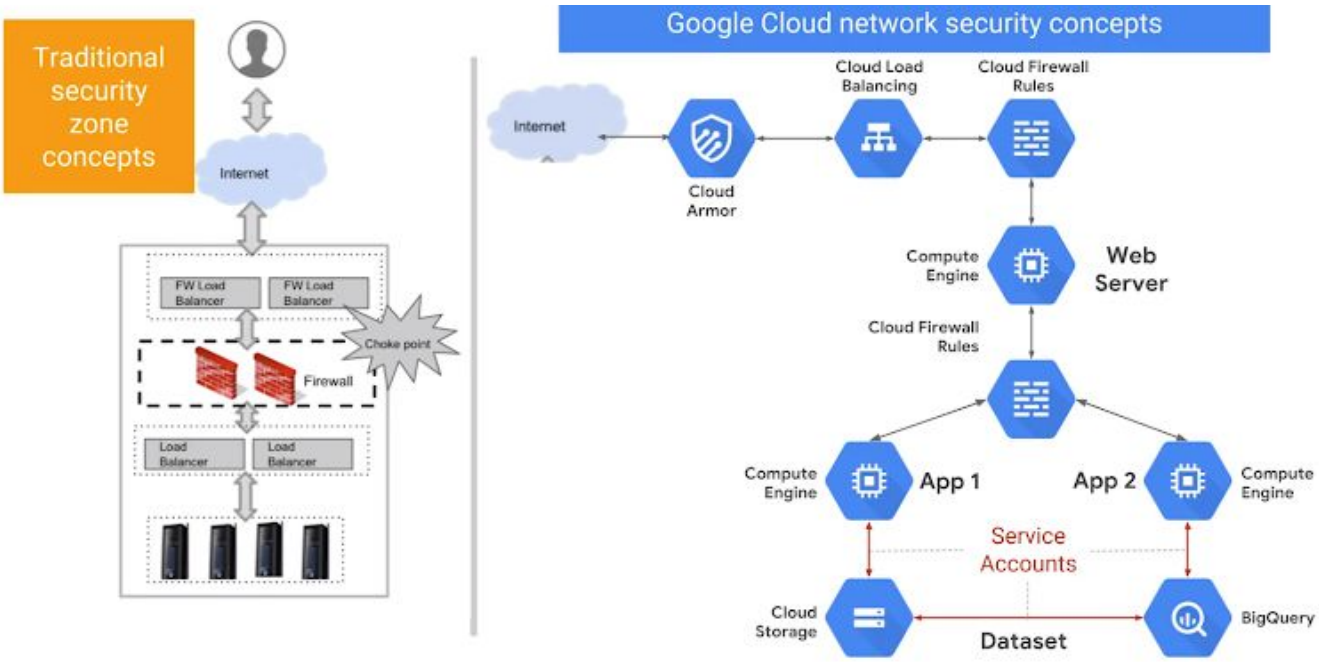
- Identity and Access
  - Separate responsibilities
  - Always have a backup or alternative in case the responsible person is unreachable
  - Have a separate maintenance path when the normal paths aren't working
  - Use groups to allocation permissions, then separately manage group membership
  - Customize roles for greater granularity of permissions
  - Give each group only the permissions they need to perform that job or task
  - Place critical functions on service machines to create accountability trail
  - Backup/spare logs and records; have a review, analysis, and monitoring strategy

- Encryption options

Encryption of VM disks and Cloud Storage buckets

Default Encryption	Customer Managed Encryption Keys (CMEK)	Customer-Supplied Encryption Keys (CSEK)	Client-Side Encryption
Data is automatically encrypted prior to being written to disk Each encryption key is itself encrypted with a set of master keys	Google-generated **data** encryption key (DEK) still used. Allows you to create, use, and revoke the **key** encryption key (KEK) Uses Cloud Key Management Service (Cloud KMS)	Keep keys on premises, and use them to encrypt your cloud services Google can't recover them Disk encryption on VMs Cloud Storage encryption Keys are never stored on disk unencrypted. You provide your key at each operation, and Google purges it from its servers when each operation completes.	Data is encrypted before it is sent to the cloud Your keys Your tools Google has no idea if your data is encrypted before it is uploaded No way to recover keys If you lose your keys, remember to delete the objects!

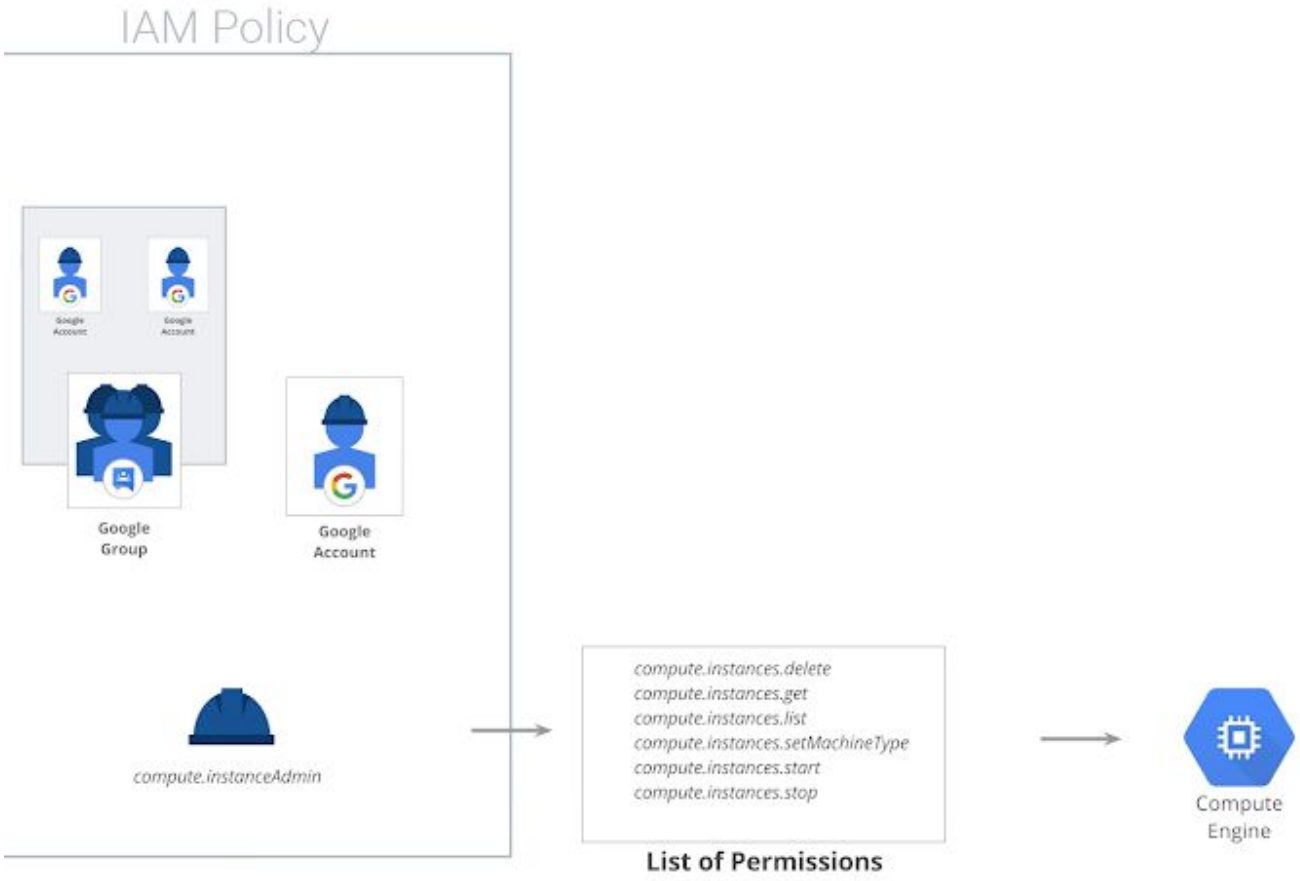
- Map from traditional security to cloud security concepts



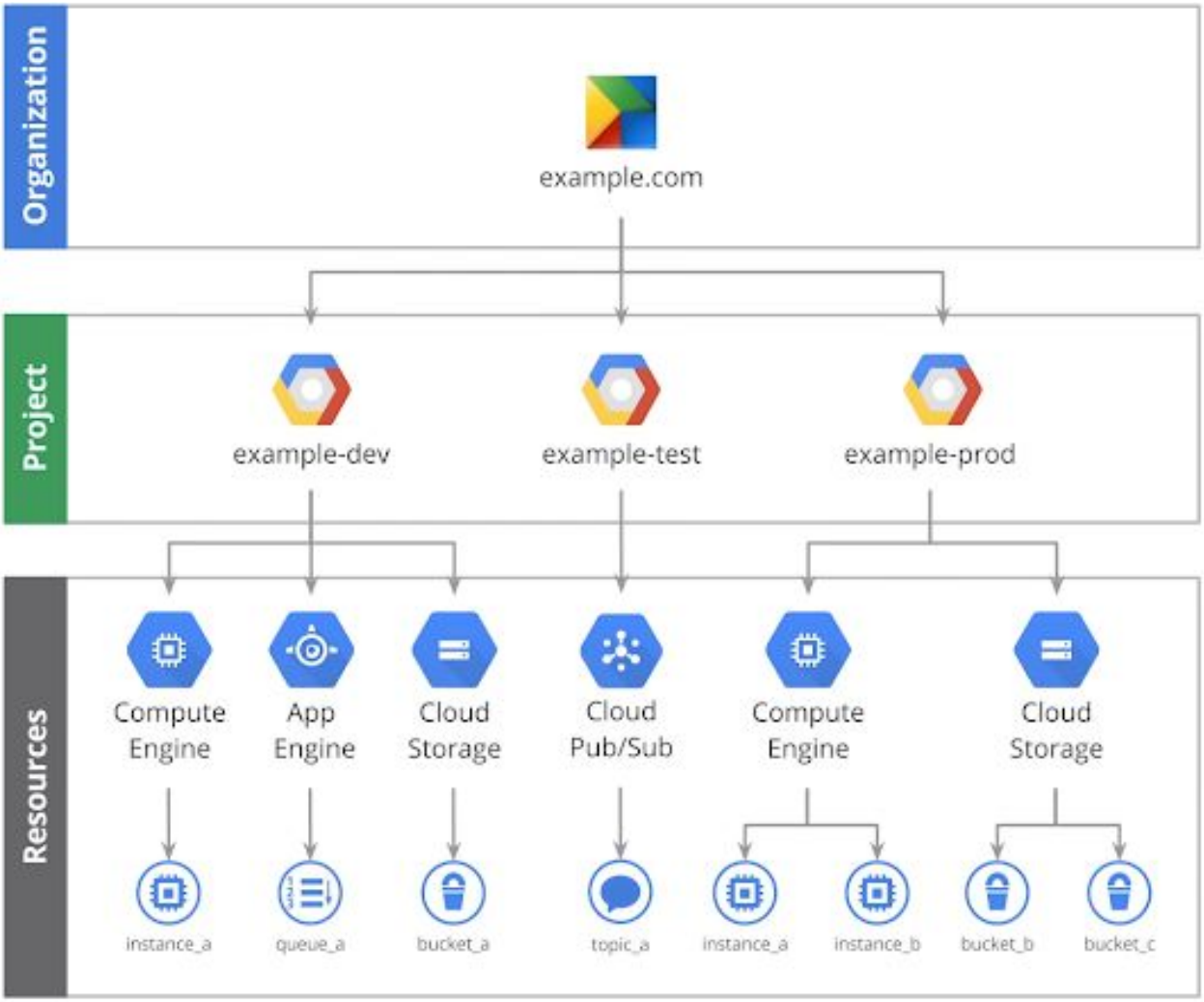
- Ensuring Solution Quality
  - IAM: understand permissions and custom roles. Under what conditions are custom roles preferred over standard predefined roles?
  - Data security, data loss prevention, cloud DLP allows you to minimize what you collect, store, expose, or copy. Classify or automatically redact sensitive data from text streams before you write to disk, generate logs, or perform analysis
  - Be familiar with these:

- Cloud IAM
    - Unified system for managing access to resources
    - Policies answer the question who can do what to which thing

    - Made of permissions
    - Identity defines WHO
        - Google Account
        - Service Account
        - Google Groups
        - Google Apps Domain
    - Resources are the various components of GCP which you can control the access
    - Permission
        - Service.Resource.Verb
            - Storage.bucket.create
            - Pubsub.topics.list
    - Role
        - Primitive roles:
            - Owner
            - Editor
            - Viewer
        - Curated roles are tied to resources


IAM Policy
Google Account
Google Account
Google Group
Google Account
compute.instanceAdmin
compute.instances.delete
compute.instances.get
compute.instances.list
compute.instances.setMachineType
compute.instances.start
compute.instances.stop
List of Permissions
Compute Engine

- Hierarchy



- Encryption, Key Management
- Data Loss Prevention API
    - Provides access to a powerful sensitive data inspection, classification, and de-identification platform
    - Over 120 built in information type detectors
    - Ability to define custom info type detectors using dictionaries, regular expressions, and contextual elements
    - Ability to detect sensitive data within streams of data, structured text, files in storage repositories such as cloud storage and bigquery, and even within images
    - Analysis of structured data to help understand its risk of being re-identitfied
- HIPAA, COPPA, FedRAMP, GDPR
    - GDPR
        - General Data Protection Regulation
            - Effective May 25, 2018. Enacted by EU

            - Clear set of guidelines for personal data protection
        - Consent
        - Data minimization
            - Organizations can only collect data if it relates to a well-defined business objective
        - Individual rights
    - HIPPA
        - Health insurance portability and accountability act of 1996
    - COPPA
        - Children's Online Privacy Protection Rule
    - FedRAMP
        - Federal Risk and Authorization Managemetn Program
  - A lot of resource administration is presented in the GCP console, but a lot of runtime information such as logs and performance is presented and reported in Stackdriver. Stackdriver provides information for troubleshooting both functional and performance issues
  - Establishing data quality at ingress by using Cloud Dataprep or by running an ETL pipeline can prevent many problems later in processing that would be difficult to troubleshoot
  - Remember the business purpose of the data processing. How resilient does the application need to be? For example, financial transactions usually cannot be dropped and must not be duplicated, but a statistical analysis might be equally valid if a small amount of data is lost. These assumptions influence the approach to rerunning failed jobs
    - Study these:
        - Cloud Dataprep
        - Fault tolerance
        - Rerunning failed jobs
        - Performing retrospective re-analysis
  - Where is the official authoritative data (sometimes called the source of truth) and where are the replicas? How frequently does data need to bshared or updated? Can smaller parts of the data be synchronized to reduce costs?
  - Where is the data stored? Where is the data going to be processed? Can data storage and data processing be in locations near each other?
  - When will the data need to be exported>? How difficult and expensive will it be to export data? For example, you might want to store data in a different location in a different type of storage to meet business requirements for portability
    - See: multi cloud data residency requirements
- Practice Exam Questions 5
  - Groups Analyst1 and Analyst2 should not have access to each other's BigQuery data
    - Place the data in separate datasets and assign appropriate group access
- Provide Analyst3 secure access to BigQuery query results, but not the underlying tables or datasets

    - Create a BigQuery authorized view and assign a project-level user role to Analyst3
- Secure data infrastructure and legal compliance

Topic	Tips
Identity and Access Management (IAM)	Use groups.
Data security	Encryption, location, access control. Principle of least privilege.
Penetration testing	Shape the scope and the objectives of the test.
Separation of Duties (SoD) security control	Isolate key roles. Always have an alternate/secondary who can take over if needed. Service accounts.

TIP
You can declare regional endpoints in Cloud Dataflow for security and network performance

TIP
Can you apply the "principle of least privilege" in case examples?

    - Bastion host
    - Designing for legal compliance

Topic	Tips
Legislation (e.g., Health Insurance Portability and Accountability Act (HIPAA), Children's Online Privacy Protection Act (COPPA), etc.)	PII - personally identifiable information Firewall, IAM, and Keys. CSEK? Do your policies conform to audit requirements? Make sure that growth doesn't break compliance
Audits	How are you monitoring security and compliance? What logs and reports are available? What responses will you take? Are they documented?

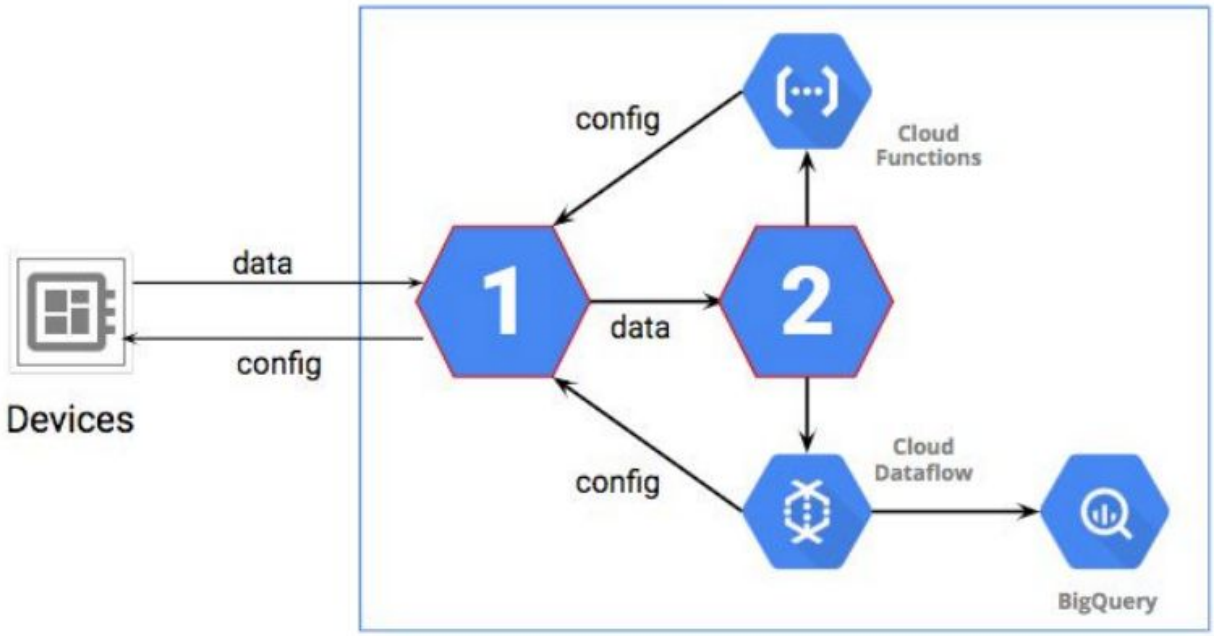
TIP
Infrastructure services can add protection to the data solution.

    - Know the default behavior of GCP so you don't duplicate it unnecessarily
- Preparing for Reliability, Policy, and Security Quiz
    - **Attempt 1: 18/25**
    - Which one is recommended best practice for Identity and Access Management?
        - Assign roles to individual users for better audit reporting
        - Assign roles to groups, then administer group membership
    - What point was made about monitoring and displaying parts of a solution?

        - That some services provide their own display and dashboard features, such as TensorBoard for TensorFlow, Stackdriver, and others
        - That google Data Studio contains all dashboards and displays you will need in all cases
    - What is one difference between Failover and Disaster Recovery?
        - Failover has very short downtime. Disaster Recovery tolerates may incur delays before service is restored
        - Failover is better, faster, and less expensive than disaster recovery
    - What is a coincidental benefit of distributing work as a scaling strategy?
        - The distribution algorithm acts like a firewall, increasing security
        - If a single unit goes out of service, it is a smaller portion of the overall service, so it increase reliability
- Practice Exam Quiz
    - Storage of JSON files with occasionally changing schema, for ANSI SQL queri
        - Store in BigQuery. Select "Automatically detect" in the Schema section.
    - Low-cost one-way one-time migration of two 100-TB file servers to GCP; data will only be accessed from Germany.
        - Use Transfer Appliance. Transfer to a Cloud Storage Regional storage bucket.
    - Cost-effective backup to GCP of multi-TB databases from another cloud including monthly DR drills.
        - Use Storage Transfer Service. Transfer to Cloud Storage Nearline bucket.
    - 250,000 devices produce a JSON device status every 10 seconds. How do you capture event data for outlier time series analysis?
        - Capture data in Cloud Bigtable. Use the Cloud Bigtable cbt tool to display device outlier data
    - Event data in CSV format to be queried for individual values over time windows. Which storage and schema to minimize query costs?
        - Use Cloud Bigtable. Design tall and narrow tables, and use a new row for each single event version.
    - Customer wants to maintain investment in existing Apache Spark code data pipeline.
        - Cloud Dataproc
    - Host a deep neural network machine learning model on GCP. Run and monitor jobs that could occasionally fail.
        - Use Cloud Machine Learning Engine to host your model. Monitor the status of the Jobs object for 'failed' job states.
    - Cost-effective way to run non-critical Apache Spark jobs on Cloud Dataproc?
        - Set up a cluster in standard mode with high-memory machine types. Add 10 additional preemptible worker nodes.
    - Promote a Cloud Bigtable solution with a lot of data from development to production and optimize for performance.

        - Change your Cloud Bigtable instance type from Development to Production, and set the number of nodes to at least 3. Verify that the storage type is SSD.
    - As part of your backup plan, you want to be able to restore snapshots of Compute Engine instances using the fewest steps.
        - Use the snapshots to create replacement disks. Use the disks to create instances as needed.
    - You want to minimize costs to run Google Data Studio reports on BigQuery queries by using prefetch caching.
        - Set up the report to use the Owner's credentials to access the underlying data in BigQuery, and verify that the 'Enable cache' checkbox is selected for the report.
    - A Data Analyst is concerned that a BigQuery query could be too expensive.
        - Use the SELECT clause to limit the amount of data in the query. Partition data by date so the query can be more focused.
    - BigQuery data is stored in external CSV files in Cloud Storage; as the data has increased, the query performance has dropped.
        - Import the data into BigQuery for better performance
    - Source data is streamed in bursts and must be transformed before use.
        - Use Cloud Pub/Sub to buffer the data, and then use Cloud Dataflow for ETL.
    - Calculate a running average on streaming data that can arrive late and out of order.
        - Use Cloud Pub/Sub and Cloud Dataflow with Sliding Time Windows
    - Testing a Machine Learning model with validation data returns 100% correct answers
        - The model is overfit. There is a problem.
    - A client is using Cloud SQL database to serve infrequently changing lookup tables that host data used by applications. The applications will not modify the tables. As they expand into other geographic regions they want to ensure good performance. What do you recommend?
        - Read replicas
    - A client wants to store files from one location and retrieve them from another location. Security requirements are that no one should be able to access the contents of the file while it is hosted in the cloud. What is the best option?
        - Client-side encryption
    - Three Google Cloud services commonly used together in data engineering solutions. (Described in this course).
        - Cloud Pub/Sub, Cloud Dataflow, BigQuery
    - What is AVRO used for?
        - Serialization and de-serialization of data so that it can be transmitted and stored while maintaining an object structure.

- A company has a new IoT pipeline. Which services will make this design work? Select the services that should be used to replace the icons with the number "1" and number "2" in the diagram.



  - Cloud IoT Core, Cloud Pub/Sub
- A company wants to connect cloud applications to an Oracle database in its data center. Requirements are a maximum of 9 Gbps of data and a Service Level Agreement (SLA) of 99%.
  - Partner Interconnect
- A client has been developing a pipeline based on PCollections using local programming techniques and is ready to scale up to production. What should they do?
  - They should use the Cloud Dataflow Cloud Runner.
- A company has migrated their Hadoop cluster to the cloud and is now using Cloud Dataproc with the same settings and same methods as in the data center. What would you advise them to do to make better use of the cloud environment?
  - Store persistent data off-cluster. Start a cluster for one kind of work then shut it down when it is not processing data.
- An application has the following data requirements. 1. It requires strongly consistent transactions. 2. Total data will be less than 500 GB. 3. The data does not need to be streaming or real time. Which data technology would fit these requirements?
  - Cloud SQL