Google Certified Professional Data Engineer Master Cheat Sheet

Overview

This test assesses your ability to:

1. Design data processing systems
2. Build and Operationalize Data Processing Systems
3. Operationalize Machine Learning Models
4. Ensure Solution Quality

Those categories are explained in more detail here.

At a testing facility you will have 2 hours to solve 50 multiple choice questions on a computer. You can mark questions you are unsure about and can review them towards the end of the test. After you submit your test, you will see a provisional result of either PASS or FAIL. The official results will be sent to you in the next few days. If everything goes well you'll receive this certification. Good luck!

Ingestion

Transfer Appliance

- Transfer large amounts of data quickly and cost-effectively to GCP.
- Transfers directly to GCS or BQ
- Data Size >= 20TB
- Offline Data Transfer
- Takes more than 1 week to upload data.

Workflow

- Receive Transfer Appliance and configure it and connect it to your network.
- Before data is stored, it is deduplicated, compressed and encrypted with AES 256 algorithm using a password and passphrase specified by user.
- Data integrity check is performed.
- Transfer Appliance is shipped back to Google.
- Encrypted data is copied to GCS staging bucket.

    - Still compressed, deduplicated, and encrypted.
- Email will be sent to user notifying the rehydration process can start.
- Transfer Appliance Rehydrator application is run specifying the GCS destination bucket.
    - This application is run on GCE.
    - Compared CRC32C hash value of each file being rehydrated.
    - If checksums don't match, file is skipped and appears in skip file list with Data corruption detected.
- Data integrity check performed again.
- Appliance securely wiped and re-imaged.

Use Cases

- Data Collection
    - Geographical, environmental, medical, or financial data for analysis.
    - Need to transfer data from researchers, vendors, or other sites to GCP.
- Data Replication
    - Supporting current operations with existing on prem infrastructure but experimenting with cloud.
    - Allows decommissioning duplicate datasets, test cloud infrastructure, and expose data to machine learning analysis.
- Data Migration
    - Offline data transfer is suited for moving large amounts of existing backup images and archives to ultra-low-cost, highly durable, and highly available archival storage (Nearline/Coldline).

NOTE - Should not be used for repeated data transfers.

Storage Transfer Service

- Transfers data from an online data source (Amazon S3, HTTP/HTTPS location, GCS bucket) to a data sink (always GCS bucket).
- Schedule one-time transfer operations or recurring ones
- Delete existing objects in the destination bucket if they don't have a corresponding object in source
- Delete source objects after transferring them
- Schedule periodic synchronization from data source to data sink with advanced filters based on file creation data, file-name filters, and the times of day you prefer to import data.

Use cases:

- Backup data to GCS from other storage providers
- Move data from one GCS bucket to another (enables availability to different groups of users or applications)
- Periodically move data as part of a processing pipeline or analytical workflow

Transfer Service vs. Gsutil

- On premise data source : gsutil
- Another cloud storage provider data source : Transfer Service

Storage

Google Cloud Storage

- Blob storage. Content not indexed.
- Virtually unlimited storage.

- Can have domain name buckets
- Can make requesters pay (ex. requester in different project)
- PubSub can have notifications based on operations to buckets/objects
- Objects are immutable
- Can set Cache-Control metadata for frequently accessed objects
- Keep in mind compliance requirements when storing data in certain regions.
- No native directory support
    - Forward slashes have no special meaning
    - Performance of a native filesystem is not present.
- Storage classes can change, but the objects (files) within them retain their storage class.
- Not ideal for high volume read/write
- A way to store data that can be commonly used by Dataproc and Bigquery
- Signed Urls
    - Provided limited permission and time to make a request.
    - Contain auth information in query string, allowing users without credentials to perform specific actions on a resource.
- Signed URL to give temporary access and users do not need to be GCP users
    - TODO

Storage Classes

Multi-Regional

- Serving website content, interactive workloads, mobile game/gaming applications
- Highest availability
- Geo-redundant: Stores data in at least 2 regions separated by at least 100 miles within the multi-regional location of the bucket.

Regional

- Storing data used by Compute Engine
- Better performance for data-intensive computation

Nearline

- Accessed once a month max
- 30 day min. storage duration
- Ex. Data backup, disaster recovery, archival storage

Coldline

- Accessed once a year max
- 90 day min. storage duration
- Ex. Data stored for legal or regulatory reasons

Versioning

- Needs to be enabled
- Things this enables:
    - List archived versions of an object
    - Restore live version of an object from an older state
    - Permanently delete an archived version
- Archived versions retain ACLs and does not necessarily have same permissions as live version of object.

IAM vs ACLs

- IAM
    - Apply to all objects within a bucket.
    - Standard Roles
        - Storage.objectCreator
        - Storage.objectViewer
        - Storage.objectAdmin
        - Storage.admin – full control over buckets
            - Can apply to a specific bucket.
    - Primitive Roles

- ACL (Access Control List)
    - Use if need to customize access to individual objects within a bucket.
    - Permissions – What actions can be performed?
    - Scopes – Which defines who can perform the actions (user or group)
    - Reader/Writer/Owner

Encryption

Encryption at rest (Google-Managed Encryption Keys)

- Default (AES-256)
- Use TLS or HTTPS to protect data as it travels over Internet

Server-side encryption

- Layers on top of default encryption
- Occurs after GCS receives data, but before written to disk

Customer-supplied encryption keys

- Provide key for each GCS operation
- Key purged from servers after operation is complete
- Stores only a cryptographic hash of key for future requests
- Transfer Service, Dataflow, and Dataproc do not support this currently
- Key rotation
    - Edit .boto config file
    - Encryption_key = [NEW_KEY]
    - Decryption_key1 = [OLD_KEY]
    - gsutil rewrite -k gs://[BUCKET]/[OBJECT]

Customer-managed encryption keys

- Generate and manage keys using Cloud Key Management Service (KMS)
- KMS can be independent from the project that contains buckets (separation of duties)
- Uses service accounts to encrypt/decrypt
- Cloud SQL exports to GCS and Dataflow do not support this currently

Client-side encryption

- Occurs before data sent to GCS
- GCS performs default encryption on it as well.

Cloud SQL

- Managed/No ops relational database (PostgreSQL, MySQL)
    - Complex queries perform better in postgresql
- Best for **gigabytes** of data with **transactional** nature
    - Low latency
    - Doesn't scale well beyond GB's
    - Data structures and underlying infrastructure required
- Too slow for analytics/BI/warehousing (OLAP)
- 2nd Generation Allow
    - Cloud Proxy Support
    - Higher availability configurations
    - Maintenance won't take down the server
- Use SSD for production (instead of hard disk (persistent disk))
- Enable binary logging
    - For Point-in-time recovery and replication
- Bulk Loading Data
    - Copy data to GCS from SQL dump file or CSV files
        - SQL dump files cannot contain triggers, views, stored procedures
    - Import it into DB using copy from csv or something similar.
    - Use correct flags for dump file.
    - Compress data to reduce costs
        - Cloud SQL can import compressed .gz files
    - Use InnoDB for Second Generation instances
- Limited to 10 TB and is regional (not global)
    - Use Spanner if not good enough

- Use Case:
    - Medical Records
    - Blogs
- Read Replicas
    - In same region as master
    - Purpose to offload requests for analytics traffic from master.
- What to do when data size limits performance?
    - Many smaller tables perform better than one larger.

IAM

- Cloudsql.admin
- Cloudsql.editor
    - Can't see or modify permissions, users, or ssl certs.
    - No ability to import data or restore from backup, nor clone, delete, or promote instances.
    - No ability to delete databases, replicas, or backups.
- Cloudsql.viewer
    - Read only access to all Cloud SQL instances.
- Cloudsql.client
    - Connectivity access to Cloud SQL instances from App Engine and the Cloud SQL Proxy.
    - Not required for accessing an instance using IP addresses.

Cloud Spanner

- Distributed and scalable solution for RDBMS (more expensive)
- Horizontal scaling: Add more machines
- Use when:
    - Need high availability
    - Strong consistency
    - Transactional support for reads and writes (especially writes)
- Don't use when:
    - Data is not relational, or not even structured
    - Want an open source RDBMS

- Strong consistency/availability is overkill

Data Model

- Specifies a parent-child relationship for efficient storage
- Interleaved representation (like HBase)

Parent Child Relationship

- Between tables
- Cause physical location for fast access
  - i.e. query Students and Grades together, make Grades child of Student
- Primary key of parent table must to be part of the key in the interleaved child table.

Interleaving

- Rows are stored in sorted order of primary key values
- Child rows are inserted between parent rows with that key prefix

Hotspotting

- Need to choose primary keys carefully (like HBase)
- Do not use monotonically increasing values, else writes will be on the same locations.
  - No timestamps (also sequential)
    - Use descending order if timestamps are required.
- Use hash of key value if using naturally monotonically ordered keys (serial in postgres)

Splits

- Parent-child relationship can get complicated (i.e. 7 layers deep)
- Spanner is distributed – uses "splits"
- Split – Range of rows that can be moved around independent of other rows
- Added to distribute high read-write data (to break up hotspots)

Secondary Indices

- Key-based storage ensures fast sequential scan of keys (like HBase)
- Can also add secondary indices (unlike HBase)

    - Can cause data to be stored twice
        - i.e. Grades -> Course table | Grades -> Students table

- Fine grained control on use of indices
    - Force query to use specific index: Index Directives
    - Force column to be copied into secondary index (use a STORING clause)

- Data Types
    - Non-normalized types such as ARRAY and STRUCT available too.
        - STRUCTs: NOT OK in tables, but can be returned in queries
        - ARRAYs: OK in tables, but ARRAYs of ARRAYs are not

Transactions

- Supports serializability
    - All transactions appear if they were executed in a serial order, even if some operations of distinct transactions actually occurred in parallel.
- Stronger than traditional ACID
    - Transactions commit in an order that is reflected in their commit timestamps
    - Commit timestamps are “real time”
- 2 Transaction Modes
    - Locking read-write
        - Slow
        - Only one that supports writing data
    - Read-only
        - Fast
        - Only requires read locking
- If making a one-off read use “Single Read Call”
    - Fastest, no transaction checks needed!

Staleness

- Can set timestamp bounds
    - Strong: Read latest data
    - Bounded Staleness: Read version no later than …
        - Could be in past or future

Multitenancy

- Classic way is to create a separate database for each customer.
- Recommended way for Spanner: Include a CustomerId key column in tables.

Replicas

- Paxos-based replication scheme in which voting replicas take a vote on every write request before it is committed.
- Writes
    - Client write requests always go to leader replica first, even if a non-leader is closer geographically.
    - Leader logs incoming write, forwards it in parallel to other replicas that are eligible to vote.
    - Replicas complete its write and then responds back to leader with a vote on whether the write should be committed.
    - Write is committed when a quorum agrees.
- Reads
    - Reads that are part of a read-write transaction are served from the leader replica, since the leader maintains the locks required to enforce serializability.
    - Single read and reads in a read-only transaction might require communication with leader, depending on concurrency mode.
- Single-region instances can only use read-write replicas. (3 in prod)
- Types
    - Read-write
        - Maintain a full copy of your data.
        - Can vote, can become leader, can serve reads
    - Read-only
        - Maintain a full copy of your data, which is replicated from read-write replicas.
        - Can serve reads
        - Do not participate in voting to commit writes -> location of read-only replicas never contribute to write latency.
        - Allow scaling of read capacity without increasing quorum size needed for writes (reduces total time of network latency for writes)
    - Witness
        - Can vote
        - Easier to achieve quorums for writes without the storage and compute resources required by read-write replicas to store a full copy of data and serve reads.

Production Environment

- At least 3 nodes
- Best performance when each CPU is under 75% utilization

Architecture

- Nodes handle computation for queries, similar to that of BigTable.
    - Each node serves up to 2 TB of storage.
    - More nodes = more CPU/RAM = increased throughput.
- Storage is replicated across zones (and regions, where applicable).
    - Like BigTable, storage is separate from computing nodes.
- Whenever an update is made to a database in one zone/region, it is automatically replicated across zones/regions.
    - Automatic synchronous replications.
        - When data is written, you know it has been written.
        - Any reads guarantee data accuracy.

IAM

- Project, instance, or database level
- Roles/spanner._____
    - Admin – Full access to Spanner resources
    - Database Admin – Create/edit/delete databases, grant access to databases
    - Database Reader – Read databases and execute SQL queries and view schema.
    - Database User – Read and write to DB, execute sql on DB including DML and Partitioned DML, view and update schema.
    - Viewer – View that instances and databases exist
        - Cannot modify or read from database.

Datastore

- Typically, not used for either OLTP or OLAP
    - Fast lookup on keys is the most common use case.
- Serverless

- Specialty is that query execution depends on the size of the returned result and not the size of the data set.
    - Best for lookup of non-sequential keys (needle in haystack)
    - Built on top of BigTable
        - Non-consistent for every row.
        - Document DB for non-relational data.
        - MongoDB equivalent (JSON-oriented NoSQL).
- Suitable for:
    - Atomic transactions
        - Can execute a set of operations where all succeed, or none occur.
    - ACID transactions, SQL-like queries.
    - Structured data.
    - Hierarchical document storage such as HTML and XML
- Query
    - Can search by keys or properties (if indexed)
    - Key lookups somewhat similar to Amazon DynamoDB
    - Allow for SQL-like querying down to property level.
    - Does not support:
        - Join operations
        - Inequality filtering on multiple properties.
            - Only 1 inequality filter per query is allowed.
    - Filtering on data based on a result of a subquery.
- Performance
    - Fast to Terabyte scale, low latency.
    - Quick read, slow write as it relies on indexing every property (default) and must update indexes as updates/writes occur.
- Comparison to RDBMS
    - All Datastore queries use indices.
    - Query time depends on size of result set alone in Datastore whereas RDBMS also depends on size of data set.
    - Entities (rows) of the same kind (table) can have different properties (fields).
    - Different entities can have properties with same name, but different value type.

- Properties can vary between entities.
    - Think optional tags in HTML.
- Avoid DataStore when:
    - You need very strong transaction support.
    - Non-hierarchical or unstructured data (use BigTable instead)
    - Need extreme scale (10M+ read/writes per second) - BigTable
    - Analytics/BI/Data warehousing (BQ instead)
    - If application has a lot of writes and updates on key columns.
    - You need near zero latency (use in memory db Redis)
- Use DataStore when:
    - Scaling of read performance – to virtually any size.
    - Use for hierarchical documents with KV data.
    - Apps that need highly available structured data at scale.
    - Product catalogs, real time inventory
    - User profiles – mobile apps
    - Game save states
- Single Datastore database per project
- Where can you host?
    - Multi-regional for wide access
    - Single region for lower latency for single location
    - Cannot change after assignment… (have to delete project)

Entity Groups

- Hierarchical relationship between entities.
- Ancestor Paths and Child Entities.

Index Types

- Built in – default option
    - Allows single property queries
- Composite – specified with index configuration file (index.yaml)

    - gcloud datastore create-indexes index.yaml
        - Creating/updating

Deleting Index

- datastore indexes cleanup
    - Deletes all indexes for the production Datastore mode instance that are not mentioned in the local version of index.yaml.

Exploding Indexes

- Default – create entry for every possible combination of property values
- Results in higher storage and degraded performance
- Solutions
    - Use custom index.yaml file to narrow index scope
    - Do not index properties that don't need indexing

Full Indexing

- Built in indices on each property (~field) of each entity kind (~table row).
- Composite indices on multiple property values.
- Can exclude properties from indexing if certain it will never be queried.
- Each query is evaluated using its "perfect index"

Perfect Index

- Given a query, which is the index that most optimally returns query results?
- Depends on following (in order)
    - Equality filter
    - Inequality filter
    - Sort conditions if any specified.

Implications of Full Indexing

- Updates are really slow.
- No joins possible.
- Can't filter results based on subquery results.

- Can’t include more than one inequality filter.

Multi-Tenancy

- Separate data partitions for each client organizations.
- Can use the same schema for all clients, but vary the values.
- Specified via a namespace (inside which kinds and entities can exist)

Transaction Support

- Can optionally use transactions – not required
- Stronger than BigQuery and BigTable

Consistency

- Strongly consistent
    - Return up to date result, however long it takes
    - Ancestor query
        - Those that execute against an entity group
        - Can set the read policy of a query to make this eventually consistent.
    - key-value operations
- Eventually consistent
    - Faster, but might return stale data
    - Global queries/projections

Deleting entities in bulk?

- Use Dataflow
    - Datastore delete template that can be used to delete entities selected by a GQL query.

Exporting Entities

- Deploy App Engine service that calls Datastore mode managed export feature.
- Can run this service on a schedule with an App Engine Cron Service.

Cloud Firestore

- Newest version of Datastore.
- Native Mode
    - New strongly consistent storage layer.
    - New data model:
        - Kind => Collection Group
        - Entity => Document
        - Property => Field
        - Key => Document ID
    - Real-time updates
    - Mobile and Web client libraries
        - Scales to millions of concurrent clients.
    - Datastore Mode
        - Removes previous consistency limitations of Datastore.
        - Strongly consistent queries across the entire database.
        - Transactions can access any number of entity groups.
        - Scales to millions of writes per second.

IAM Roles

- Datastore.owner with Appengine.appAdmin
    - Full access to Datastore mode.
- Datastore.owner without Appengine.appAdmin
    - Cannot enable Admin access
    - Cannot see if Datastore mode Admin is enabled
    - Cannot disable Datastore mode writes
    - Cannot see if Datastore mode writes are disabled.
- Datastore.user
    - Read/write access to data in Datastore mode database.
    - Intended for application developers and service accounts.
- Datastore.viewer
    - Read access to all Datastore mode resources.
- Datastore.importExportAdmin
    - Full access to manage import and exports.
- Datastore.indexAdmin
    - Full access to manage index definitions.

BigTable

- HBase equivalent
  - Work with it using HBase API
  - Advantages over HBase
    - Scalability (storage autoscales)
    - Low ops/admin burden
    - Cluster resizing without downtime
    - Many more column families before performance drops (~100K)
- Stored on Google's internal store Colossus
- Not transactional (can handle petabytes of data)
- Fast scanning of sequential key values
- Column oriented NoSQL database
  - Good for sparse data
- Sensitive to hot spotting (like Spanner)
  - Data is sorted on key value and then sequential lexicographically similar values are stored next to each other.
  - Need to design key structure carefully.
- Designed for Sparse Tables
  - Traditional RDBMS issues with sparse data
    - Can't ignore with petabytes of data.
    - Null cells still occupy space.
- Use BigTable When:
  - Very fast scanning and high throughput
    - Throughput has linear growth with node count if correctly balanced.
  - Non-structured key/value data
  - Each data item is < 10MB and total data > 1TB
  - Writes are infrequent/unimportant (no ACID) but fast scans crucial
  - Time Series data
- Avoid BigTable When:
  - Need transaction support
  - Less than 1TB data (can't parallelize)
  - Analytics/BI/data warehousing
  - Documents or highly structured hierarchies
  - Immutable blobs > 10MB each

4-Dimensional Data Model

- Row-Key
  - Uniquely identifies a row
  - Can be primitives, structures, arrays
  - Represents internally as a byte array
  - Sorted in ascending order
  - **NOTE** - Can only query against this key.
- Column Family
  - Table name in RDBMS
  - All rows have the same set of column families
  - Each column family is stored in a separate data file
  - Set up at schema definition time
    - Columns can be added on the fly
  - Can have different columns for each row
- Column
  - Columns are units within a column family.
- Timestamp
  - Support for different versions based on timestamps of same data item. (like Spanner)
  - Omit timestamp gets you the latest data.

Hotspotting

- Overloading a node with requests.
- Row keys to Use
  - Field Promotion
    - Move fields from column data that you need to search against should be included in a single row key.
    - Use in reverse URL order like Java package names
      - Keys have similar prefixes, but different endings
  - Salting
    - Hash the key value
  - Timestamps as suffix in key (reverse timestamp)
- Row Keys to Avoid
  - Domain names (as opposed to field promotion)
    - Will cause common portion to be at end of row key leading to adjacent values to not be logically related.
  - Sequential numeric values.
  - Timestamps alone
  - Timestamps as prefix of row-key.

- Mutable or repeatedly updated values.

Schema Design

- Each table has just 1 index – row key
- Rows sorted lexicographically by row key
- All operations are atomic at row level
- Keep all entity info in a single row.
- Related entities should be in adjacent rows
  - More efficient reads.
- Tables are sparse: Empty columns don't take up any space.
  - Create a very large number of columns even if most are empty in most rows.

"Warming the Cache"

- BigTable will improve performance over time.
- Will observe read and write patterns and redistribute data so that shards are evenly hit.
- Will try to store roughly same amount of data in different nodes.
- Testing over hours is important to get true sense of performance.

SSD or HDD Disks

- Use SSD unless skimping costs.
  - Can be 20x faster on individual row reads.
    - Less important with batch reads or sequential scans.
  - More predictable throughput too (no disk seek variance)
- When to use HDD?
  - If > 10 TB storage
  - All batch queries
  - Not using the data to back a user-facing or latency-sensitive application
  - Data archival, where you write very large amounts of data and rarely read that data.
- The more random access, the stronger case for SSD
  - Purely random -> maybe use DataStore
- Impossible to switch between SSD and HDD
  - Export data from the existing instance and import data into a new instance.
  - OR write a cloud Dataflow or Hadoop MapReduce job that copies the data from one instance to another.

Poor Performance Explained

- Poor schema design
- Inappropriate workloads
    - Too small (< 300 GB)
    - Used in short bursts
- Cluster too small
- Cluster just fired up or scaled up
- HDD instead of SSD
- Dev. Vs. Prod instance

Replication on Performance (new cluster)

- Reads and Scans
    - Replication can improve throughput especially with multi-cluster routing.
    - Can reduce read latency by placing BigTable data geographically closer to application's users.
- Write throughput
    - Replication does not increase write throughput. May actually go down since replication requires each additional cluster to do more work.
    - Adding more nodes intra cluster will increase write throughput.

Key Visualizer

- Provides daily scans that show usage patterns for each table in a cluster.
- Makes it easy to check whether your usage patterns are causing undesirable results, such as hotspots on specific row keys or excessive CPU utilization.

Data Update

- Deleting/updating actually write a new row with the desired data.
- Append only, cannot update a single field
- Tables should be tall and narrow
    - Tall – Store changes by appending new rows
    - Narrow – Collapse flags into a single column

Production & Development

- Prod:
  - Standard instance with 1-2 clusters
  - 3 or more nodes in each cluster
    - Use replication to provide high availability
  - Replication available, throughput guarantee
- Development:
  - Low cost instance with 1 node cluster
  - No replication
- Create Compute Engine instance in same zone as Big Table instance

Resizing

- Add and remove nodes and clusters with no downtime.

Tools for interacting with BigTable

- cbt (Command Line tool)
  - Tool for doing basic interactions with BigTable
  - Use this if possible as it is simpler than HBase shell.
- HBase Shell
  - Command-line tool performs admin tasks such as creating and deleting tables.
  - Can update the following without any downtime:
    - Number of clusters/replication settings
    - Upgrade a development instance to production (permanent)

Architecture

- Entire BigTable project is called an instance.
- BigTable instance comprise of Clusters and Nodes
- Tables belong to instances
  - If multiple clusters, you cannot assign a table to an individual cluster

- Structure (1 Cluster below)
  - Data is never stored in BigTable nodes; each node has pointers to a set of tables stored on Colossus (GCS is built on this as well).
    - Rebalancing tablets from one node to another is very fast because the data is not actually copied. Pointers are simply updated.
    - Recovery from the failure of a node is very fast, only metadata needs to be migrated to the replacement node.

        - When BigTable fails, no data is lost.
    - Single table is sharded across multiple tablets.
- Compared to DataStore
    - BigTable queries are on the Key rather than an Index
    - BigTable supports atomicity only on a single row – no transactions

IAM

- Project wide or instance level
- Bigtable.admin
- Bigtable.user
    - App developer or service accounts.
- Bigtable.reader
    - Data scientists, dashboard generators, and other data analytics.
- Bigtable.viewer
    - Provides no data access.
    - Minimal set of conditions to access the GCP Console for BigTable.

BigQuery

- Hive equivalent
- No ACID properties
- Great for analytics/business intelligence/data warehouse (OLAP)
- Fully managed data warehouse
- Has connectors to BigTable, GCS, Google Drive, and can import from Datastore backups, CSV, JSON, and AVRO.
- Performance
    - Petabyte scale
    - High latency
        - Worse than BigTable and DataStore

Data Model

- Dataset = set of tables and views
- Table must belong to dataset
- Dataset must belong to a project

- Tables contain records with rows and columns (fields)
    - Nested and repeatable fields are OK

Table Schema

- Can be specified at creation time
- Can also specify schema during initial load
- Can update schema later too

Query

- Standard SQL (preferred) or Legacy SQL (old)
    - Standard
        - Table names can be referenced with backticks
            - Needed for wildcards
    - Cannot use both Legacy and SQL2011 in same query.
    - Table partitioning
    - Distributed writing to file for output (i.e. file-0001-of-0002)
    - User defined functions in JS (UDFJS)
        - Temporary – Can only use for current query or command line session.
    - Query jobs are actions executed asynchronously to load, export, query, or copy data.
    - If you use the LIMIT clause, BigQuery will still process the entire table.
    - Avoid SELECT * (full scan), select only columns needed (SELECT * EXCEPT)
    - Denormalized Data Benefits
        - Increases query speed
        - Makes queries simpler
        - BUT: Normalization makes dataset better organized, but less performance optimized.
    - Types
        - Interactive (default)
            - Query executed immediately
            - Counts towards
                - Daily usage
                - Concurrent usage
        - Batch
            - Scheduled to run whenever possible (idle resources)
            - Don’t count towards limit on concurrent usage.

        - If not started within 24hr, BQ makes them interactive.

Data Import

- Data is converted into columnar format for Capacitor.
- Batch (free)
    - web console (local files), GCS, GDS, Datastore backups (particularly logs)
    - Other Google services (i.e. Google Ad Manager, Google Ads)
- Streaming (costly)
    - Data with CDF, Cloud Logging, or POST calls
    - High volume event tracking logs
    - Realtime dashboards
    - Can stream data to datasets in both the US and EU
    - Streaming into ingestion-time partitioned tables:
        - Use tabledata.insertAll requests
        - Destination partition is inferred from current date based on UTC time.
            - Can override destination partition using a decorator like so: `mydataset.table$20170301`
        - Newly arriving data will be associated with the UNPARTITIONED partition while in the streaming buffer.
        - A query can therefore exclude data in the streaming buffer from a query by filtering out the NULL values from the UNPARTITIONED partition by using one of the pseudo-columns ([_PARTITIONTIME]) or [_PARTITIONDATE] depending on preferred data type.
    - Streaming to a partitioned table:
        - Can stream data into a table partitioned on a DATE or TIMESTAMP column that is between 1 year in the past and 6 months in the future.
        - Data between 7 days prior and 3 days in the future is placed in the streaming buffer, and then extracted to corresponding partitions.
        - Data outside that window (but within 1 year, 6 month range) is extracted to the UNPARTITIONED partition and loaded to corresponding partitions when there's enough data.
    - Creating tables automatically using template tables
        - Common usage pattern for streaming is to split a logical table into many smaller tables to create smaller sets of data.
        - To create smaller tables by date -> partitioned tables
        - To create smaller tables that are not date based -> template tables
            - BQ will create the tables for you.
            - Add templateSuffix parameter to your insertAll request.

        - <target_table_name> +
        - Only need to update template table schema then all subsequently generated tables will use the updated schema.
    - Quotas
        - Max row size: 1MB
        - HTTP request size limit: 10MB
        - Max rows per second: 100,000 rows/s for all tables combined.
        - Max bytes per second: 100MB/s per table
- Raw Files
    - Federated data source, CSV/JSON/Avro on GCS, Google sheets
- Google Drive
    - Loading is not currently supported.
    - Can query data in Drive using an external table.
- Expects all source data to be UTF-8 encoded.
- To support (occasionally) schema changing you can use automatically detect (not default setting).
    - Available while:
        - Loading data
        - Querying external data
- Web UI
    - Upload a file greater than 10MB in size
    - Upload multiple files at the same time
    - Upload a file in SQL format
    - Cannot load multiple files at once.
        - Can with CLI though.

Loading Compressed and Uncompressed Data

- Avro preferred for loading compressed data.
    - Faster to load since it can be read in parallel, even when data blocks are compressed.
- Parquet Binary format also a good choice
    - Efficient per-column encoding typically results in better compression ratio and smaller files.
- ORC Binary format offers benefits similar to Parquet
    - Fast to load because data stripes can be read in parallel.
    - Rows in each stripe are loaded sequentially.
    - To optimize load time: data stripe size of 256MB or less.
- CSV and JSON

    - BQ load uncompressed files significantly faster than compressed.
    - Uncompressed can be read in parallel.
    - Uncompressed are larger => bandwidth limitations and higher GCS costs for data staged prior to being loaded into BQ.
    - Line ordering not guaranteed for compressed or uncompressed.
- If bandwidth limited, compress with GCIP before uploading to GCS.
- If speed is important and you have a lot of bandwidth, leave uncompressed.

Loading Denormalized, Nested, and Repeated Data

- BQ performs best with denormalized data.
- Increases in storage costs worth the performance gains of denormalized data.
- Joins require data coordination (communication bandwidth)
    - Denormalization localizes the data to individual slots so execution can be done in parallel.
- If need to maintain data while denormalizing data
    - Use nested and repeated fields instead of completely flattening data.
    - When completely flattened, network communication (shuffling) can negatively impact query performance.
- Avoid denormalization when:
    - Have a star schema with frequently changing dimensions.
    - BQ complements an OLTP system with row-level mutation, but can't replace it.

BigQuery Transfer Service

- Automates loading data into BQ from Google Services:
    - Campaign Manager
    - Cloud Storage
    - Amazon S3
    - Google Ad Manager
    - Google Ads
    - Google Play
    - YouTube – Channel Reports
    - YouTube – Content Owner Reports

Partitions

- Improves query performance => reduces costs

- **Cannot change an existing table into a partitioned table.**
- **Types**
  - **Ingestion Time**
    - Partition based on data's ingestion date or arrived date.
    - Pseudo column `_PARTITIONTIME`
    - Reserved by BQ and can't be used.
    - Need to update schema of table before loading data if loading into a partition with a different schema.
  - **Partitioned Tables**
    - Tables that are partitioned based on a `TIMESTAMP` or `DATE` column.
    - 2 special partitions are created
      - **NULL** paritition
        - Represents rows with NULL values in the partitioning column
      - **UNPARTITIONED** partition
        - Represents data that exists outside the allowed range of dates
    - All data in partitioning column matches the date of the partition identifier with the exception of those 2 special partitions.
      - Allows query to determine which partitions contain no data that satisfies the filter conditions.
      - Queries that filter data on the partitioning column can restrict values and completely prune unnecessary partitions.
- **Wildcard tables**
  - Used if you want to union all similar tables with similar names. (i.e. project.dataset.Table*)
  - Filter in WHERE clause
    - AND _TABLE_SUFFIX BETWEEN 'table003' and 'table050'

Windowing

- Window functions increase the efficiency and reduce the complexity of queries that analyze partitions (windows) of a dataset by providing complex operations without the need for many intermediate calculations.
- Reduce the need for intermediate tables to store temporary data.

Bucketing

- Like partitioning, but each split/partition should be the same size and is based on the hash function of a column.

- Each bucket is a separate file, which makes for more efficient sampling and joining data.

Legacy vs. Standard SQL

- Standard: 'project.dataset.tablename*'
- Legacy: [project.dataset.tablename]
- It is **set each time you run a query**
- Default query language is
- Legacy SQL for classic UI
- Standard SQL for Beta UI

Anti-Patterns

- Avoid self joins
- Partition/Skew
  - Avoid unequally sized partitions
  - Values occurring more often than other values..
- Cross-Join
  - Joins that generate more outputs than inputs
- Update/Insert Single Row/Column
  - Avoid a specific DML, instead batch updates/inserts
- Anti-Patterns: https://cloud.google.com/bigtable/docs/schema-design

Table Types

- **Native Tables**
  - Backed by native BQ storage
- **External Tables**
  - Backed by storage external to BQ (federated data source)
  - BigTable, Cloud Storage, Google Drive
- **Views**
  - Virtual tables defined by SQL query.
  - Logical – not materialized
  - Underlying query will execute each time the view is accessed.
  - Benefits:
    - Reduce query complexity
    - Restrict access to data
    - Construct different logical tables from same physical table

- Cons:
  - Can't export data from a view
  - Can't use JSON API to retrieve data
  - Can't mix standard and legacy SQL
    - e.g. standard sql cannot access legacy sql view
  - No user-defined functions allowed
  - No wildcard table references
    - Due to partitioning
  - Limit of 1000 authorized views per dataset

Caching

- No charge for a query that retrieves results from cache.
- Results are cached for 24 hours.
- Caching is per user only.
- bq query –nouse_cache ''
- Cached by Default unless
  - A destination table is specified.
  - If any referenced tables or logical units have changed since results previously cached.
  - If any referenced tables have recently received streaming inserts even if no new rows have arrived.
  - If the query uses non-deterministic functions such as CURRENT_TIMESTAMP(), NOW(), CURRENT_USER()
  - Querying multiple tables using a wildcard
  - If the query runs against an external data source.

Export

- Destination has to be GCS.
  - Can copy table to another BigQuery dataset though.
- Can be exported as JSON/CSV/Avro
  - Default is CSV
- Only compression option: GZIP
  - Not supported for Avro
- To export > 1 GB
  - Need to put a wildcard in destination filename
  - Up to 1 GB of table data in a single file

- bq extract 'project:dataset.table' gs://bucket

Query Plan Explanation

- In web UI, click on "Explanation"
- Good for debugging complex queries not running as fast as needed/expected.
- Monitoring Query Performance (UI)
    - `Details` button after running query.
    - Colors
        - Yellow – Wait
        - Purple – Read
        - Orange – Compute
        - Blue – Write
    - Less parallel inputs => better performance => best cost

Slots

- Unit of computational capacity needed to run queries.
- BQ calculates on basis of query size, complexity
- Usually default slots are sufficient
- Might need to be expanded over time, complex queries
- Subject to quota policies ($$)
- Can use StackDriver Monitoring to track slot usage.

Clustered Tables

- Order of columns determines sort order of data.
- Think of Clustering Columns in Cassandra
- When to use:
    - Data is already partitioned on date or timestamp column.
    - You commonly use filters or aggregation against particular columns in your queries.
- Does not work if the clustered column is used in a complex filter (used in a function in the filter expression)

BigQuery ML

- Create and execute machine learning models in BQ using standard SQL
- Supported models
    - Linear regression
    - Binary Logistic regression
    - Multiclass logistic regression for classification
    - K-means clustering for data segmentation (beta)
    - Import TensorFlow Models (alpha)
- TensorFlow DNN
    - Classifier
    - Regressor
- Benefits from not having to export and re-format data
    - Brings machine learning to the data.

Best Practices

Costs

- Avoid SELECT *
    - Query only columns you need.
- Sample data using preview options
    - Don't run queries to explore or preview table data.
- Price your queries before running them.
    - Before running queries, preview them to estimate costs.
- Limit query costs by restricting the number of bytes billed.
    - Use the maximum bytes billed setting to limit query costs.
- LIMIT doesn't affect cost
    - Do not use LIMIT clause as a method of cost control as it does not affect the amount of data that is read.
- View costs using a dashboard and query your audit logs
    - Create a dashboard to view your billing data so you can make adjustments to your BigQuery usage. Also consider streaming audit logs to BigQuery to analyze usage patterns.
- Partition data by date
- Materialize query results in stages
    - Break large query into stages where each stage materializes the results by writing to a destination table.
    - Querying smaller destination table reduces amount of data that is read and lowers costs.
- Consider cost of large result sets

    - Use default table expiration time to remove data when not needed.
    - Good for when writing large query results to a destination table.
- Use streaming inserts with caution
    - Only use if data is needed immediately available.

Query Performance

- Input data and data sources (I/O)
    - Control projection – Avoid SELECT *
    - Prune partitioned queries
        - Use partition columns to filter
    - Denormalize data when possible
        - JSON, Parquet, or Avro
        - When creating, specify Type in the Schema as RECORD
    - Use external data sources appropriately
        - If performance is a top priority, do not use external source
    - Avoid excessive wildcard tables
        - Use most granular prefix possible
- Communication between nodes (shuffling)
    - Reduce data before using a JOIN
    - Do not treat WITH clauses as prepared statements
    - Avoid tables sharded by date
        - Use time-based partitioned tables instead
            - Copy of schema and metadata is maintained for each sharded table.
            - BQ might have to verify permissions for each queries table. (overhead)
    - Avoid oversharding tables
- Computation
    - Avoid repeatedly transforming data via SQL queries
    - Avoid JavaScript user-defined functions.
        - Use native UDFs instead.
    - Use approximate aggregation functions
        - COUNT(DISTINCT) vs. APPROX_COUNT_DISTINCT()
    - Order query operations to maximize performance
        - Only use in the outermost query or within window clauses.
        - Push complex operations to the end of the query.
    - Optimize join patterns
        - Start with the largest table

    - Prune partitioned queries
- Outputs (materialization)
    - Avoid repeated joins and subqueries
    - Carefully consider materializing large result sets
    - Use LIMIT clause with large sorts
- Anti-patterns
    - Self-joins
        - Potentially doubles number of output rows
        - Use window function instead
    - Data skew
        - If query processes keys that are heavily skewed to a few values, filter your data as early as possible.
    - Cross joins (Cartesian product)
        - Avoid joins that generate more outputs than inputs.
        - Pre-aggregate data first if it is required.
    - DML statements that update or insert single rows
        - Use batch.

Storage Optimization

- Use expiration settings to remove unneeded tables and partitions
    - Configure default table expiration for datasets
    - Configure expiration time for tables
    - Configure partition expiration for partitioned tables
- Take advantage of long term storage
    - Untouched tables (90 days) are as cheap as GCS Nearline
    - Each partition is considered separately.
- Use pricing calculator to estimate storage costs

Architecture

- Jobs (queries) can scale up to thousands of CPU's across many nodes, but the process is completely invisible to end user.
- Storage and compute are separated, connected by petabit network.
- Columnar data store
    - Separates records into column values, stores each value on different storage volume.
    - Poor writes (BQ does not update existing records)

Components

- Dremel - Execution Engine
    - Turns SQL query into an execution tree.
    - Leaves on the tree are 'slots'
        - Does the heavy lifting of reading data from Colossus and doing any computation necessary.
    - Branches of tree are 'mixers'
        - Perform aggregation
    - In between is 'shuffle'
        - Uses Google's Jupiter network to move data extremely rapidly from one place to another.
    - Mixers and Slots all run by Borg
        - Doles out hardware resources.
    - Dynamically appropriates slots to queries on an as needed basis, maintaining fairness amongst multiple users who are all querying at once.
    - Widely used at Google
        - Search
        - Ads
        - YouTube
        - BigQuery
- Colossus - Distributed Storage
    - Google's latest generation distributed file system.
    - Colossus cluster in each Google datacenter.
        - Each cluster has enough disks to give every BigQuery user thousands of dedicated disks at a time.
    - Handles replication, recovery (when disk crash), and distributed management (so no single point of failure).
    - Leverages columnar storage
- Borg - Compute
    - Gives thousands of CPU cores dedicated to processing your task.
    - Large-scale cluster management system.
    - Run on dozens of thousands of machines and hundreds of thousands of cores.
    - Assigns resources to jobs - Dremel cluster in this scenario.
- Jupiter - The Network

    - Can deliver 1 Petabit/sec of total bisection bandwidth.
        - Enough to allow 100,000 machines to communicate with any other machines at 10Gbs.
    - Full duplex bandwidth means that locality within cluster is not important.
        - Each machine can talk at 10Gbs regardless of rack.

Cost

- Based on:
    - storage (amount of data stored)
    - querying (amount of data/number of bytes processed by query)
    - streaming inserts.
- Storage options are active and long term
    - Modified or not past 90 days
- Query options are on-demand and flat-rate

IAM

- Security can be applied at project and dataset level, but not at table or view level.
- Predefined roles BQ
    - Admin – Full access
    - Data owner – Full dataset access
    - Data editor – edit dataset tables
    - Data viewer – view datasets and tables
    - Job User – run jobs
    - User – run queries and create datasets (but not tables)
    - metaDataViewer
    - readSessionUser – Create and use read sessions within project.
- **Authorized views** allow you to share query results with particular users/groups without giving them access to underlying data.
    - Restrict access to **particular columns or rows**
    - Create a **separate dataset** to store the view.
    - How:
        - Grant IAM role for data analysts (bigquery.user)
            - They won't have access to query data, view table data, or view table schema details for datasets they did not create.
        - (In source dataset) Share the dataset, In permissions go to Authorized views tab.
            - View gets access to source data, not analyst group.

Data Processing

Streams Introduction

- How can MapReduce be used to maintain a running summary of real-time data from sensors?
    - Send temp readings every 5 minutes

Batches

- Bounded datasets
- Slow pipeline from data ingestion to analysis
- Periodic updates as jobs complete
- Order of data received unimportant
- Single global state of the world at any point in time
- Typically small/singular source
- Low latency not important
- Often stored in storage services GCS, Cloud SQL, BigQuery

Streams

- Unbounded datasets
- Processing immediate, as data is received
- Continuous updates as jobs run constantly
- Order important, but out of order arrival tracked
- No global state, only history of events received
- Typically many sources sending tiny (KB) amounts of data
- Requires low latency
- Typically paired with Pub/Sub (ingest) and Dataflow (real-time processing)

Process data one entity at a time or a collection of entities as a batch

- Filter error messages (logs)
- Find a reference to latest movies (tweets)

- Track weather patterns (sensor data)

Store, display, act on filtered messages

- Trigger an alert
- Show trending graphs
- Warn of sudden squalls

Stream-First Architecture

- Data items can come from multiple sources
  - Files, DBs, but at least one from a Stream
- All files are aggregated and buffered in one way by a Message Transport (Queue)
  - i.e. Kafka, PubSub
- Passed to Stream Processing system
  - Flink or Spark Streaming

Micro-batches

- Message Transport
  - Buffer for event data
  - Performant and persistent
  - Decoupling multiple source from processing
- Stream Processing
  - High throughput, low latency
  - Fault tolerant with low overhead
  - Manage out of order events
  - Easy to use, maintainable
  - Replay streams
- A good approximation of stream processing is the use of micro-batches
  - Group data items (time they were received)
  - If small enough it approximates real-time stream processing
- Advantages
  - Exactly once semantics, replay micro-batches
  - Latency-throughput trade off based on batch sizes
    - Can adjust to use case
    - Low latency better
    - High throughput better

- Spark Streaming or Storm Trident

Dataflow

- Executes Apache Beam Pipelines
- Can be used for batch or stream data
- Scalable, fault-tolerant, multi-step processing of data.
- Often used for data preparation/ETL for data sets.
- Integrates with other tools (GCP and external)
    - Natively – PubSub, BigQuery, Cloud AI Platform
        - BQ I/O connector can stream JSON through Dataflow to BQ
    - Connectors – BigTable, Apache Kafka
- Pipelines are regional-based
- Follows the Flink Programming Model
    - Data Source -> Transformations -> Data Sink
- Use when:
    - No dependencies on Apache Hadoop/Spark
    - Favor hands-off/serverless
    - Preprocessing for machine learning with Cloud ML Engine
- Templates
    - Google provided templates.
        - WordCount
        - Bulk compress/decompress GCS Files
        - PubSub to (Avro, PubSub, GCS Text, BigQuery)
        - Datastore to GCS Text
        - GCS Text to (BigQuery, PubSub, DataStore)
- Requires a `Staging Location` where intermediate files may be stored.
- **System Lag**
    - Max time an element has been waiting for processing in this stage of the pipeline.

- **Wall Time**
    - How long the processing takes.

Apache Beam Architecture

- **Pipeline**
    - Entire set of computations
    - Not linear, it is a DAG
    - Beam programs start by constructing a Pipeline object.
- A single, potentially repeatable job, from start to finish, in Dataflow.
- Defined by driver program.
    - Actual computations run on a backend, abstracted in the driver by a runner.
        - Driver: Defines DAG
        - Runner: Executes DAG
- Supports multiple backends
    - Spark
    - Flink
    - Dataflow
    - Beam Model
- **Element**
    - A single entry of data (e.g. table row)
- **PCollection**
    - Distributed data set in pipeline (immutable)
    - Specialized container classes that can represent data sets of virtually unlimited size.
        - Fixed size: Text file or BQ table
        - Unbounded: Pub/Sub subscription
    - Side inputs
        - Inject additional data into some PCollection
        - Can inject in ParDo transforms.
- **PTransform**
    - Data processing operation (step) in pipeline
        - Input: 1 or more PCollection
        - Processing function on elements of PCcollection
        - Output: 1 or more PCollection
- **ParDo**
    - Core of parallel processing in Beam SDKs
    - Collects the zero or more output elements into an output PCollection.

- Useful for a variety of common data processing operations, including:
  - Filtering a data set.
    - Better than a Filter transform in the sense that a filter transform can only filter based on input element => no side input allowed.
  - Formatting or type-converting each element in a data set.
  - Extracting parts of each element in a data set.
  - Performing computations on each element in a data set.

Dealing with late/out of order data

- Latency is to be expected (network latency, processing time, etc.)
- PubSub does not care about late data, that is resolved in Dataflow.
- Resolved with Windows, Watermarks, and Triggers.
- **Windows** = Logically divides element groups by time span.
- **Watermarks** = Timestamp
  - Event time – When data was generated
  - Processing time – when data processed anywhere in pipeline
  - Can use Pub/Sub provided watermark or source generated.
- **Trigger** = Determine when results in window are emitted.
  - (Submitted as complete)
  - Allow late-arriving data in allowed time window to re-aggregate previously submitted results.
  - Timestamps, element count, combinations of both.

Stopping a Dataflow Jobs

Cancelling

- Immediately stop and abort all data ingestion and processing.
- Buffered data may be lost.

Draining

- Cease ingestion but will attempt to finish processing any remaining buffered data.
- Pipeline resources will be maintained until buffered data has finished processing and any pending output has finished writing.

Pipeline Update

- Replace an existing pipeline in-place with a new one and preserve Dataflow's exactly-once processing guarantee.
- When updating pipeline manually, use DRAIN instead of CANCEL to maintain in flight data.
    - Drain command is supported for streaming pipelines only
- Pipelines cannot share data or transforms.

Handling Invalid Inputs in Dataflow

- Catch exception, log an error, then drop input.
    - Not ideal
- Have a dead letter file where all failing inputs are written for later analysis and reprocessing.
    - Add a side output in Dataflow to accomplish this.

Converting from Kafka to PubSub

- CloudPubSubConnector is a connector to be used with Kafka Connect to publish messages from Kafka to PubSub and vice versa.
- Provides both a sink connector (to copy messages from Kafka to PubSub) and a source connector (to copy messages from PubSub to Kafka.

Windowing

- Can apply windowning to streams for rolling average for the window, max in a window etc.

Fixed Time Windows (Tumbling Window)

- Fixed window size
- Non-overlapping time
- Number of entities differ within a window

Sliding Time Windows (overlapped)

- Fixed window size

- Overlapping time
- Number of entities differ within a window
- Window Interval: How large window is
- Sliding Interval: How much window moves over

Session Windows

- Changing window size based on session data
- No overlapping time
- Number of entities differ within a window
- Session gap determines window size
- Per-key basis
- Useful for data that is irregularly distributed with respect to time.

Single Global Window

- Late data is discarded
- Okay for bounded size data
- Can be used with unbounded but use with caution when applying transforms such as GroupByKey and Combine
- Default windowing behavior is to assign all elements of a PCollection to a single, global window even for unbounded PCollections.

Triggers

- Determines when a Window's contents should be output based on a certain being met.
    - Allows specifying a trigger to control when (in processing time) results for the given window can be produced.
    - If unspecified, the default behavior is to trigger first when the watermark passes the end of the window, and then trigger again every time there is late arriving data.

Time-Based Trigger

Event Time Triggers

- Operate on event time, as indicated by timestamp on each data elements.
- This is the default trigger.

Processing Time Triggers

- Operate on the processing time – the time when the data element is processed at any given stage in the pipeline.

Data-Driven Trigger

- Operate by examining the data as it arrives in each window, and firing when that data meets a certain property.
- Currently, only support firing after a certain number of data elements.

Composite Triggers

- Combine multiple triggers in various ways.

Watermarks

- System's notion of when all data in a certain window can be expected to have arrived in the pipeline.
- Tracks watermark because data is not guaranteed to arrive in a pipeline in order or at predictable intervals.
- No guarantees about ordering.
- Indicates all windows ending before or at this timestamp are closed.
- No longer accept any streaming entities that are before this timestamp.
- For unbounded data, results are emitted when the watermark passes the end of the window, indicating that the system believes all input data for that window has been processed.
- Used with Processing Time

IAM

- Project-level only – all pipelines in the project (or none)
- Pipeline data access separate from pipeline access.
- Dataflow Admin
    - Full pipeline access
    - Machine type/storage bucket config access
- Dataflow.developer
    - Full pipeline access
    - No machine type/storage bucket access (data privacy)

- Dataflow Viewer
  - View permissions only.
- Dataflow.worker
  - Enables service account to execute work units for a Dataflow pipeline in Compute Engine.
  - Dataflow API also needs to be enabled.

Dataproc

- Managed Hadoop (Spark, SparkML, Hive, Pig, etc…)
- Automated cluster management, resizing
- Code/Query only
- Job management screen in the console
- Think in terms of a 'job specific resource' – for each job, create a cluster and then delete it.
- Used if **migrating existing on-premise Hadoop or Spark infrastructure** to GCP without redevelopment effort.
- Can sale even when jobs are running.
- Use Dataflow for streaming instead. **This is better for batch.**

- Connecting to Web Interface of Dataproc Cluster
  - Allow necessary web ports access via firewall rules, and limit access to your network.
    - Tcp:8088 (Cluster Manager)
      - :8088
    - Tcp:50070 (Connect to HDFS name node)
      - :50070
  - OR SOCKS proxy (routes through an SSH tunnel for secure access)
    - gcloud compute ssh [master_node_name]

Cost

- Standard Compute Engine machine type pricing + managed Dataproc premium.
- Premium = $0.01 per vCPU core/hour
- Billed by the second, with a minimum of 1 minute.

Storage

- Can use on disk (HDFS) or GCS

HDFS

- Split up on the cluster, but requires cluster to be up.

GCS

- Allows for the use of preemptible machines that can reduce costs significantly.
    - DO not need to configure startup and shutdown scripts to gracefully handle shutdown, Dataproc already handles this.
    - Cluster MUST have at least 2 standard worker nodes however.
- Separate cluster and storage.
- Cloud Storage Connector
    - Allows you to run Hadoop or Spark jobs directly on GCS.
    - Quick startup
        - In HDFS, a MapReduce job can't start until the NameNode is out of safe mode.
        - With GCS, can start job as soon as the task nodes start, leading to significant cost savings over time.

- Cluster Machine Types
    - Build using Compute Engine VM instances
    - Cluster – need at least 1 master and 2 workers
- High Availability Mode
    - 3 masters rather than 1
    - 3 masters run in an Apache Zookeeper cluster for automatic failover.
- Restartable Jobs

- Jobs do NOT restart on failure (default)
- Can change this – useful for long running and streaming jobs (ex. Spark Streaming)
- Mitigates out-of-memory errors, unscheduled reboots

Updating Clusters

- Can only change # workers/preemptible VM's/labels/toggle graceful decommission.
  - Graceful Decommissioning
    - Finish work in progress on a worker before it is removed from Dataproc cluster.
    - Incorporates graceful YARN decommissioning.
    - May fail for preemptible workers.
      - Can forcefully decommission preemptible workers at any time.
    - Will always work with primary workers.
    - gcloud dataproc clusters update –gracefult-decommission-timeout
      - Default to "0s" – forceful decommissioning.
      - Need to provide a time.
      - Max value 1 day.
  - Automatically reshards data for you.

Migrating and Optimizing for GCP Best Practices

- Move data first
  - Generally to GCS buckets.
  - Possible exceptions
    - Apache HBase data to BigTable
    - Apache Impala to BigQuery
    - Can still choose to move to GCS if BigTable/BQ feature not needed.
- Small scale experimentation
  - Use a subset of data to test.
- Think of it in terms of ephemeral clusters.
- Use GCP tools to optimize and save costs.

Converting from HDFS to GCS

- Copy data to GCS
  - Install connector or copy manually

- Update file prefix in scripts
    - Hdfs:// to gs://
- Use Dataproc, and run against/output to GCS

Connectors

- BQ/BigTable (copies data to GCS) /CloudStorage

Optional Components

- Anaconda, Druid
- Hive WebHCat
- Jupyter
- Kerberos
- Presto
- Zeppelin
- Zookeeper

- How to configure Hadoop to use all cores?
    - Think spark executor cores
- How to handle out of memory errors?
    - Hint - Executor memory
- How to install other components?
    - Hint – Initialization actions

IAM

- Project level only (primitive and predefined roles)
- Cloud Dataproc Editor, Viewer, and Worker
    - Editor – Full access to create/edit/delete clusters/jobs/workflows
    - Viewer – View access only
    - Worker – Assigned to service accounts
        - Read/write GCS, write to Cloud Logging

Hadoop

Distributed Computing

- Lots of cheap hardware
  - HDFS
- Replication and Fault Tolerance
  - YARN
- Distributed Computing
  - MapReduce

HDFS

- GCS is used on GCP.
  - Don't use HDFS as you would have to pay for a VM on Compute Engine.
- Suited for batch processing.
  - Data access has high throughput rather than low latency.

Architecture

- **Name Node**
  - 1 master node
  - Contains YARN resource manager
  - Manages overall file system
  - Stores
    - The directory structure
    - Metadata on the files
- **Data Nodes**
  - Physically stores the data in the files.

Storing Data

- Break data into blocks of equal size
  - Different length files are treated the same way
  - Storage is simplified
  - Unit for replication and fault tolerance
- Blocks are of size 128 MB
  - Larger -> Reduces parallelism

    - Smaller -> Increases overhead (more metadata)
- Stores the blocks across the data nodes
    - Each node contains a partition or a split of data
    - How do we know where the splits of a particular file are?
        - Name Node (File 1 | Block 1 | Data Node)

High Availability

- Can have multiple name nodes.
- Kept in sync with Zookeeper

Default Replication Strategy

- Maximize Redundancy
    - 1st location chosen at random
    - 2nd has to be on a different rack (if possible)
    - 3rd will be on same rack as the second, but on a different node.
        - Reduces inter-rack traffic and improves write performance.
    - Read operations are sent to the rack closest to the client.
- Minimize Write Bandwidth
    - Data is forwarded from first data node to the next replica location.
    - Forwarded further to the next replica location.
    - Forwarding requires a large amount of bandwidth.
        - Increases cost of writes.

MapReduce

Map

- An operation performed in parallel, on small portions of dataset.
- Outputs KV pairs

Reduce

- Mapper outputs become one final output.

SQL interface over MapReduce = Hive

- Data analysts understand SQL but not Java code.

1. What {key, value} pairs should be emitted in the map step?
2. How should values with the same key be combined?

YARN (Yet Another Resource Negotiator)

- Coordinate tasks running on the cluster.
- Assign new nodes in case of failure.

Architecture

Resource Manager

- Runs on a single master node
- Schedules tasks across nodes
- Starts Application Master within containers.

Node Manager

- Run on all other nodes
- Manages tasks on the individual node.
- Can have multiple containers.
- Can request containers for mappers and reducers.

Application Master

- If additional resources are required, Application Master makes the request.
- 1 instance per application.
- Client communicates directly to get status, progress updates via an application-specific protocol.

Container

- All processes are run within a container in a Node Manager.
- Package of resources including RAM, CPU, Network, HDD etc on a single node.
- Executes the application code.
- Can communicate with Application Master itself.

Location Constraint

- Assign a process to the same node where the data to be processed lives.
- If CPU/Memory not available, WAIT!

Scheduling Policies

- FIFO Scheduler
  - Queue
- Capacity Scheduler
  - Priority Queue
- Fair Scheduler
  - Jobs assigned equal share of all resources

HBase

- Database management system on top of Hadoop.
- Integrates with your application just like a traditional database.

Columnar Store

- Advantages
  - Sparse Tables
    - No wastage of space when storing data.
  - Dynamic Attributes
    - Update attributes dynamically without changing storage structure.
    - Do not need to change schema.

Denormalized Storage

- Column names repeat across rows.
- Normalization Reduces data duplication => Optimizes storage.
  - Storage is cheap in a distributed file system.
  - Optimize number of disk seeks instead by denormalization.
    - Don't have to join tables.
- Read a single record to get all details about an employee in one read operation.

Only CRUD operations

- No comparisons/sorting/inequality checks across multiple rows
  - No joins

    - No group by
    - No order by
- No operations involving multiple tables
- No indexes on tables
- No constraints

ACID at ROW level

- Updates to a single row are atomic
    - All columns are updated, or none are.
- Updates to multiple rows are not atomic
    - Even if update is on the same column in multiple rows.

Hive

- Provides a SQL interface to Hadoop.
- Bridge to Hadoop for people without OOP exposure.
- Not suitable for very low latency apps due to HDFS.
- HiveQL ~= SQL
- Wrapper on top of MapReduce

Metastore

- HCatalog
- Bridge between HDFS and Hive
- Stores metadata for all tables in Hive
- Maps the files and directories in Hive to tables
- Holds the definitions and the schema for each table
- Any database with a JDBC driver can be used as a metastore.
- Development
    - Use built-in Derby database
    - Embedded metastore
    - Only one session can connect.
- Production
    - Local metastores
        - Allow multiple sessions to connect to Hive
        - DB is a separate process and can be on separate host.
    - Remote metastores

        - Separate processes for Hive and the metastore
        - Metastore runs in its own JVM process.
        - Processes communicate with Metastore using Thrift network API (hive.metastore.uris property)
        - Does not require admin to share JDBC login info for the metastore db with each Hive user.
- Hive vs. RDBMS
    - Large vs. Small datasets
    - Parallel vs. serial computations
    - High vs. low latency
    - Read vs. Read/write operations
    - Not ACID compliant vs. ACID compliant
- HiveQL vs. SQL
    - High latency
        - Records not indexed.
        - Fetching a row runs a MapReduce which may take minutes.
        - Not owner of the data.
            - It exists in HDFS
        - Schema-on-read
    - Not ACID compliant
        - Data can be dumped into Hive tables from any source
    - Row level updates, deletes as a special case
    - Many more built in functions
    - Only equi-joins allowed
- OLAP in Hive
    - Partitioning
        - State specific queries will run only on data in one directory.
        - Splits NOT of the same size.
    - Bucketing
        - Size of each split should be the same.
            - Hash of a column value
        - Each bucket is a separate file
        - Makes sampling and joining data more efficient
            - Reduces search space
    - Join Optimizations
        - Join operations are Map Reduce jobs under the hood
            - Optimize joins by reducing the amount of data held in memory
        - Reducing data held in memory

        - On a join, one table is held in memory while the other is read from disk
            - Hold smaller in memory
    - Structuring Joins as Map-Only Operation
        - Filter queries (only these rows)
            - Mapper needs to use null as key
- Windowing in Hive
    - A suite of functions which are syntactic sugar for complex queries.
    - e.x. What revenue percentile did this supplier fall into this quarter?
        - Window = 1 quarter
        - Operation = Percentile on revenue

Pig

- ETL
- A data manipulation language
- Transforms unstructured data into a structured format
- Query this structured data using interfaces like Hive.
- Raw Data -> Pig -> Warehouse -> HiveQL -> Analytics
- Pig Latin
    - A procedural, data flow language to extract, transform and load.
        - Procedural
            - Uses a series of well-defined steps to perform operations.
            - No if statements or for loops.
            - Specifies exactly how data is to be modified at each step.
        - Data Flow
            - Focused on transformations applied to the data.
            - Written with a series of data operations in mind.
            - Nodes in a DAG
    - Data from one or more sources can be read, processed and stored in parallel.
    - Cleans data, precomputes common aggregates before storing in a data warehouse.
- Pig on Hadoop
    - Optimizes operations before MapReduce jobs are run, to speed operations up.
- Works better with Apache Tez and Spark.

Oozie

- A tool used to schedule workflows on all the Hadoop ecosystem technologies.

Apache Spark

- A distributed computing engine used along with Hadoop
- Interactive shell to quickly process datasets
- Has a bunch of built in libraries for machine learning, stream processing, graph processing, …, etc.
- Dataflow
- General purpose
    - Exploring
    - Cleaning and Preparing
    - Applying machine learning
    - Building data applications
- Interactive
    - Provides a REPL environment
        - Read Evaluate Print Loop
- Reduces boilerplate of standard MapReduce Java code.

Resilient Distributed Datasets (RDDs)

- In memory collections of objects.
- Can interact with billions of rows
- Properties
    - Partitions
    - Read-only
        - Immutable
        - Operations allowed on RDD
            - Transformations
                - Transform into another RDD
            - Actions
                - Request a result
        - Aware of it's Lineage
            - When created, RDD knows
                - A transformation
                - It's parent RDD
            - Implications of Lineage
                - Built in fault tolerance
                    - Reconstruct from source if something goes wrong

- Lazy Evaluation
  - Materialize only when necessary

Spark Core

- Basic functionality of Spark
- Written in Scala
- Runs on a Storage System and Cluster Manager
  - Plug and play components
  - Can be HDFS and YARN

Spark ML

- MLlib is Spark's machine learning library.
- Provides tools such as:
  - ML Algorithms: classification, regression, clustering, collaborative filtering.
  - Featurization: feature extraction, transformation, dimensionality reduction, and selection
  - Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
  - Persistence: saving and load algorithms, models, and Pipelines
  - Utilities: linear algebra, statistics, data handling, etc.

PubSub

- Server-less messaging "middleware"
- Many to many asynchronous messaging
- Decouples sender and receiver
- Attributes can be set by sender (KV pairs)
- Glue that connects all components
- Order not guaranteed
- Encoding as a Bytestring (utf-8) required for publishing.
- Publishers: Any app that can make HTTPS requests to googleapis.com

Message Flow

- Publisher app creates a topic object and sends a message to the topic.
- Messages persisted in message store until acknowledged by subscribers

- Messages forwarded from topic to all subscriptions individually.
- Subscriber receives pending messages from its subscription and acknowledges each one to the Cloud Pub/Sub service.
    - Push
        - WebHook endpoint (must accept POST HTTPS request)
        - Lower latency, more real time.
    - Pull
        - Subscriber explicitly calls pull method which requests messages for delivery.
        - More efficient message deliver/consume mechanism
        - Better for large volume of messages – batch delivery.
    - Acknowledgement Deadline
        - Per subscriber
        - Once a deadline has passed, an outstanding message becomes unacknowledged.
- When acknowledged, it is removed from the subscriptions message queue.

Architecture

- Data Plane
    - Handles moving messages between publishers and subscribers
    - Forwarders
- Control Plane
    - Handles assignment of publishers and subscribers to server on the data plane.
    - Routers

Use Cases

- Balancing workloads in a network cluster
- Implementing async workflows
- Distributing event notifications
- Refreshing distributed caches
    - i.e. An app can publish invalidation events to update the IDs of objects that have changed
- Logging to multiple systems
- Data streaming from various processes or devices
- Reliability improvement
    - i.e. a single-zone GCE service can operate in additional zones by subscribing to a common topic, to recover from failures in a zone or region.

Deduplicate

- Database table to store hash value and other metadata for each data entry.
- Message_id can be used to detect duplicate messages

Out of Order Messaging

- Messages may arrive from multiple sources out of order.
- Pub/Sub does not care about message ordering
- Dataflow is where out of order messages are processed/resolved.
    - Ingest – Pub/Sub
    - Process - Dataflow
- Can add message attributes to help with ordering.

Handling Order

- Order does not matter at all
    - i.e. queue of independent tasks, collection of statistics on events
    - Perfect for PubSub. No extra work needed.
- Order in final result matters
    - i.e. Logs, state updates
    - Can attach a timestamp to every event in the publisher and make the subscriber store the messages in some underlying data store (such as Datastore) that allows storage or retrieval by the sorted timestamp.
- Order of processed messages matters
    - i.e. transactional data where thresholds must be enforced
    - Subscriber must either:
        - Know the entire list of outstanding messages and the order in which they must be processed, or
            - Assigning each message a unique identifier and storing in some persistent place (Datastore) the order in which messages should be processed.
            - Subscriber check persistent storage to know the next message it must process and ensure that it only processes that message next.
        - Have a way to determine all messages it has currently received whether or not there are messages it has not received that it needs to process first.
            - Cloud Monitoring to keep track of the oldest_unacked_message_age metric.

        - Temporarily put all messages in some persistent storage and ack the messages.
        - Periodically check the oldest unacked message age and check against the publish timestamps of the messages in storage.
        - All messages published before the oldest unacked message are guaranteed to have been received, so those messages can be removed from the persistent storage and processed in order.
    - Single synchronous publisher/subscriber
        - Can just use a sequence number to ensure ordering.
        - Requires the use of a persistent counter.

Cost

- Data volume used per month (per GB)

IAM

- Control access at project, topic, or subscription level
- Resource types: Topic, Subscription, Project
- Service accounts are best practice.
- Pubsub.publisher
- Pubsub.subscriber
- Pubsub.viewer or viwer
- Pubsub.editor or editor
- Pubsub.admin or admin

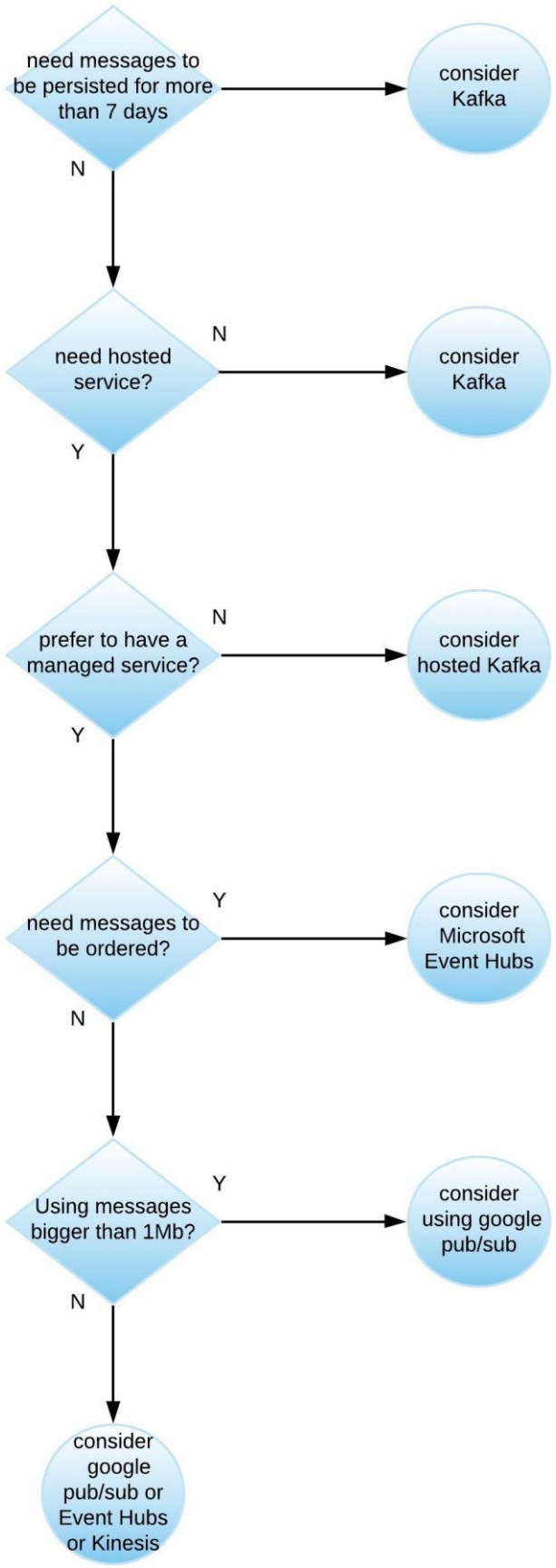
Apache Kafka

- Stream processing for unbounded datasets.
- Similar to PubSub
- Kafka Connect
    - A tool for scalably and reliably streaming data between Apache Kafka and other systems.
    - There are connectors to PubSub/Dataflow/BigQuery

Compared to PubSub

- Can have precisely once delivery with Spark direct Connector in addition to at least once.

    - Only at least once with PubSub
- Guaranteed ordering within a partition.
    - No ordering guaranteed with PubSub
- No max persistence period
    - 7 days or until acknowledged by all subscribers for PubSub
- Partitioning under user control
    - Partitioning control abstracted away with PubSub
- Cluster Mirroring for disaster recovery
    - Automated disaster recover for PubSub
- 1MB max size for data blobs
    - 10 MB max size for PubSub
- Can change partitioning after setup (does not repartition existing data)
    - Not under user control with PubSub
- Pseudo push model supported using Spark.


need messages to be persisted for more than 7 days
consider Kafka
N
need hosted service?
N
consider Kafka
Y
prefer to have a managed service?
N
consider hosted Kafka
Y
need messages to be ordered?
Y
consider Microsoft Event Hubs
N
Using messages bigger than 1Mb?
Y
consider using google pub/sub
N
consider google pub/sub or Event Hubs or Kinesis

Cloud Dataprep

- Intelligent data preparation
- Partnered with Trifecta for data cleaning/processing service
- Fully managed, serverless, and web based
- User friendly interface
    - Clean data by clicking on it
- Supported file types
    - Inputs
        - CSV, JSON, Plain Text, Excel, LOG, TSV, and Avro
    - Outputs
        - CSV, JSON, Avro, BQ Table
            - CSV/JSON can be compressed or uncompressed

How it works

- Backed by Cloud Dataflow
    - After preparing Dataflow processes via Apache Beam pipeline
    - "User-friendly Dataflow pipeline"
- Dataprep Process
    - Import data
    - Transform sampled data with recipes
    - Run Dataflow job on transformed dataset
        - Batch Job
        - Every recipe step is its own transform
    - Export results (GCS, BigQuery)
- Intelligent Suggestions
    - Selecting data will often automatically give the best suggestion
    - Can manually create recipes, however simple tasks (remove outliers, de-duplicate) should just use auto-suggestions.

IAM

- Dataprep.projects.user
    - Run Dataprep in a project
- Dataprep.serviceAgent
    - Gives Trifecta necessary access to project resources.
        - Access GCS buckets, Dataflow Developer, BQ user/data editor
        - Necessary for cross-project access + GCE service account

Cost

- 1.16 * cost of a Dataflow job

Flows

- Add or import datasets to process with recipes
- Public Bucket for testing: gs://dataprep-samples
- For large datasets:
    - UI only shows a sample to work with
    - Recipe created is then applied to entirety of dataset

Jobs

- Create a dataset in BQ first
- Click on Run Job
    - Default option is CSV in GCS bucket
    - Choose BQ dataset instead
    - Name table
    - Run Job: Create Apache Beam pipeline with Dataflow

Cloud Composer

- Fully managed workflow orchestration service based on Apache Airflow.
    - No need to provision resources.
- Pipelines are configured as DAGs
- Workflows live on-premises, in multiple clouds, or full within GCP
- Provides ability to author, schedule, and monitor your workflows in a unified manner.
- Multi-cloud
- Can use Python to dynamically author and schedule workflows.

Environments

- Airflow is a micro-service architected framework.
    - To deploy in a distributed setup, Cloud Composer provisions several GCP components, collectively known as an Environment.
- Can create one or more inside a project.

- Self contained Airflow deployments based on GKE.
- Work with GCP services through connectors built into Airflow.

Architecture

- Distributes environment's resource between a Google-managed tenant project and a customer project.
- For unified Cloud IAM access control and an additional layer of data security, Cloud Composer deploys Cloud SQL and App Engine in the tenant project.

Tenant Project

- Cloud SQL
  - Stores Airflow metadata.
  - Composer limits database access to the default or specified custom service account used to create the environment.
  - Metadata backed up daily.
  - Cloud SQL proxy in GKE cluster
    - Used to remotely authorize access to your Cloud SQL database from an application, client, or other GCP service.
- App Engine
  - Hosts the Airflow web server.
  - Integrated with Cloud IAM by default.
  - Assign composer.user role to grant access only to Airflow web server.
  - Can deploy a self-managed Airflow web server in customer project (for orgs with additional access-control reqs)

Customer Project

- Cloud Storage
  - Used for staging DAGs, plugins, data dependencies, and logs.
  - To deploy workflows (DAGs), copy files to the bucket for you environment.
  - Composer takes care of synchronizing DAGs among workers, schedulers, and the web server.
- GKE
  - Scheduler, worker nodes, and CeleryExecutor here.
  - Redis

        - Message broker for the CeleryExecutor
            - Runs a StatefulSet application so that messages persist across container restarts.
    - Allows use of KubernetesPodOperator to run any container workload.
    - Composer enables auto-upgrade and auto-repair to protect against security vulnerabilities.
        - Can perform manual upgrade too.
    - Service Accounts
        - Worker and scheduler nodes and the web server run on different service accounts.
        - Scheduler and workers
            - If service account is not specified during environment creation, default Compute Engine service account is used.
        - Web Server
            - Auto generated during environment creation and derived from webserver domain.

Stackdriver

- Integrates to have a central place to view all Airflow service and workflow logs.
- Can view logs of scheduler and worker emit immediately instead of waiting for Airflow logging module synchronization (due to streaming nature of Stackdriver)

Airflow

- DAGs
    - Composed of Tasks
    - Connects independent tasks and executes in specified sequence.
- Tasks
    - Created by instantiating an Operator class.
    - Logical unit of code.
    - Link Tasks/Operators in your DAG python code.
- Operators
    - Template to wrap and execute a task.
    - BashOperator is used to execute a bash script.
    - PythonOperator is used to execute python code.
    - Specify DAG when instantiating Operator.
    - Sensors

        - Special type of Operator that will keep running until a certain criterion is met.
- Task Instance
    - Represents a specific run of a task is is characterized by a combination of a dag, a task, and a point in time.
    - Has a state (running, success, failed, skipped, up for retry, etc)
- CeleryExecutor
    - Used to execute multiple DAGs in parallel
    - Requires a message broker.
    - SequentialExecutor is used for one DAG at a time.

IAM

- Composer.Admin
- Composer.environmentAndStorageObjectAdmin
- Composer.environmentAndStorageObjectViewer
- Composer.user
- Composer.worker – service accounts

Machine Learning

Basics of Machine Learning

Types of Problems:

- Classification
- Regression
- Clustering
- Rule Extraction

Supervised Learning

- Labels associated with the training data are used to correct the algorithm.

Unsupervised Learning

- The model has to be set up right to learn the structure in the data.

Representation Learning Algorithms

- Feature learning. Algorithm identifies important features on its own.

Deep Learning

- Algorithms that learn what features matter.
- Neural Networks
    - Most common class of deep learning algorithms.
    - Used to build representation learning systems.
    - Composed of neurons (binary classifiers)
    - Wide
        - Better for memorization
    - Deep
        - Better for generalization

Neurons

- Apply 2 functions on inputs.
- Best values of W and b found by using cost function, optimizer, and training data.
- Back Propagation

Linear (affine) transformation

- Like linear regression.
- X1 * W1 + b
    - W = Weights
        - Shape of W
            - First dimension is equal to number of dimensions of feature vector.
            - Second dimension is equal to the number of params required to be tuned. (same goes for b)
    - B = Bias
        - Determined during training process.

Activation Function

- Helps to model non linear functions. (Logistic regression)

- Introduces non-linearity into the network.
- Ex.
    - ReLu (Rectified Linear Unit)
        - Max(Wx + b, 0)
    - SoftMax
        - Multi-class logistic regression
        - Layer right before output.
            - Must have same number of nodes as the output layer.
        - Use candidate sampling for large # of output classes.
    - Sigmoid
        - Binary classification logistic regression

Failure Cases for Backpropagation

Vanishing Gradients

- Gradients for lower layers (closer to input) can become very small.
- Leads to very slow training, if at all.
- ReLu as an activation function can help prevent vanishing gradients.

Exploding Gradients

- Weights in a network are very large, the gradients for the lower layers involve products of many large terms.
- Gradients get too large to converge.
- Batch normalization and/or lowering learning rate can prevent this.

Dead ReLU Units

- Weighted sum of a ReLU falls below 0, ReLU can get stuck.
- Outputs 0 activation, contributing nothing to network's output, and gradients can no longer flow through it during backpropogation.
- Lowering learning rate can help keep ReLU from dying.

Modeling Linear Regression

- 1 neuron with just an affine transformation.
- Y = Ax + b

- Minimize Least Square Error

Optimizers for Best Fit

- Method of Moments
- Method of Least Squares
- Maximum likelihood Estimator

Reducing Loss

- Convergence: When loss stops changing or at least changes extremely slowly.
- Gradient is a vector.
- Learning rate is a scalar.
- Gradient is multiplied by the learning rate.

Hyperparameters

- Configuration settings used to tune hot the model is trained.
- Steps
    - Total number of training iterations. One step calculates the loss from one batch and uses that value to modify the model's weights once.
- Batch Size
    - Number of examples (chosen at random) for a single step.
    - Total # of trained examples = Batch Size * Steps
- Learning rate

Stochastic Gradient Descent

- Random samples from data set to estimate.
- Uses a batch size of 1 per iteration.
- Works (given enough iterations), but noisy

Mini-Batch Stochastic Gradient Descent

- Compromise between full batch and SGD
- Typically between 10 – 10K examples chosen at random.
- Reduce noise, but still more efficient than full-batch.

Periods

- the # of training examples in each period = batch size * steps / period
- Controls granularity of reporting.
    - If periods = 7 and steps = 70, the loss value will be output every 10 steps.
- Modifying period value does not alter what model learns.

Generalization

- The less complex an ML model, the more likely that a good empirical result is not just due to the peculiarities of the sample.
- Overfitting occurs when a model tries to fit the training data so closely that it does not generalize well to new data.
- Identify Overfitting
    - Loss for the validation set is significantly higher than for the training set. (look at loss curve (loss/iterations))
    - Validation loss eventually increases with iterations.
- If the key assumptions of supervised ML are not met, then we lose important theoretical guarantees on our ability to predict new data.
- 3 Basic Assumptions
    - We draw examples independently and identically at random from the distribution. I.e. examples don't influence each other.
    - The distribution is stationary; that is it does not change within the data set.
    - We draw examples from partitions from the same distribution.

Training, Validation, and Test Sets

- Training set – a subset to train a model.
- Test set – a subset to test the trained model.
    - Must be large enough to yield statistically meaningful results.
    - Is representative of the data set as a whole. i.e. don't pick a test set with different characteristics than the training set.
- Doing many rounds of just using a training and test set might cause implicit fitting to the peculiarities of the specific test set.
    - Use a validation set too!
    - Flow
        - Train model
        - Use model on validation set

        - Update hyperparams
        - Repeat
        - Finally test on test set

Representation

- Process of mapping data to useful features.
- Discrete feature
    - A feature with a finite set of possible values.
    - Categorical feature are an example

One-Hot Encoding

- A sparse vector in which:
    - One element is set to 1
    - All other elements are set to 0
- Commonly used to represent strings or identifiers that have a finite set of possible values.

Feature Engineering

- Process of determining which features might be useful in training a model, and then converting raw data from log files and other sources into said features.
- Sometimes called feature extraction.

Qualities of Good Features

- Avoid rarely used discrete feature values.
    - Should appear more than 5 or so times in a data set.
    - Having many examples with the same discrete value gives the model a chance to see the feature in different settings, and in turn, determine when it's a good predictor for the label.
- Prefer clear and obvious meanings
    - Ex. house_age_years vs. house_age
    - Some cases, noisy data causes unclear values, such as data coming from sources that didn't check for appropriate values.
        - Ex. user_age_years: 277
- Don't mix "magical" values with actual data
    - Ex. quality_rating between 0 and 1.

        - If no value, it is set to -1
        - Create a Boolean feature to indicate if quality rating was defined.
    - Replace "magical" values as follows
        - For a variable that take a finite set of values (discrete variables), add a new value to the set and use it to signify that feature value is missing.
        - For continuous variables, ensure missing values do not affect the model by using the mean value of the feature's data.
- Account for upstream instability
    - Definition of a feature shouldn't change over time.

Cleaning Data

- Scaling feature vectors
    - Converting floating point feature values from their natural range (100 to 900) to a standard range (0 to 1 or -1 to 1)
    - Scaling ~= Normalization
    - If only 1 feature, little to no practical benefit.
    - Multiple features, great benefits
        - Helps gradient descent convere more quickly
        - Helps avoid NaN traps
            - One number in the model becomes a NaN (value exceeds floating point precision limit during training) and due to math operations, every other number in the model also eventually becomes NaN.
        - Helps the model learn appropriate weights for each feature. Without scaling, the model pays too much attention to features having a wider range.
- Handling extreme outliers
    - Log scaling
        - Still leaves a tail on distribution
    - Cap or Clipping
        - Reduce feature values that are greater than a set maximum value down to that maximum value.
        - Also, increasing feature values that are less than a specific minimum value up to that minimum value.
    - Binning (Bucketing)
        - Converting a (usually continuous) feature into multiple binary features called buckets or bins, typically based on a value range.
- Scrubbing
    - Data can be unreliable due to:
        - Omitted values

        - Duplicate examples
        - Bad labels
        - Bad feature values
    - "Fix" by removing them from data set.
    - Omitted and duplicate easy to detect.
    - Detecting bad data in aggregate by using Histograms
    - Stats can also help identifying bad data:
        - Max and Min
        - Mean and Median
        - Standard Deviation
- Follow These Rules:
    - Keep in mind what your data should look like
    - Verify that the data meets these expectations
        - Or that you can explain why it doesn't
    - Double check that the training data agrees with other soruces
        - i.e. dashboards

Feature Crosses

- A synthetic feature formed by crossing (Cartesian product) individual binary features obtained from categorical data or from continuous features via bucketing.
- Helps represent nonlinear relationships.
- Encoding Nonlinearity
- Crossing One-Hot Vectors

Regularization

- Minimize loss + complexity
    - Structural Risk Minimization
    - Penalizes complexity to prevent overfitting
- 2 Common Ways to Think About Model Complexity
    - As a function of the weights of all the features in the model
        - L2 Regularization
        - A feature weight with a high absolute value is more complex than one with a low absolute value.
        - L2 = w1^2 + w2^2 + ... + wn^2
        - Consequences of L2 Regularization
            - Encourages weight values toward 0 (but not exactly 0)

        - Encourages the mean of the weights toward 0, with a normal (bell shaped or Gaussian) distribution.
    - As a function of the total number of features with nonzero weights
- Most developer tune the overall impact of the regularization term by multiplying it by a scalar known as lambda (regularization rate)
    - Minimize(loss function + lambda(regularization function))
    - When choosing a lambda value, the goal is to strike the right balance between simplicity and training-data fit
        - Lambda too high
            - Model will be simple but run the risk of underfitting data.
        - Lambda too low
            - Model will be more complex and run the risk of overfitting data.

Early Stopping

- Ending training before the model reaches convergence (training loss finishes decreasing).
- End model training when loss on a validation dataset starts to increase, that is, when generalization performance worsens.

Sparsity

- Sparse vectors often contain many dimensions.
- Creating a feature cross results in even more dimensions.
- High Dimensionality -> Large Model Size -> Large RAM reqs
- L1 Regularization
    - Penalizes absolute value of weights. (|weight|)
    - Derivative of L1 is a constant, k. (2 * weight for L2)
        - A force that subtracts some constant value from the weight every time.
    - Pushes weights toward 0
        - Efficient for wide models.
        - Reduces # of features -> smaller model size
    - May cause informative features to get a weight of exactly 0:
        - Weakly informative features
        - Strongly informative features on different scales
        - Informative features strongly correlated with other similarly informative features.

Dropout

- Useful for neural networks.
- Randomly dropping out unit activations in a network for a single gradient step.
- 0.0 = No dropout regularization.
- 1.0 = Drop out everything. Model learns nothing.

Logistic Regression

- A model that generates a probability for each possible discrete label value in classification problems by applying a sigmoid function to a linear prediction.
- Often used in binary classification problems, but can also be used in multi-class classification problems (multinomial regression)
- Sigmoid Function
    - Maps logistic or multinomial regression output (log odds) to probabilities, returning a value between 0 and 1.
    - Can serve as an activation function in neural networks.
- Loss and Regularization
    - Loss function is Log Loss
    - Regularization
        - L2 or Early Stopping
- One vs All
    - Classification problem with N possible solutions.
    - A one-vs-all solution consists of N separate binary classifiers.

Classification

- Classification Threshold (Decision Threshold)
    - Determines what the probability output from logistic regression is classified as.

Accuracy

- Number of correct predictions over total number of predictions
- TP + TN / (TP + TN + FP + FN)

Class Imbalanced Dataset

- Labels have significantly different frequencies in a classification problem.
- Accuracy is not enough in this scenario.

Confusion Matrix

- An NxN table that summarizes how successful a classification model's predictions were.
- Useful when calculating precision and recall

Precision

- Identifies the frequency with which the model was correct when predicting the positive class.
- TP/ (TP + FP)
- i.e. how many predicted cats are actually cats
- Raising classification threshold reduces FP, thus improving precision.

Recall

- Out of all the possible positive labels, how many did the model correctly identify.
- TP / (TP + FN)
- i.e. number of predicted cats out of all cats
- Raising classification threshold will cause # of TP to decrease or stay the same and will cause the # of FN to increase or stay the same. Thus recall will either stay constant or decrease.
- Improving precision often reduces recall and vice versa.

ROC Curve

- Receiver Operating Characteristic Curve
- Shows performance of classification model at all classification thresholds.
- TP rate (TP / TP + FN) vs. FP rate (FP / FP + TN)
- Lowering classification threshold increase TP and FP.

AUC

- Area Under the ROC Curve
- Provides an aggregate measure of performance across all possible classification thresholds.
- 0 – worst model
- 1 – best model
- Desirable Because:

    - Scale Invariant
        - Measures how well predictions are ranked, rather than their absolute values.
    - Classification Threshold Invariant
        - Measures the quality of the model's predictions irrespective of what classification threshold is chosen.
- Limitations
    - Scale invariance is not always desirable
        - We may need well calibrates probability outputs and AUC won't tell us that.
    - Classification threshold invariance is not always desirable
        - In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one type of classification error.

Prediction Bias

- = average of predictions – average of labels
- Different than bias, b, in wx + b
- Possible root causes of prediction bias:
    - Incomplete feature set
    - Noisy data set
    - Buggy pipeline
    - Biased training sample
    - Overly strong regularization
- Avoid Calibration Layer as a fix
    - Fixing symptoms rather than cause.
    - Built a more brittle system that you must now keep up to date.
- Examine prediction bias on a bucket of examples

Embeddings

- D-Dimensional Embeddings
    - Assumes something can be explained by d aspects.
- Map items to low-dimensional real vectors in a way that similar items are close to each other.

ML API's

Pre-trained ML API's

- For App Developers

Sight

Vision AI

- Image Recognition/analysis
- Label Detection
  - Extracts info in image across categories
- Text Detection (OCR)
  - Detect and extract text from images
- Safe Search
  - Recognize explicit content
- Landmark Detection
- Logo Detection
- Image Properties
  - Dominant colors, pixel counts
- Crop Hints
  - Crop coordinates of dominant object/face
- Web Detection
  - Find matching web entries
- Object Localizer
  - Returns labels and bounding boxes for detected objects.
- Product Search
  - Uses image and specific region(s) or largest object of interest to return matching items from product set.

AutoML Vision

- Object Detection
  - Bounding box smart multi-object detection, Google Vision API on steroids.
- Edge
  - The IoT version of Vision detection for Edge Devices.
  - Optimized to achieve high accuracy for low latency use cases on memory-constrained devices.

    - Use Edge Connect to securely deploy the AutoML model to IoT devices (such as Edge TPUs, GPUs, and mobile devices) and run predictions locally on the device.

Video Intelligence API

- Has pre-trained models that recognize a vast number of objects, places, and actions in stored and streaming video.
- Labels, shot changes, explicit content, subtitles
- Use cases:
    - Content moderation
    - Recommended content
    - Media archives
    - Contextual advertisements

AutoML Video Intelligence

- Video media tagging.
- Train custom video classification models.
- Ideal for projects that require custom labels which aren't covered by the pre-trained Video Intelligence API.
- Detect shot changes
    - Detect scene changes in a segment or throughout the video.

Language

Natural Language API

- Syntax analysis
- Entity analysis
- Sentiment analysis
- Content classification
- Multi-language

AutoML Natural Language

- Handling things like domain specific sentiment analysis and more.
- Can classifies text using own custom labels.

Translation API

- Detect and translate languages
- Beta:
    - Glossary
    - Batch translations

AutoML Translation

- Upload translated language pairs -> Train -> Evaluate

Conversation

Cloud Speech-to-Text API

- Convert audio to text
- Multi-lingual support
- Understand sentence structure

Cloud Text-to-Speech API

- Convert text to audio
- Multiple languages/voices
- Natural sounding synthesis

Dialogflow Enterprise Edition

- Conversational experiences
- Virtual assistants
- Sentiment Analysis
    - Model chat-oriented conversations and responses, to assist you as you build interactive chatbots.
- Text-to-Speech
    - Chatbots trigger synthesized speech for more natural user interaction.

Cloud AutoML

- Enables developers with limited machine learning expertise to train high-quality models specific to their business needs.
- Relies on transfer learning and neural architecture search technology.

AutoML Tables

- Workflow:
    - Table input
    - Define data schema and labels
    - Analyze input features
    - Train (automatic)
        - Feature engineering
            - Normalize and bucketize numeric features
            - Create one-hot encoding and embeddings for categorical features
            - Perform basic processing for text features
            - Extract date- and time-related features from Timestamp columns.
        - Model selection
            - Parallel model testing
                - Linear
                - Feedforward deep neural network
                - Gradient Boosted Decision Tree
                - AdaNet
                - Ensembles of various model architectures
        - Hyperparameter tuning
    - Evaluate model behavior
    - Deploy
- Structured Data
    - Can use data from BigQuery or GCS (CSV)

AutoML Tables vs BigQuery ML

- BQ
    - More focused on rapid experimentation or iteration with what data to include in the model and want to use simpler model types for this purpose.
        - Can potentially return model in minutes
- AutoML
- Have finalized the data.
- Optimizing for maximizing model quality without needing to manually do feature engineering, model selection, ensembling, and so on.

- Willing to wait longer to attain that model quality.
    - Takes at least an hour to train.
- Have a wide variety of feature inputs (beyond numbers and classes) that would benefit from the additional automated feature engineering that AutoML Tables provides.

Cloud Job Discovery

- More relevant job searches
- Power recruitment, job boards

Basic Steps for Most APIs

- Enable API
- Create API key
- Authenticate with API key
- Encode in base64 (optional)
- Make an API request
- Requests and outputs via JSON

Structured Data

- AutoML Tables
- Cloud Inference API
    - Quickly run large scale correlations over types time series data.
- Recommendations AI (Beta)
- BigQuery ML (beta)

Cost

- Pay per API request per feature
- Feature as in Landmark Detection

How to convert images, video, etc for use with API?

- Can use Cloud Storage URI for GCS stored objects
- Encode in base64 format

How to combine API's for scenarios?

- Search customer service calls and analyze sentiment
    - Speech to Text then Sentiment Analysis with Natural Language

AI Platform

- Can use multiple ML platforms such as TensorFlow, scikit-learn and XGBoost

Workflow

- Source and prepare data
    - Data analysis
        - Join data from multiple sources and rationalize it into one dataset.
        - Visualize and look for trends.
        - Use data centric languages and tools to find patterns in data.
        - Identify features in your data.
        - Clean the data to find any anomalous values caused by errors in data entry or measurement.
    - Data preprocessing
        - Transform valid, clean data into the format that best suits the needs of your model.
        - Examples
            - Normalizing numeric data to a common scale.
            - Applying formatting rules to data. Ex. removing HTML tagging from a text feature.
            - Reducing data redundancy through simplification. Ex. converting a text feature to a bag of words representation.
            - Representing text numerically. Ex. assigning values to each possible value in a categorical feature (or 1 hot).
            - Assigning key values to data instances.
    - Develop model
    - Train an ML model on your data
        - Benefits of Training Locally
            - Quick iteration
            - No charge for cloud resources
    - Deploy trained model
        - Upload to GCS bucket
        - Create a model resource in AI Platform specifying GCS path

        - Scenario: Maximize speed and minimize cost of model prediction and deployment:
            - Export trained model to a SavedModel format.
            - Deploy and run on Cloud ML Engine.
    - Send prediction requests to your model
        - Online
        - Batch
    - Monitor predictions on an ongoing basis
        - APIs to examine running jobs.
        - Stackdriver
        - Jobs that can occasionally fail
            - Monitor status of Jobs object for 'failed' jobs states.
    - Manage models and model versions
        - gcloud ai-platform

Preparing Data

- Gather data
- Clean data
    - Clean data by column (attribute)
    - Instances with missing features.
    - Multiple methods of representing a feature.
        - Length measurement in different scale/format
    - Features with values far out of the typical range (outliers)
    - Significant change in data over distances in time, geographic location, or other recognizable characteristics.
    - Incorrect labels or poorly defined labeling criteria.
- Split data
    - Train, Validation, Test
    - Better to randomly sample the subsets from one big dataset than use pre-divided data. Otherwise could be non-uniform => overfitting.
    - Size of datasets: training > validation > test
- Engineer data features
    - Can combine multiple attributes to make one generalizable feature.
        - Address and timestamp => position of sun
    - Can use feature engineering to simplify data.
    - Can get useful features and reduce number of instances in dataset by engineering across instances. I.e. calculate frequency of something.
- Preprocess features

Training Overview

- Upload datasets already split (training, validation) into something AI Platform can read from.
- Sets up resources for your job. One or more virtual machines (training instances)
    - Applying standard machine image for the version of AI Platform your job uses.
    - Loading application package and installing it with pip.
    - Installing any additional packages that you specify as dependencies.
- Distributed Training Structure
    - Running job on a given node => replica
    - Each replica given a single role or task in distributed training:
        - Master
            - Exactly 1 replica
            - Manages others and reports status for the job as a whole.
            - Status of master signals overall job status.
            - Single process job => the sole replica is the master for the job
        - Worker(s)
            - 1 or more replica
            - Do work as designated in job configuration.
        - Parameter Servers
            - 1 or more replicas
            - Coordinate shared model state between the workers.
    - Tiers
        - Scale tiers
            - Number and types of machines you need.
        - CUSTOM tier
            - Allows you to specify the number of Workers and parameter servers.
        - Add these to TrainingInput object in job configuration.
    - Exception
        - The training service runs until your job succeeds or encounters an unrecoverable error.
        - Distributed Case – status of the master replica that signals the overall status.
        - Running a Cloud ML Engine training job locally (gcloud ml-engine local train) is especially useful in the case of testing distributed models.
- Start training
    - Package application with any dependencies required
    - 2 ways

        - Submit by running `gcloud ai-platform jobs submit training`
        - Send a request to the API ar `projects.jobs.create`
            - Need `ml.jobs.create` permission.
    - Job ID
        - Define base name for all jobs associated with a given model and then append a data/time.
    - Job-Dir
        - Save model checkpoints to this GCS path.
        - Useful for VM restarts.
        - Used for job output.
    - CPU, GPU, or TPU?
        - CPUs
            - Quick prototyping that requires maximum flexibility
            - Simple models that do not take long to train
            - Small models with small effective batch sizes
            - Models that are dominated by custom TensorFlow operations written in C++
            - Models that are limited by available I/O or the networking bandwidth of the host system.
        - GPUs
            - Models that are not written in TensorFlow or cannot be written in TensorFlow.
            - Models for which source does not exist or is too onerous to change.
            - Models with a significant number of custom TensorFlow operations that must run at least partially on CPUs
            - Models with TensorFlow ops that are not available on Cloud TPU
            - Medium to large models with larger effective batch sizes
        - TPUs
            - Tensor Processing Units
            - Google's custom developed ASICs used to accelerate machine learning workloads with TensorFlow.
            - Models dominated by matrix computations
            - Models with no custom TensorFlow operations inside the main training loop
            - Models that train for weeks or months
            - Larger and very large models with very large effective batch sizes.
            - Steps

- Authorize Cloud TPU service account name associated with GCP project
- Add service account as a member of your project with role Cloud ML Service Agent.

Hyperparameter Tuning

- –config hptuning_config.yaml
- Hyperparameter: Data that governs the training process itself.
  - DNN
    - Number of layers
    - Number of nodes for each layer
- Usually constant during training.
- How it works:
  - Running multiple trials in a single training job.
  - Each trail is a complete execution of your training application with values for chosen hyperparameters, set within limits specified.
- Tuning optimizes a single target variable (hyperparameter metric)
  - Multiple params per metric.
- Default name is `training/hptuning/metric`
  - Recommended to change to custom name.
  - Must set `hyperparameterMetricTag` value in `HyperparameterSpec` object in job request to match custom name.
- How to actually tune?
  - Define a command line argument in main training module for each tuned hyperparameter.
  - Use value passed in those arguments to set the corresponding hyperparameter in application's TensorFlow code.
- Types
  - Double
  - Integer
  - Categorical
  - Discrete – List of values in ascending order.
- Scaling
  - Recommended for Double and Integer types.
  - Linear, Log, or Reverse Log Scale
- Search Algorithm
  - Unspecified
    - Same behavior as when you don't specify a search algo.

        - Bayesian optimization
    - Grid Search
        - Useful when specifying a number of trials that is more than the number of points in feasible space.
            - In such cases AI Platform default may generate duplicate suggestions.
        - Can't use with any params being Doubles
    - Random Search

Online and Batch Prediction

- Can process one or more instances per request.
- Can serve predictions from a TensorFlow SavedModel.
- Can make requests
    - Legacy Editor
    - Legacy Viewer (Online only)
    - AI Platform Admin or Developer

Online

- Optimized to minimize the latency of serving predictions.
- Predictions returned in the response message.
- Input passed directly as a JSON string.
- Returns as soon as possible.
- Runs on runtime version and in region selected when deploying model.
- Can serve predictions from a custom prediction routine.
- Can generate logs if model is configured to do so. Must specify option when creating model resource.
    - onlinePredictionLogging or –enable-logging (gcloud)
- Use when making requests in responses to application input or in other situations where timely inference is needed.

Batch

- Optimized to handle a high volume of instances in a job and to run more complex models.
- Predictions written to output files in Cloud Storage location that you specify.
    - Can verify predictions before applying them. (sanity check)
- Input data passed directly as one or more UIRs of files in Cloud Storage locations.

- Asynchronous request.
- Can run in any available region, using any runtime version.
    - Should run with defaults for deployed model versions.
- Only Tensorflow supported. (Not XGBoost or scikit)
- Ideal for processing accumulated data when you don't need immediate results.
    - i.e. a periodic job that gets predictions for all data collected since the last job.
- Generates logs that can be viewed on Stackdriver.
- Slow because AI Platform allocates and initializes resources for a batch prediction job when the request is sent.

Prediction Nodes and Resource Allocation

- Think of a Node as a VM

Batch

- Scales nodes to minimize elapsed time job takes.
- Allocates some nodes to handle your job when you start it.
- Scales the number of nodes during the job in an attempt to optimize efficiency.
- Shuts down nodes as soon as job is done.

Online

- Scales nodes to maximize number of requests it can handle without too much latency.
- Allocates some nodes the first time you request predictions after a long pause in requests.
- Scales number of nodes in response to request traffic, adding nodes when traffic increases, removing them when there are fewer requests.
- Keeps at least 1 node ready over a period of several minutes, to handle requests even when there are none to handle.
- Scales down to zero after model version goes several minutes without a prediction request.

Predictions from Undeployed Models

- Batch only
- Specify URI of a GCS locations where the model is stored.
- Explicitly set runtime version in request.

IAM

- Project Roles
  - Ml.admin
  - Ml.developer
  - Ml.viewer
- Model Roles
  - Ml.modelOwner
  - Ml.modelUser

Tensorflow

- OS Machine learning/ Deep learning platform
- Lazy evaluate during build, full evaluate during execution.
- TensorFlow Estimator API
  - High level object oriented API
  - Makes it easy to build models.
  - Specifies predefined architectures, such as linear regressors or neural networks.
- Tf.layers, tf.losses, tf.metrics
  - Reusable libraries for common model components.
- Python TensorFlow
  - Provides Ops, which wrap C++ Kernels
- Can run on CPU, GPU, or TPU
  - Kernels work on more than one platform.
- Feature Engineering
  - Often means converting raw log file entries to tf.Example protocol buffers. See also tf.Transform

Kubeflow

- Helps orchestrate machine learning training pipelines across on-prem and cloud-based resources.
- Can containerize training and serving infrastructure.

Components

- Support for distributed TensorFlow training via the TFJob CRD
  - TFJob is a Kubernetes custom resource used to run TensorFlow training jobs on Kubernetes.

- The ability to serve trained models using TensorFlow Serving
    - Flexible, high performance serving system for machine learning models.
    - Installed by default when deploying Kubeflow.
- A JupyterHub installation with many commonly required libraries and widgets included in the notebook installation, included those needed for TensorFlow Model Analysis (TFMA) and TensorFlow Transform (TFT)
    - JupyterHub – Multi user server for Jupyter notebooks
    - TFMA – Library for evaluating TF models.
        - Uses Apache Beam
    - TFT – Preprocessing/feature engineering
        - Can prevent Training Serving Skew
            - Difference between performance during training and performance during serving. This can be caused by:
                - Discrepancy between how you handle data in the training and serving pipelines.
                - Change in the data between when you train and when you serve.
                - Feedback loop between model and algorithm.
        - Uses Apache Beam
- Kubeflow Pipelines
    - Has UI for managing and tracking experiments, jobs, and runs.
    - An engine for scheduling multi-step ML workflows
    - An SDK for defining and manipulating pipelines and components
    - Notebooks for interacting with the system using the SDK

Exploration/Visualization

Datalab

- Managed Jupyter notebooks
- Great for use with a Dataproc cluster to write PySpark jobs
- Powerful interactive tool to explore, analyze, transform and visualize data and build machine learning models on GCP.
- Supports Python, SQL (BQ), and JavaScript (for BQ user-defined functions)
    - In Cell
        - %%bq query –name queryname
            - SQL underneath

        - %%bq execute -q queryname

- Runs on GCE instance, dedicated VPC and Cloud Source Repository
    - datalab create
    - datalab-network (VPC) is created
    - datalab connect
    - Cloud Source Repository
        - Used for sharing notebook between users

3 Ways to Run:

- Locally
    - Good if only one person using
- Docker on GCE
    - Better
    - Use by multiple people through SSH or CloudShell
    - Uses resources on GCE
- Docker + Gateway
    - Best
    - Uses a gateway and proxy
    - Runs locally

Notebooks

- Can be in Cloud Storage Repository (git repo)
    - Use ungit to commit changes to notebooks

Persistent Disk

- Notebooks can be cloned from GCS to VM persistent disk.
- This clone => workspace => add/remove/modify files
- Notebooks autosave, but you need to commit.

Kernel

- Opening a notebook => Backend kernel process manages session and variables.
- Each notebook has 1 python kernel

- Kernels are single-threaded
- Memory usage is heavy – execution is slow – pick machine type accordingly

APIs and Services

- Enable Compute Engine API

Sharing Notebook Data:

- GCE access based on GCE IAM roles:
    - Must have Compute Instance Admin and Service Account Actor roles to connect to datalab instance.
        - Service Account Actor role deprecated. Use Service Account Token Creator instead.
- Notebook access per user only
- Sharing data performed via shared Cloud Source Repository
- Sharing is at the project level.

Creating Team Notebooks

- 2 Options
    - Team lead creates notebooks for users using –for user option:
        - datalab create [instance] –for-user bob@blah.net
    - Each user creates their own datalab instance/notebook
- Everyone accesses same shared repository of datalab/notebooks
- NO web console option
- Machine Type
    - Standard n1 by default
    - Multi-threading does not work, but can use high memory
    - Custom machine types supported as well.
- Can disable creating shared cloud repository
    - --no-create-repository

Connecting

- SSH tunnels to notebook on port 8081
- datalab connect
    - RSA key is passphrase

- Can configure idle timeouts on the actual webpage

Cost

- Free
- Only pay for GCE resources Datalab runs on and other GCP services you interact with.

Data Studio

- Easy to use data visualization and dashboards.
    - Drag and drop report builder.
- Part of G Suite, not GCP:
    - Uses G Suite access/sharing permissions, not Google Cloud (no IAM)
    - Google account permissions in GCP will determine data source access.
    - Files saved in Google Drive.
- Connect to many Google, Google Cloud, and other services:
    - BQ, Cloud SQL, GCS, Spanner
    - YouTube Analytics, Sheets, AdWords, local upload
        - Local
            - Stored in managed GCS bucket
            - First 2GB free
        - Many third party integrations

Cost

- Free
- BQ access run normal query costs

Basic Process

- Connect to data source
- Visualize data
- Share with others

Creating Charts

- Use combinations of dimensions and metrics
- Create custom fields if needed
- Add date range filters with ease

Caching (most relevant for BQ)

- Options for using cached data performance/costs
- 2 choices
    - Query Cache
        - Remembers query issues by reports components (i.e. charts)
        - When performing same query, pulls from cache.
        - If query cache cannot help, goes to prefetch cache.
        - Cannot be turned off.
    - Prefetch Cache (exam material?)
        - "Smart Cache" – predicts what might be requested
        - If prefetch cache cannot serve data, pulls from live data set
        - Only active for data sources that use owner's credentials for data access
            - If I create table that pulls from BQ table that does not use my credentials for data access, prefetch will be disabled.
        - Can be turned off.
- When to turn caching off:
    - Need to view "fresh data" from rapidly changing data set.

Security

Cloud Identity and Access Management (IAM)

- Provides administrators the ability to manage cloud resources centrally by controlling who can take what action on specific resources.
- Primitive Roles
    - Owner, Editor, Viewer
- Predefined Roles
    - Finer grained access control than primitive roles.
- Custom Roles
    - Create to tailor permissions to the needs of your org when predefined do not meet your needs.

- Can set Cloud IAM policy at any level in resource hierarchy:
    - Organization
    - Project
    - Resource

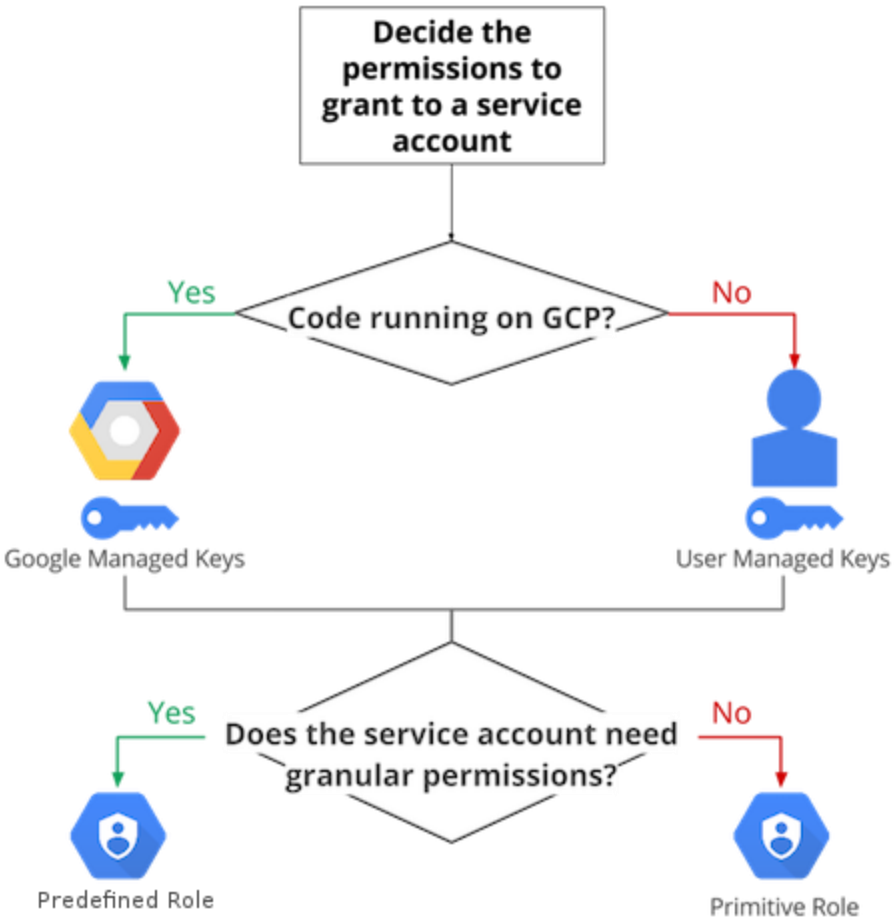
Using IAM Securely

Least Privilege

- Avoid primitive roles, instead grant predefined roles
- Grant primitive in the following cases:
    - GCP service does not provide a predefined role.
    - You want to grant broader permissions for a project. (dev or test environments)
    - Work in a small team where the team members don't need granular permissions.
- Treat each component of your application as a separate trust boundary.
    - Multiple services that require different permissions, create a service account for each service and then grant only the required permissions to each SA.
- Remember that a policy set on a child resource cannot restrict access granted on its parent.
- Grant roles at the smallest scope needed.
- Restrict who can act as service accounts. Users granted the Service Account Actor role for a SA can access all the resources for which the SA has access.
- Restrict who has access to create and manage SAs in your project.
- Granting the Project IAM Admin and Folder IAM Admin predefined roles will allow access to modify Cloud IAM policies without also allowing direct read, write, and administrative access to all resources.
    - Granting Owner role to a member will allow them to access and modify almost all resources, including modifying Cloud IAM policies.

Service Account and Service Account Keys

- Rotate your service account keys using the Cloud IAM service account API.
    - Create a new key, switching applications to use new key, deleting old key.
- Implement processes to manage user-manages service account keys.
- Be careful not to confuse encryption keys with SA keys.
- Do not delete SAs that are in use by running instances.
- Use the display name of a SA to keep track of what they are used for and what permissions they should have.
- Don't check in the SA keys into source code or leave them in the Downloads directory.



Auditing

- Use Cloud Audit Logging logs to regularly audit changes to your Cloud IAM policy.
- Export audit logs to GCS to store for long periods of time.
- Audit who has the ability to change your IAM policies on your project.
- Restrict access to logs using Logging roles.
- Apply the same access policies to the GCP resource that you use to export logs as applied to the logs viewer.
- Use Cloud Audit Logging logs to regularly audit access to SA keys.

Policy Management

- Set organization level Cloud IAM policies to grant access to all projects in your organization.
- Grant roles to a Google group instead of individual users when possible.
    - Easier to add/remove members to/from a group.

- If you need to grant multiple roles to allow a particular task, create a Google group, grant the roles to that group, and then add users to that group.

Billing

- Billing Account Administrator role
  - Allows management of payments and invoices without granting permission to view the project contents.
- Billing Account User role
  - Gives SA permissions to enable billing and therefore permit the SA to enable APIs that require billing to be enabled.
- Billing Account Creator role
  - Allows developers to create new billing accounts and to attach billing accounts to the projects.
- Viewer role
  - Allows developers to view the expenses for the projects they own.
- https://cloud.google.com/docs/enterprise/best-practices-for-enterprise-organizations#identity-and-access-management
- https://cloud.google.com/iam/docs/understanding-roles

Data Loss Prevention API

- Handle sensitive data (especially redaction of PII data)
- Data -> Data Loss Prevention API -> Redacted Data
- Understand encryption techniques (in Cloud Storage Section)

Legal Compliance

- Health Insurance Portability and Accountability Act (HIPAA)
- Children's Online Privacy Protection Act (COPPA)
- FedRAMP
- General Data Protection Regulation (GDPR)

Machine Learning

- https://developers.google.com/machine-learning/glossary/
- https://developers.google.com/machine-learning/crash-course/ml-intro
- https://developers.google.com/machine-learning/guides/rules-of-ml/

Architecture Solutions on GCP:

- https://gcp.solutions/

Cloud Next

- https://cloud.google.com/blog/products/ai-machine-learning/all-ai-announcements-from-google-next19-the-smartest-laundry-list

Apache Beam

- https://cloud.google.com/blog/products/gcp/why-apache-beam-a-google-perspective

Advice (before taking exam)

The documentation on each service is intimidating but it is the most important thing to read.

Compute

Compute Engine

Example

```
# CREATE INSTANCE WITH 4 vCPUs and 5 GB MEMORY

gcloud compute instances create my-vm --custom-cpu 4 --custom-memory 5


# ENABLE PREEMPTIBLE OPTION

gcloud compute instances create my-vm --zone us-central1-b --preemptible
```

Features:

- predefined machine types up to 160 vCPUs and 3.75 TB of memory
- custom machine types

- persistent disk (64 TB in size)
    - ssd or hdd
    - can take snapshots and create new persistent disks
- local ssd for scratch storage (3TB in size)
- debian, centos, coreos, suse, ubuntu, etc images including bring your own
- batch processing (preemptible vms)
- encryption
- per second billing
- committed use discounts

Premptible instances can run a custom shutdown script. Instance is given 30 seconds.

App Engine

Platform as a Service (PaaS).

Features:

- Languages: javascript, ruby, go, .net, java, python, and php
- fully managed
- monitoring through stackdriver
- easy app versioning
- traffic splitting A/B tests and incremental features
- ssl/tls managed
- integrated with GCP services

Kubernetes Engine

Hosted kubernetes engine. Super simple to get started a Kubernetes cluster.

Features:

- integrated with Identify & Access management
- hybrid networking (IP Address for clusters to coexist with private ips)
- HIPAA and PCI DSC 3.1 compliant
- integrated with stackdriver logging and monitoring
- autoscale
- auto upgrade (upgrade nodes)
- auto repair (repair process for nodes)

- resource limits (kubernetes specify resource limits of each container)
- stateful applications (storage backed)
- supports docker image formats
- fully manager with SRE from google
- google managed OS
- private containers (integrates with Google Container Registry)
- builds with (Google Cloud Build)
- can be easily transfered to on premise
- gpu support (easy to run machine learning applications)

GKE In-Prem

Reliable, efficient, and secured way to run kubernetes clusters anywhere. Integration using IAM. Similar features to GKE.

Cloud Functions

Automatically scales, highly available and fault tolerant. No servers to provision.

- file storage events
- events (pub/sub)
- http
- Firebase, and Google Assistant
- stackdriver logging

Languages: python 3.7 and node.

Access and IAM. VPC access to cloud functions. Can not connect vpc to cloud functions. IAM controls on the invoke of the function. `--member` allow you to control which user can invoke the function. IAM check to make sure that appropriate identity.
Serverless containers are coming soon.

Knative

Kubernetes + Serverless addon for GKE.

- serving: event driven, scale to zero, request driven compute model
- build: cloud natie source to container orchestration

- events: universal subscriptin, delivery, and management of events
- add ons: enable knative addons

Sheilded VMs

Shielded VMs protect against:

- verify VM identity
- quickly protect VMs against advanced threats: root kit, etc.
- protect secrests

Only available for certain images.

Containers Security

- patch applied regular on frequent builds and image security reviews
- containers have less code and a smaller attack surface
- isolate processes working with recourses (separate storage from networking)

Storage

Cloud Storage (Object)

Object storage. Unified S3 object storage api. Great for storage of blobs and large files (great bandwidth) not necessarily lowest latency.

Single api for accessing:

- storage
    - multi-regional (you pay for data transfer between regions)
        - reduce latency and increse redundancy
    - regional (higher performance local access and high frequency analytics workloads)
- backup
    - nearline highly durable storage for data accessed less than **once a month**
    - coldline highly durable storage for data accessed less than **once a year**

Object Lifecycle Mangement

- delete files after older than X days
- move files to coldline storage after X days

Requester Pays: you can enable requester of resources for pay for transfer.

You can label a bucket.

You can version all objects.

Encrypt objects with your own encryption keys (locally)

Encrypt objects using Cloud Key Management Storage

Controlling Access:

- `AllUsers:R` makes an object world readable
- `allUsers:objectViewer` makes a bucker or group of objects world readable
- Signed URLs give time-limited read or write access to a specific Cloud Storage resource
- programmatically create signed URLs

Redundancy is

Persistent Disk (Block)

Durable and high performance block storage. SSD and HHD available. Can be attached to any compute engine instance or as google kubernetes storage volumes. Transparently resized and easy backups. Both can be up to 64 TB in size. More expensive per GB than storage. No charge for IO.

- zonal persistent HHD and SSD (efficient reliable block storage)
- regional peristent disk HHD and SSD. Replicated in two zones
- local SSD: high performance transient local block-storage
- cloud storage buckets (mentioned before) affordable object storage

Persistent disk performance is predictable and scales linearly with provisioned capacity until the limits for an instance's provisioned vCPUs are reached.

Persistent disks have built-in redundancy.

Cloud Storage for Firebase (Object)

Free to get started. Uses google cloud storage behind the scenes. Easy way to provide access to files to users based on authentication. Trigger functions to process these files. Clients provides sdks for reliable uploads on spotty connections. Targets mobile.

Cloud Filestore (Filesystem)

Cloud Filestore is a managed file storage service (NAS).

Connects to compute instances and Kubernetes engine instances. Low latency file operations. Performance equivalent to a typical HHD. Can get SSD performance for a premium.

Size must be between 1 TB - 64 TB. Price per gigabyte per hour. About 5x more expensive than object storage. About 2-3X more expensive than Blob storage.

Cloud filestore exists in the zone that you are using.

Is an NFS fileshare.

Drive Enterprise (Google Drive)

$8 per active user/month + $0.04 per GB/month and includes Google Docs, Sheets, and Slides.

Migration

Covers sending files to google.

Data Transfer (Online Transfer)

Use your network to move data to Google Cloud storage

- draga and drop
- gsutil
- json api inject tool (python api for example)

Cloud Storage Transfer Service

use for cloud transfters like AWS -> GCP

- Storage Transfer Service allows you to quickly import online data into Cloud Storage. You can also set up a repeating schedule for transferring data, as well as transfer data within Cloud Storage, from one bucket to another.
    - schedule one time transfer operations or recurring transfer operations
    - deleete existing object in the destination bucket if they dont have a corresping object in the source
    - delete source objects after transfering them
    - schedule periodic synchronization from source to data (with filters)

Transfer Appliance
install storage locally move data and send to google

- two rackable applicances capable of 100 TB + 100 TB
- standalone 500 TB - 1 PB storage to transfer.
- greater than 20 TB it is worth it.
- capable of high upload speeds (> 1GB per second)

- Big Query Data Transfer Service
    - The BigQuery Data Transfer Service automates data movement from Software as a Service (SaaS) applications such as Google Ads and Google Ad Manager on a scheduled, managed basis. Your analytics team can lay the foundation for a data warehouse without writing a single line of code.

data sources
campaign manager, cloud storage, google ad manager, google ads, google play, youtube channel reports, youtube content owner reports.

Databases

Cloud SQL (MYSQL POSTGRESQL)

Transactions.

Fully manged MYSQL and Postgresql service. Susstained usage discount. Data replication between zones in a region.

- Fully managed MySQL Community Edition databases in the cloud.
- Second Generation instances support MySQL 5.6 or 5.7, and provide up to 416 GB of RAM and 10 TB data storage, with the option to automatically increase the storage size as needed.

- First Generation instances support MySQL 5.5 or 5.6, and provide up to 16 GB of RAM and 500 GB data storage.
- Create and manage instances in the Google Cloud Platform Console.
- Instances available in US, EU, or Asia.
- Customer data encrypted on Google's internal networks and in database tables, temporary files, and backups.
- Support for secure external connections with the Cloud SQL Proxy or with the SSL/TLS protocol.
- Support for private IPbeta (private services access).
- Data replication between multiple zones with automatic failover.
- Import and export databases using mysqldump, or import and export CSV files.
- Support for MySQL wire protocol and standard MySQL connectors.
- Automated and on-demand backups, and point-in-time recovery.
- Instance cloning.
- Integration with Stackdriver logging and monitoring.
- ISO/IEC 27001 compliant.

Some statements and features are not supported.

Cloud BigTable (NoSQL similar to Cassandra HBASE)

High throughput and consistent. Sub 10 ms latency. Scale to billions of rows and thousands of columns. can store TB and PB of data. Each row consists of a key. Large amounts of single keyed data with low latency. High read and write throughput. Apache HBase API.

Replication among zones. Key value map.

- key to sort among rows
- column families for combinations of columns

Performance

- row keys should be evenly spread among nodes

How to choose a row key:

- reverse domain names (domain names should be written in reverse) com.google for example.
- string identifiers (do not hash)

- timestamp in row key
- row keys can store multiple things - keep in mind that keys are sorted (lexicographically)

Cloud bigtable

- 10 MB per cell and 100 MB per row max

Cloud Spanner (SQL)

Globally consistent cloud database.

Fully managed.

Relational Semantics requiring explicit key.

Highly available transactions. Globally replicated.

Cloud Datastore (NoSQL)

SQL like query language. ACID transactions. Fully managed.

Eventually consistent. Not for relational data but storing objects.

- Atomic transactions. Cloud Datastore can execute a set of operations where either all succeed, or none occur.
- High availability of reads and writes. Cloud Datastore runs in Google data centers, which use redundancy to minimize impact from points of failure.
- Massive scalability with high performance. Cloud Datastore uses a distributed architecture to automatically manage scaling. Cloud Datastore uses a mix of indexes and query constraints so your queries scale with the size of your result set, not the size of your data set.
- Flexible storage and querying of data. Cloud Datastore maps naturally to object-oriented and scripting languages, and is exposed to applications through multiple clients. It also provides a SQL-like query language.
- Balance of strong and eventual consistency. Cloud Datastore ensures that entity lookups by key and ancestor queries always receive strongly consistent data. All other queries are eventually consistent. The consistency models allow your application to deliver a great user experience while handling large amounts of data and users.
- Encryption at rest. Cloud Datastore automatically encrypts all data before it is written to disk and automatically decrypts the data when read by an authorized user. For more information, see Server-Side Encryption.

- Fully managed with no planned downtime. Google handles the administration of the Cloud Datastore service so you can focus on your application. Your application can still use Cloud Datastore when the service receives a planned upgrade.

Cloud Memorystore (redis)

- redis as a service
- high availablility, failover, patching, and monitoring
- sub millisecond latency and throughput
- can support up to 300 GB instances with 12 Gbps throughput
- fully compatible with redis protocol

Ideal for low latency data that must be shared between workers. Failover is separate zone. Application must be tollerant of failed writes.

Cloud Firestore (Datastore like)

NoSQL database built for global apps. Compatible with Datastore API. Automatic multi region replication. ACID transactions. Query engine. Integrated with firebase services.

Cloud firebase Realtime Database

Real time syncing of JSON data. Can collaborate across devices with ease.

Could this be used for jupyter notebooks? Probably not due to restrictions... cant see exactly what text has changed.

Networking

Virtual Private Cloud (VPC)

A private space within google cloud platform. A single VPC can span multiple regions without communicating accress public Internet. Can allow for single connection points between VPC and on-premise resources. VPC can be applied at the organization level outside of projects. No shutdown or downtime when adding IP scape and subnets.

Get private access to google services such as storage big data, etc. without having to give a public ip.

VPC can have:

- firewall rules
- routes (how to send traffic)
- forwarding rules
- ip addresses

Ways to connect:

- vpn (using ipsec)
- interconnect

Google Cloud Load Balancing

Cloud load balances can be divided up as follows:

- global vs. regional load balancing
- external vs internal
- traffic type

Global:

- http(s) load balancing
- ssl proxy
- tcp proxy

Global has a single anycast address, health checks, cookie-based affinity, autoscaling, and monitoing.

Regional:

- internal TCP/UDP load balancing
- network tcp/udp load balancing

Single IP address per region.

Google Cloud Armor

Protect against DDoS attacks. Permit deny/allow based on IP.

Cloud CDN

Low-latency, low-cost content delivery using Google global network. Recently ranked the fastest cdn. 90 cache sites. Always close to users. Cloud CDN comes with SSL/TLS.

- anycast (single IP address)
- http/2 support these push abilities
- https

invalidation
take down cached content in minutes

- logging with stackdriver
- serve content from compute engine and cloud storage buckets. Can mix and match.

Cloud Interconnect

Connect directly to google.

- interconnect (Direct access with SLA)
  - dedicated interconnect
  - partner interconnect
  - ipsec VPN
- peering (google's public ips only) no physical connection to google
  - direct peering
  - carries peering

Cloud DNS

Scalable, reliable, and amanaged DNS syetm. Guarantees 100% availability. Can create millions of DNS records. Managed through API.

Network Service Tiers (no studied)

Network Telemetry

Allows for monitoring on network traffic patterns.

Management Tools

- Stackdriver
  - monitoring
  - logging
  - error reporting with triggers
  - trace

Cloud Console
web interface for working with google cloud recources

Cloud shell
command line management for web browser

Cloud mobile app
available on android and OSX

- Seperate billing accounts for managing paying for projects
- Cloud Deployment Manager for managing google cloud infrastructure
- Cloud apis for all GCP services

API Platform and Ecosystems

Apigee API Platform

Many many features around apis. Features: design, secure, deploy, monitor, and scale apis. Enforce api policies, quota management, trasformation, autorization , and access control.

- create api proxies from Open API specifications and deploy in the cloud.
- protect apis. oauth 2.0, saml, tls, and protection from traffic spikes
  - dynamic routing, caching, and rate limiting policies
- publish apis to a developer portar for developers to be able to explore
- measure performance and usage integrating with stackdriver.

Free trial, then $500 quickstart and larger ones later. Monetization.

Healthcare, Banking, Sense (protect from attacks).

API Monetization

Tools for creating billing reports for users. Flexible report models etc. This is through apigee

$300/month to start. Engagement, operational metrics, business metrics.

Cloud Endpoints

Google Cloud Endpoints allows a shared backend. Cloud endpoints annotations. Will generate client libraries for the different languages.

Nginx based proxy. Open API specification and provides insight with stackdriver, monitoring, trace, and logging.

Control who has acess to your API and validate every call with JSON web tockens and google api keys. Integration with Firebase Authentication and Auth0.

Less than 1ms per call.

Generate API keys in GCP console and validate on every API call.

Developer Portal

Have dashboards and places for developers to easily test the API.

Developer Tools

Cloud SDK

cli for GCP products

- `gcloud` manages authentication, local configuration, developer workflow, and interactions with cloud platforms apis

bq

big query through the command line

kubectl
management of kubernetes

gsutil
command line access to manage cloud storage buckets and objects

- powershell tools (which I will never use)

Container Registry

More than a private docker repository.

- secure private registry
- automatically build and push images to private registry
- vulnerability scanning
- multiple registries
- fast and highly available
- prevent deployment of images that are risky and not allowed

Allows:

- project names images
- domain named images
- access control
- authentication `gcloud docker -a` to use authenticate with gcp credentials
- `mirror.gcr.io` mirrors docker registry
- notifications when container images change
- vulnerability monitoring

Cloud Build

Cloud Build lets you commit to deploy in containers quickly.

- stages: build, test, deploy
- push changes from Github, cloud source repositories, or bitbucket
- run builds on high cpu vms
- automate deployment
- custom workflows
- privacy
- integrations with GKE, app engine, cloud functions, and firebase
- spinnaker supported on all clouds - 120 build minutes per day.

YAML file to specify build.

Cloud Source Repositories

- unlimited private git repositories
- build in continuous integration (automatic build and testing)
- fast code search
- pub/sub notification
- permissions
- logging with stackdriver
- prevent security keys from being commit.

Cloud Scheduler

CRON job scheduler. Batch and big data jobs, cloud infrastructure operations. Automate everything retries, manage all automated tasks in one place.

Serverless scheduling. Cloud scheduler. HTTP or pub/sub tasks up to 1 minutes intervals.

Data Analytics

Big Query

Free up to 1 TB of data analyzed each month and 10 GB stored.

- Data Warehouses
- Business Intelegence
- Big Query
- Big Query ML
    - Adding labels in SQL query and training model in SQL
    - linear regresion, classification logistic regressin, roc curve, model weight inspection.
    - feature distribution analysis
    - integrations with Data Studio, Looker, and Taeblo

Accessible through REST api and client libraries, including command line tool.

Data is stored in Capacitor columnar data format and offers the standard database conceps of tables, paritions, columns, and rows.

Data in Big Query Can be loaded in:

- batch loads
    - cloud storage
    - other google services (example google ad manager)
    - readable data source
    - streaming inserts
    - dml (data manipulation language) statements
    - google big query IO transformation
- streaming

Saving money on costs:

- avoid `SELECT *`
- use summary routines to only show a few rows
- use `--dry-run` command it will give you price of query
- no charge for regular loading of data. Streaming does cost money.

Cloud Dataflow

Fully managed service for transforming and reacting to data.

automated resource managerment

Cloud Dataflow automates provisioning and management of processing resources to minimize latency and maximize utilization; no more spinning up instances by hand or reserving them.

dynamic work rebalancing

automated and optimized work partitioning dynamic lagging work.

- reliable & consistent exactly-once processing
- horizontal auto scaling
- apache beam sdk

Realtime processing of incoming data.

- cleaning up data

- triggering events
- writing data to destinations SQL, bigquery, etc.

Regional Endpoints

Cloud Dataproc

Managed apache hadoop and apache spark instances.

Scalable and allow for preemptible instances.

ETL pipeline (Extract Transform Load)

Cloud Datalab

Jupyter notebooks + magiks that working google cloud.

Big query integration

Gcloud console commands

Cloud Pub/Sub

simple reliable, scalable foundation for analytics and event driven computing systems.

Can trigger events such as Cloud Dataflow.

Features:

- at least once delivery
- exactly once processing
- no provisioning
- integration with cloud storage, gmail, cloud functions etc.
- open apis and client libraries in many languages
- globally trigger events
- pub/sub is hipaa compliant.

Details:

message

the data that moves through the serivce

topic
a named entity that represetnds a feed of messages

subscription
an entity interested in receiving mesages ona particular topic

publisher
create messages and sends (published) them to a specific topic

subscriber (consumer)
recieves messages on a specified subscription

Performance(scalability):

Cloud Composer

Managed Apache airflow. (differences)

- client tooling and integrated experience with google cloud
- security IAM and audit logging with google cloud
- stackdriver intergration
- streamlined airflow runtime and environment configuration such as plugin support
- simplified DAG (workflow) maangement
- python pypi package management

Core support for:

- dataflow
- bigquery
- storage operators
- spanner
- sql
- support in many other clouds
- workflow orchestration solution

DAGS every workflow is defined as a DAG. Series of tasks to be done.

Task a set task within a workflow that you want to do.

Operators.

Builtin support for services outside GCP: http, sftp, bash , python, AWS, Azure, Databricks, JIRA, Qubole, Slack, Hive, Mongo, MySQL, Oracle, Vertica.

Kubernetes PodOperator.

Integrates with `gcloud composer`. Cloud SQL is used to store the Airflow metadata. App Engine for serving the web service. Cloud storage is used for storing python plugins and dags etc. All running inside of GKE. Stackdriver is used for collecting all logs.

Genomics

Process in parallel.

- fully integrated with GCP products
- secured with hippa compliant
- real time processing

Google analytics 360 Studio

Enterprise analytics (not going to dive deaply into).

- understand customer decision better
- used google adertising system
- adwords
- doubleclick
- allows easy collaboration
- tag manager
    - manage analytics tags
- optimize
    - personalized messages to audience (text change to website measure best variant)
- google audience
    - recognize who your most valuable audiences are
- attriubtion
    - shows how your messages are converged
- data studio
    - data import and data visualization from big query
    - custom reports
- surveys

    - accurate surveys to send out
    - brand awareness

Google Data Studio

Unite data in one place: spreadsheets, analytics, google ads, big query all combined
Explore the data

- connectors: many many services including community ones
- transformation
- you can cache queries from big query.
    - big query can cache results temporarily or using a new table

Firebase Performance Monitoring

Monitoring loading experience for users.

- wide variety of locations
- wide variaty of
- variaty, location, device, etc.

SDK in app you get performance metrics of your app as seen from the users.

AI and Machine Learning

AI Hub

A collection of end-to-end AI pipelines and out of the box algorithms for solving specific machine learning problems.

Prebuilt solutions. Very much a pre alpha product.

Cloud AutoML

Makes it approachable even if you have minimal experience.

Products:

- natrual language

- translation
- vision

Use case is that you would like to specially train your model to detect features that are more specific than the ones google provides.

Limited experience to train and make predictions. Full rest api and scalably train for labeling features.

Cloud TPU

Much faster for machine learning computations and numerical algorithms.

Cloud Machine Learning Engine

Focus on models not operations.

- multiple frameworks built in
  - scikit-klearn, xgboost, keas, tensorflow
- automate hyperparameter search
- automate resource provisioning
- train with small models for development and easily scale up
- send computations to the cloud
- cloud ml engine works with cloud dataflow

Manage model version.

gcloud commands work with it.

Cloud Talent Solution

Machine learning to solve the job search

Dialog Flow

- conversational interfaces
  - chatbots for example
  - text to speech
  - 20+ languages supported

- user sentiment analysis
- spelling correction

Cloud Natural Language

Google Cloud Natural Language reveals the structure and meaning of text both through powerful pretrained machine learning models in an easy to use REST API and through custom models that are easy to build with AutoML Natural LanguageBeta. Learn more about Cloud AutoML.

You can use Cloud Natural Language to extract information about people, places, events, and much more mentioned in text documents, news articles, or blog posts. You can use it to understand sentiment about your product on social media or parse intent from customer conversations happening in a call center or a messaging app. You can analyze text uploaded in your request or integrate with your document storage on Google Cloud Storage.

Cloud Speech to Text

Google Cloud Speech-to-Text enables developers to convert audio to text by applying powerful neural network models in an easy-to-use API. The API recognizes 120 languages and variants to support your global user base. You can enable voice command-and-control, transcribe audio from call centers, and more. It can process real-time streaming or prerecorded audio, using Google's machine learning technology.

Cloud Text-to-Speech

Google Cloud Text-to-Speech enables developers to synthesize natural-sounding speech with 30 voices, available in multiple languages and variants. It applies DeepMind's groundbreaking research in WaveNet and Google's powerful neural networks to deliver high fidelity audio. With this easy-to-use API, you can create lifelike interactions with your users, across many applications and devices.

Cloud Translation

Cloud Translation offers both an API that uses pretrained models and the ability to build custom models specific to your needs, using AutoML Translation.

The Translation API provides a simple programmatic interface for translating an arbitrary string into any supported language using state-of-the-art Neural Machine Translation. It is highly responsive, so websites and applications can integrate with Translation API for fast, dynamic translation of source text from the source language to a target language (such as French to English). Language detection is also available in cases where the source language is unknown. The underlying technology is updated constantly to include improvements from Google research teams, which results in better translations and new languages and language pairs.

Cloud Vision

Cloud Vision offers both pretrained models via an API and the ability to build custom models using AutoML Vision to provide flexibility depending on your use case.

Cloud Vision API enables developers to understand the content of an image by encapsulating powerful machine learning models in an easy-to-use REST API. It quickly classifies images into thousands of categories (such as, “sailboat”), detects individual objects and faces within images, and reads printed words contained within images. You can build metadata on your image catalog, moderate offensive content, or enable new marketing scenarios through image sentiment analysis.

Cloud Inference API

Quickly run large-scale correlations over typed time-series datasets.

Calculate correlations between data that you are getting from sensors etc. For example using big table.

Firebase Predictions

Group users based on predictive behavior.

Cloud Deep Learning VM Image

Preconfigured images for deep learning application.

Have all the interesting frameworks preinstalled.

Security

Learn if necessary.

Cloud IAM

Identity. Access is granted to `members`. Memebers can be of several types:

- google account (any google account not necissarily `@gmail.com`.
- service account (created for individual applications)
- google group `group@domain.com` (email address that contains several google users) cannot establish identity
- gsuite domain `domain.com` (cannot establish identity)
- allAuthenticatedUsers (any signed in google account)
- allUsers anyone on the web

Resource:

- you can grant access to `<service>.<resource>` resources

Permissions:

- you can grant access based on `<service>.<resource>.<verb>`.

Roles are collections of permissions. Three kinds of roles in Cloud IAM.

- primitive roles: Owner, Editor, Viewer
- predefined roles: finer access than primitive roles. `roles/pubsub.publisher` provides access to only publish messages to a cloud pub/sub topic.
- custom roles: custom roles specific to the organization.

Cloud IAM policies bind `member -> roles`.
Resource Hierarchy

You can set a Cloud IAM policy at any level in the resource hierarchy: the organization level, the folder level, the project level, or the resource level. Resources inherit the policies of the parent resource. If you set a policy at the organization level, it is automatically inherited by all its children projects, and if you set a policy at the project level, it's inherited by all its child resources. The effective policy for a resource is the union of the policy set at that resource and the policy inherited from higher up in the hierarchy.

Best Practices:

- Mirror your IAM policy hierarchy structure to your organization structure.
- Use the security principle of least privilege to grant IAM roles, that is, only give the least amount of access necessary to your resources.
- grant roles to groups when possible
- grant roles to smallest scope needed
- billing roles for administration looking over
- prefer predefined roles (not primitive)
- owner allows modifying IAM policy so grant carefully
- cloud audit logs to monitor changes to IAM policy

Service Accounts:

- belong to an application or virtual machine instead of user
- Each service account is associated with a key pair, which is managed by Google Cloud Platform (GCP). It is used for service-to-service authentication within GCP. These keys are rotated automatically by Google, and are used for signing for a maximum of two weeks.
- You should only grant the service account the minimum set of permissions required to achieve their goal.
- Compute Engine instances need to run as service accounts
- establish naming convention for service account