



## Choose a Section

[Introduction](#)[Case Studies](#)[Foundational Concepts](#)[Cloud SQL](#)[Cloud Datastore](#)[Cloud Bigtable](#)[Cloud Spanner](#)[Real Time Messaging with Cloud Pub/Sub](#)[Data Pipelines with Cloud Dataflow](#)[Cloud Dataproc](#)[BigQuery](#)[Machine Learning Concepts](#)[AI Platform](#)[Pre-trained ML API's](#)[Cloud Datalab](#)[Cloud Dataprep](#)[Data Studio](#)[Cloud Composer](#)[Additional Study Resources](#)

[Return to Table of Contents](#)

## Choose a Lesson

[What is a Data Engineer?](#)[Exam and Course Overview](#)

## *What is a Data Engineer?*

### Google's definition:

A Professional Data Engineer enables data-driven decision making by collecting, transforming, and visualizing data. The Data Engineer designs, builds, maintains, and troubleshoots data processing systems with a particular emphasis on the security, reliability, fault-tolerance, scalability, fidelity, and efficiency of such systems.

The Data Engineer also analyzes data to gain insight into business outcomes, builds statistical models to support decision-making, and creates machine learning models to automate and simplify key business processes.

### What does this include?

- Build data structures and databases:
  - Cloud SQL, Bigtable
- Design data processing systems:
  - Dataproc, Pub/Sub, Dataflow
- Analyze data and enable machine learning:
  - BigQuery, Tensorflow, Cloud ML Engine, ML API's
- Match business requirements with best practices
- Visualize data ("make it look pretty"):
  - Data Studio
- Make it secure and reliable

### Super-simple definition:

Collect, store, manage, transform, and present data to make it useful.

[Return to Table of Contents](#)

## Choose a Lesson

[What is a Data Engineer?](#)[Exam and Course Overview](#)

## *Exam and Course Overview*

[Next](#)

### **Exam format:**

- **50 questions**
- **120 minutes (2 hours)**
- **Case study + individual questions**
- **Mixture of high level, conceptual, and detailed questions:**
  - **How to convert from HDFS to GCS**
  - **Proper Bigtable schema**
- **Compared to the architect exam it is more focused and more detailed:**
  - **Architect exam = 'Mile wide/inch deep'**
  - **Data Engineer exam = 'Half mile wide, 3 inches deep'**

### **Course Focus:**

- **Very broad range of topics**
- **Depth will roughly match exam:**
  - **Plus hands-on examples**

[Return to Table of Contents](#)

## Choose a Lesson

[What is a Data Engineer?](#)[Exam and Course Overview](#)

## *Exam and Course Overview*

[Previous](#)

### **Exam topics:**

- Building data representations
- Data pipelines
- Data processing infrastructure
- Database options - differences between each
- Schema/queries
- Analyzing data
- Machine learning
- Working with business users/requirements
- Data cleansing
- Visualizing data
- Security
- Monitoring pipelines

### **Google Cloud services covered:**

- Cloud Storage
- Compute Engine
- Dataproc
- Bigtable
- Datastore
- Cloud SQL
- Cloud Spanner
- BigQuery
- Tensorflow
- ML Engine
- Managed ML API's - Translate, Speech, Vision, etc.
- Pub/Sub
- Dataflow
- Data Studio
- Dataprep
- Datalab

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

## Flowlogistic Case Study

[Next](#)

### Link:

<https://cloud.google.com/certification/guides/data-engineer/casestudy-flowlogistic>

### Main themes:

- Transition existing infrastructure to cloud
- Reproduce existing workload ("lift and shift"):
  - First step into cloud transition

### Primary cloud objectives:

- Use proprietary inventory-tracking system:
  - Many IoT devices - high amount of real-time (streaming) data
  - Apache Kafka stack unable to handle data ingest volume
  - Interact with both SQL and NoSQL databases
  - Map to Pub/Sub - Dataflow:
    - Global, scalable
- Hadoop analytics in the cloud:
  - Dataproc - managed Hadoop
  - Different data types
  - Apply analytics/machine learning

### Other technical considerations:

- Emphasis on data ingest:
  - Streaming and batch
- Migrate existing workload to managed services:
  - SQL - Cloud SQL:
    - Cloud Spanner if over 10TB and global availability needed
  - Cassandra - NoSQL (wide-column store) - Bigtable
  - Kafka - Pub/Sub, Dataflow, BigQuery
- Store data in a 'data lake':
  - Further transition once in the cloud
  - Storage = Cloud Storage, Bigtable, BigQuery
  - Migrate from Hadoop File System (HDFS)

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

## Flowlogistic Case Study

[Previous](#)[Next](#)

### Inventory Tracking Data Flow



Tracking Devices



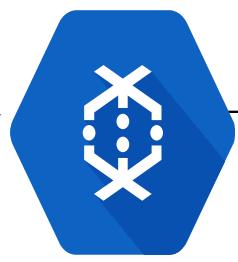
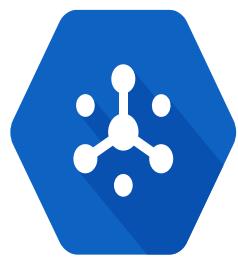
Tracking Devices



Tracking Devices



Tracking Devices



Inventory tracking

Metadata - tracking messages

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Pub/Sub is used for streaming (real-time) data ingest. Allows asynchronous (many-to-many) messaging via published and subscribed messages.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Cloud Dataflow is a data processing pipeline, transforming both stream and batch data.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Cloud SQL is a fully managed MySQL and PostgreSQL database. It is a perfect transition step for migrating SQL workloads.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X  
Cloud Bigtable is a managed, massively scalable non-relational/NoSQL database based on HBase.

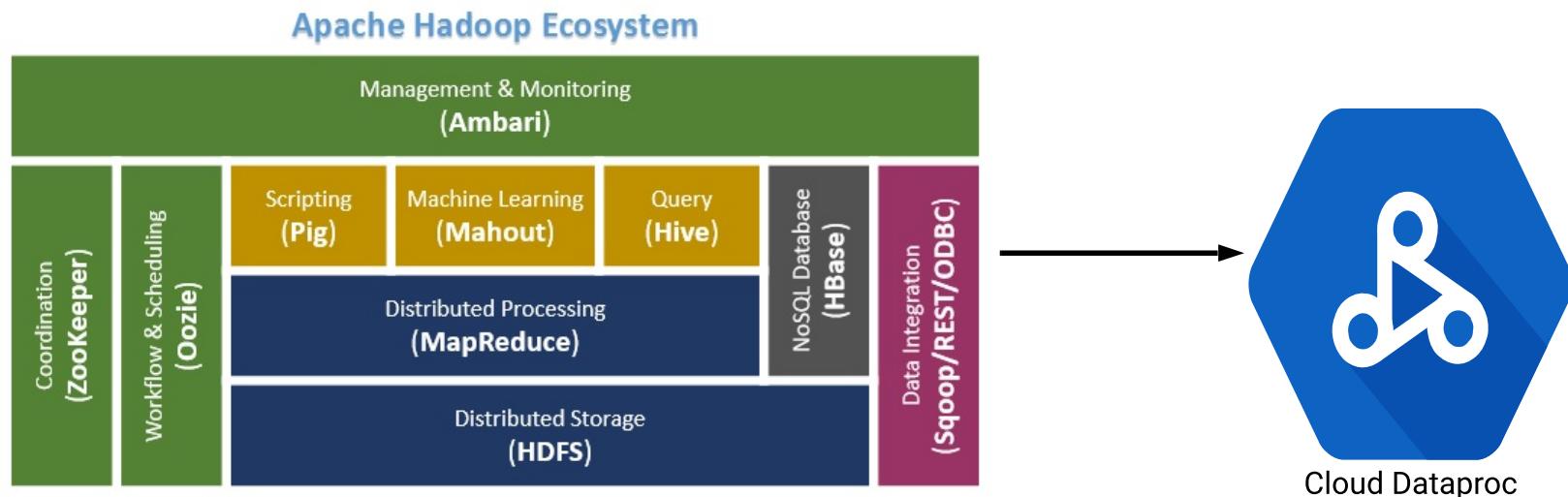
[Return to Table of Contents](#)

## Flowlogistic Case Study

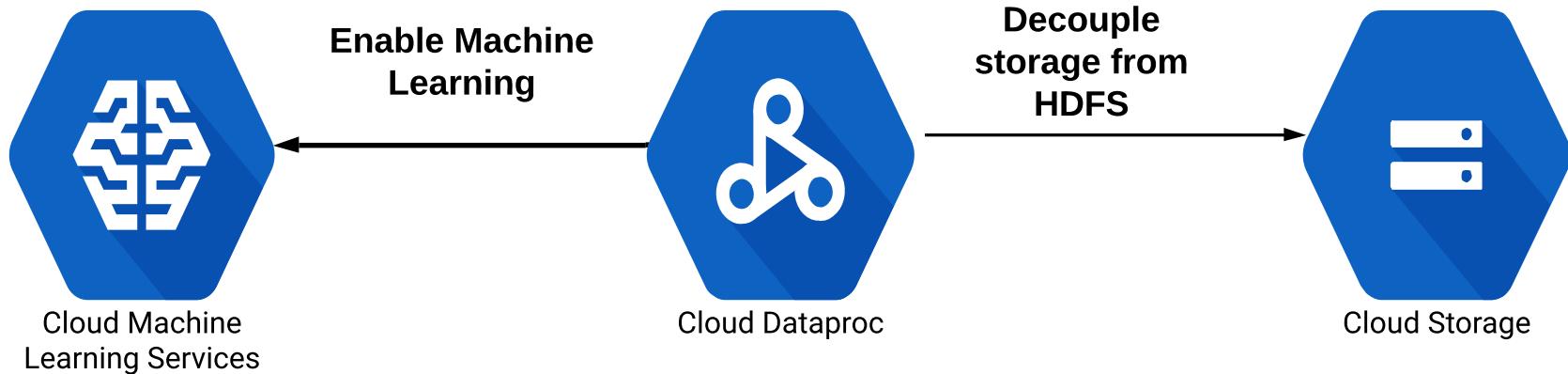
### Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)[Previous](#)

### Phase 1: Initial migration of existing Hadoop analytics



### Phase 2: Integrate other Google Cloud Services



[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

Cloud Dataproc offers fully managed Apache, Hadoop, and Spark cluster management. It integrates easily with other GCP services.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Managed machine learning service  
for predictive analytics.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Decoupling storage from the Dataproc cluster allows for destroying the cluster when the job is complete as well as widely available, high-performance storage.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

## **Case Study Overview**

- Exam has 2 possible case studies
- Exam case studies available from Google's training site:  
<https://cloud.google.com/certification/guides/data-engineer>
- Different 'themes' to each case study = insight to possible exam questions
- Very good idea to study case studies in advance!
- Case study format:
  - Company Overview
  - Company Background
  - Solution Concept – current goal
  - Existing Technical Environment – where they are now
  - Requirements – boundaries and measures of success
  - C-level statements – what management cares about

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

## *MJTelco Case Study*

[Next](#)

### Link:

<https://cloud.google.com/certification/guides/data-engineer/casestudy-mjtelco>

### Main themes:

- No legacy infrastructure - fresh approach
- Global data ingest

### Primary Cloud Objectives:

- Accept massive data ingest and processing on a global scale:
  - Need no-ops environment
  - Cloud Pub/Sub accepts input from many hosts, globally
- Use machine learning to improve their topology models

### Other technical considerations:

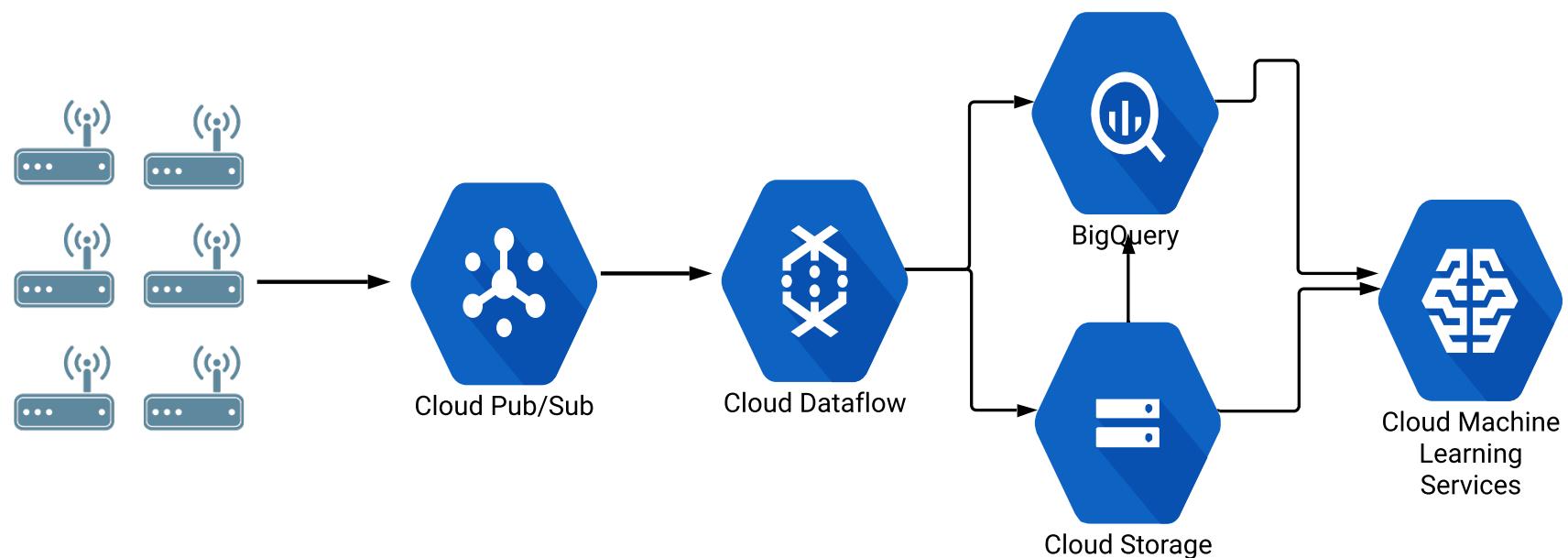
- Isolated environments:
  - Use separate projects
- Granting access to data:
  - Use IAM roles
- Analyze up to 2 years worth of telemetry data:
  - Store in Cloud Storage or BigQuery

[Return to Table of Contents](#)

### Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)[Previous](#)

### Data Flow Model



[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Cloud Storage provides globally available, long-term, high-performance storage for all data types.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

Cloud Dataflow is a data processing pipeline, transforming both stream and batch data.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X

BigQuery is a no-ops data warehouse used for massively scalable analytics.

[Return to Table of Contents](#)

## Choose a Lesson

[Case Study Overview](#)[Flowlogistic](#)[MJTelco](#)

X  
Managed machine learning service  
for predictive analytics.

[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)

# The Data Dossier

## *Data Lifecycle*

[Next](#)

- Think of data as a tangible object to be collected, stored, and processed
- Lifecycle from initial collection to final visualization
- Needs to be familiar with the lifecycle steps, what GCP services are associated with each step, and how they connect together
- Data Lifecycle steps:
  - Ingest - Pull in the raw data:
    - Streaming/real-time data from devices
    - On-premises batch data
    - Application logs
    - Mobile-app user events and analytics
  - Store - data needs to be stored in a format and location that is both reliable and accessible
  - Process and analyze - Where the magic happens. Transform data from raw format to actionable information
  - Explore and visualize - "Make it look pretty"
    - The final stage is to convert the results of the analysis into a format that is easy to draw insights from and to share with colleagues and peers

[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)

## Data Lifecycle

[Previous](#)[Next](#)

## Data Lifecycle and Associated Services

Ingest	Store	Process & Analyze	Explore & Visualize
 App Engine	 Cloud Storage	 Cloud Dataflow	 Cloud Datalab
 Compute Engine	 Cloud SQL	 Cloud Dataproc	 Google Data Studio
 Kubernetes Engine	 Cloud Datastore	 BigQuery	 Google Sheets
 Cloud Pub/Sub	 Cloud Bigtable	 Cloud ML	
 Stackdriver Logging	 BigQuery	 Cloud Vision API	
 Cloud Transfer Service	 Cloud Storage for Firebase	 Cloud Speech API	
 Transfer Appliance	 Cloud Firestore	 Translate API	
	 Cloud Spanner	 Cloud Natural Language API	
		 Cloud Dataprep	
		 Cloud Video Intelligence API	

[Return to Table of Contents](#)**Choose a Lesson**[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)[Previous](#)[Next](#)

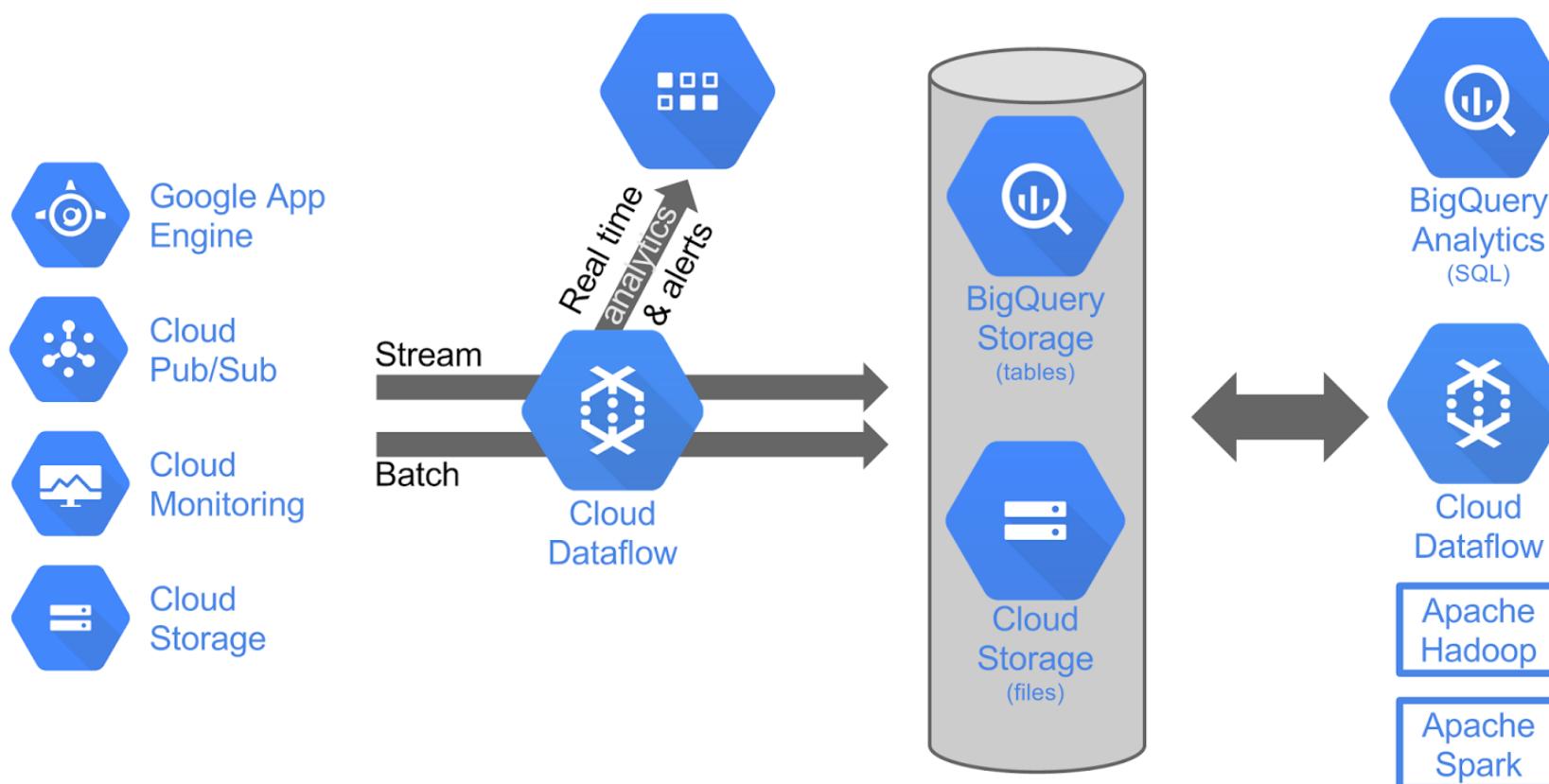
## Data Lifecycle is not a Set Order

Ingest

Process

Store

Analyze

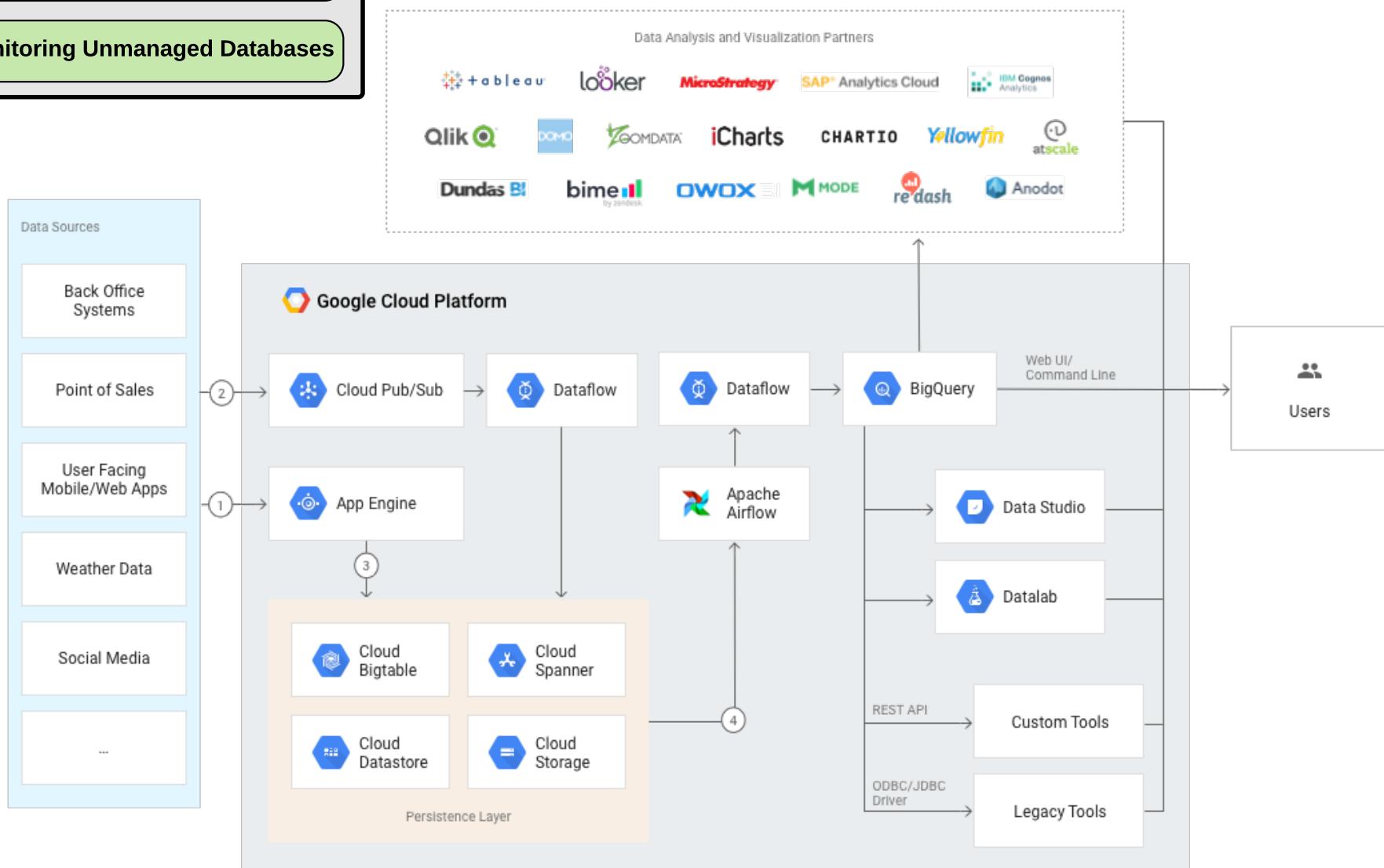


[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)[Previous](#)

## Increasing Complexity of Data Flow



[Return to Table of Contents](#)

## Choose a Lesson

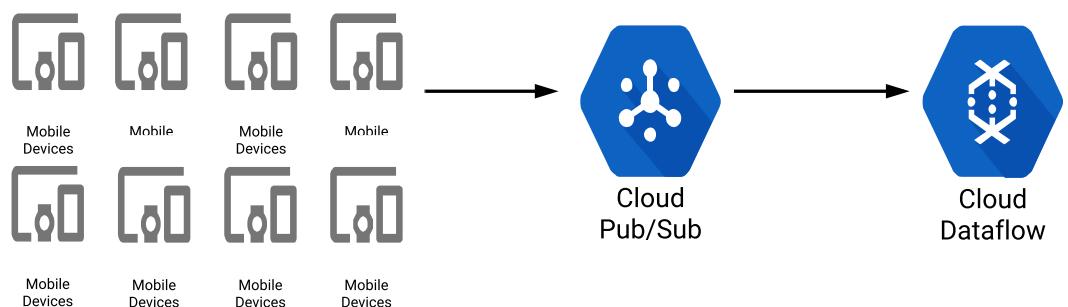
[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)

## *Streaming and Batch Data*

### Data Lifecycle = Data Ingest

#### Streaming (or real-time) data:

- Generated and transmitted continuously by many data sources
- Thousands of data inputs, sent simultaneously, in small sizes (KB)
- Commonly used for telemetry - collecting data from a high number of geographically dispersed devices as it's generated
- Examples:
  - Sensors in transportation vehicles - detecting performance and potential issues
  - Financial institution tracks stock market changes
- Data is processed in small pieces as it comes in
- Requires low latency
- Typically paired with Pub/Sub for the streaming data ingest and Dataflow for real-time processing

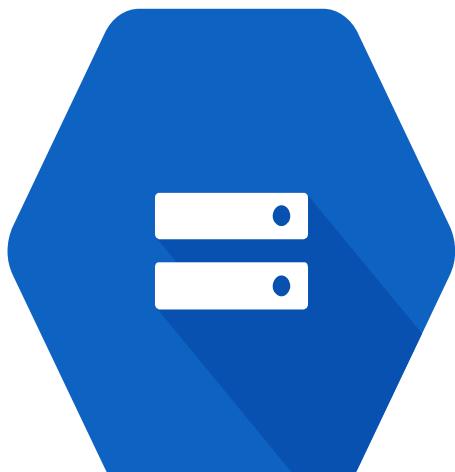


#### Batch (or bulk) data:

- Large sets of data that 'pool' up over time
- Transferring from a small number of sources (usually 1)
- Examples:
  - On-premise database migration to GCP
  - Importing legacy data into Cloud Storage
  - Importing large datasets for machine learning analysis
  - `gsutil cp [storage_location] gs://[BUCKET]` is an example of batch data import
- Low latency is not as important
- Often stored in storage services such as cloud storage, CloudSQL, BigQuery, etc.

[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)

Cloud Storage

## *Cloud Storage as Staging Ground*

[Next](#)

### Storage 'swiss army knife':

- **GCS holds all data types:**
  - All database transfer types, raw data, any format
- **Globally available:**
  - Multi-regional buckets provide fast access across regions
  - Regional buckets provide fast access for single regions
  - Edge caching for increased performance
- **Durable and reliable:**
  - Versioning and redundancy
- **Lower cost than persistent disk**
- **Control access:**
  - Project, bucket, or object level
  - Useful for ingest, transform, and publish workflows
  - Option for Public read access

### Data Engineering perspective:

- **Migrating existing workloads:**
  - Migrate databases/data into Cloud Storage for import
- **Common first step of data lifecycle - get data to GCS**
- **Staging area for analysis/processing/machine learning import:**
  - 'Data lake'

[Return to Table of Contents](#)

## Getting data in and out of Cloud Storage

### Choose a Lesson

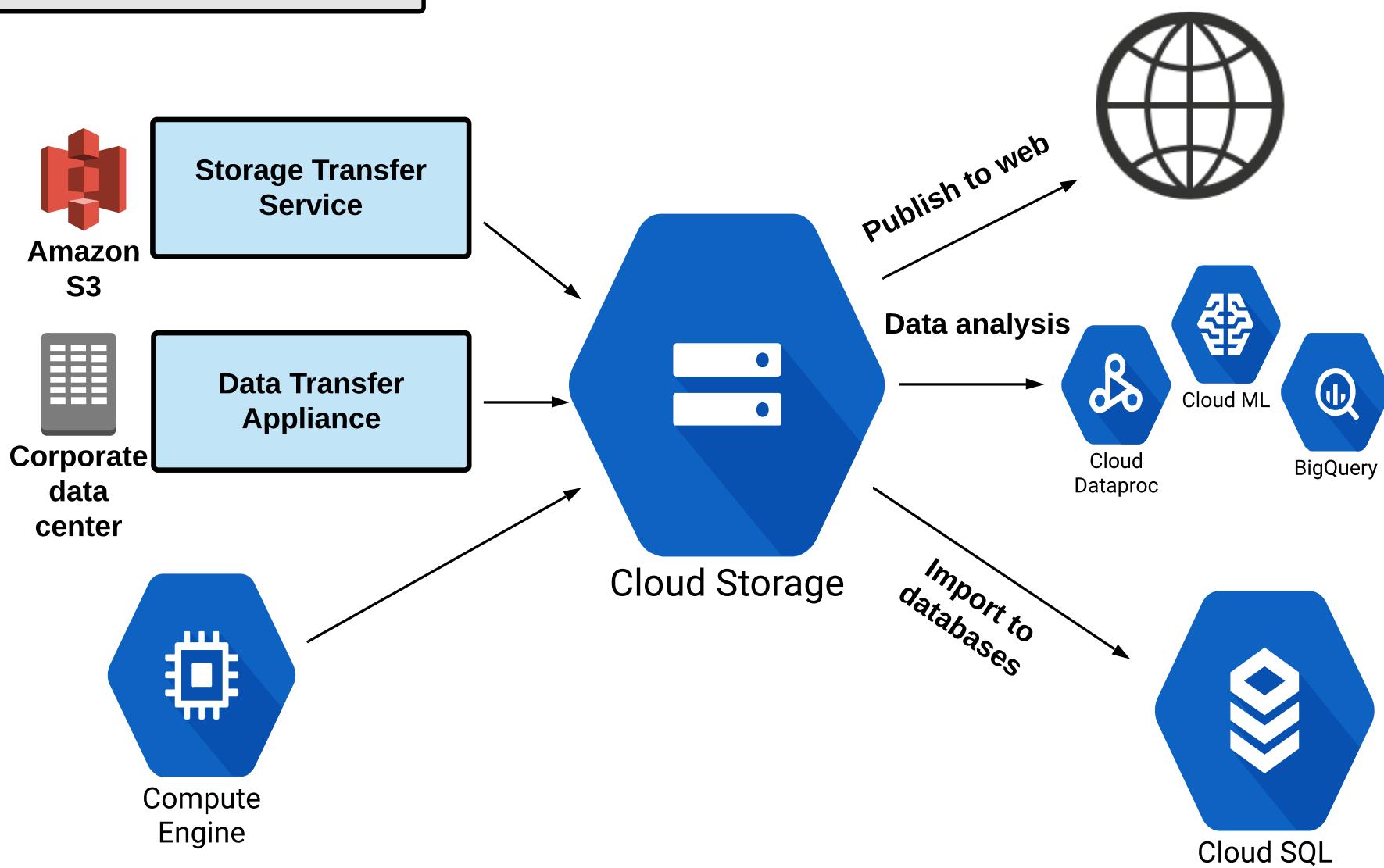
[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)[Previous](#)

### Storage Transfer Service - S3, GCS, HTTP --> GCS:

- One time transfer, periodic sync

### Data Transfer Appliance - physically shipped appliance:

- Load up to 1 petabyte, ship to GCP, loaded into bucket
- gsutil, JSON API - "gsutil cp ..."



[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)

## Database Types

[Next](#)

### Two primary database types:

- Relational/SQL
- Non-relational/NoSQL

### Relational (SQL) database:

- SQL = Structured Query Language
- Structured and standardized:
  - Tables - rows and columns
- Data integrity
- High Consistency
- ACID compliance:
  - Atomicity, Consistency, Isolation, Durability
- Examples:
  - MySQL, Microsoft SQL Server, Oracle, PostgreSQL
- Applications:
  - Accounting systems, inventory
- Pros:
  - Standardized, consistent, reliable, data integrity
- Cons:
  - Poor scaling, not as fast performing, not good for semi-structured data
- **"Consistency and reliability over performance"**

[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)[Previous](#)

## Database Types

### Non-relational (NoSQL) Database:

- Non-structured (no table)
- Different standards - key/value, wide table
- Some have ACID compliance (Datastore)
- Examples:
  - Redis, MongoDB, Cassandra, HBase, Bigtable, RavenDB
- Application:
  - Internet of Things (IoT), user profiles, high-speed analytics
- Pros:
  - Scalable, high-performance, not structure-limited
- Cons:
  - Eventual consistency, data integrity
- "Performance over consistency"

### Exam expectations:

- Understand descriptions between database types
- Know which database version matches which description
- Example:
  - "Need database with high throughput, ACID compliance not necessary, choose three possible options"

[Return to Table of Contents](#)

## Choose a Lesson

[Data Lifecycle](#)[Batch and Streaming Data](#)[Cloud Storage as Staging Ground](#)[Database Types](#)[Monitoring Unmanaged Databases](#)

## *Monitoring Unmanaged Databases*

### Logging and Monitoring in Unmanaged (GCE) Databases

- Examples: Cassandra, MySQL, MariaDB, MongoDB, HBase
- Hosted on Google Compute Engine instances

### Built In vs. Additional Monitoring

- Built in = no additional configuration needed
  - No application-level data
- **Stackdriver Logging**
  - Audit logs
    - "Who created this instance?"
  - Does not include application logs
- **Stackdriver Monitoring**
  - Instance performance metrics
    - Disk I/O, CPU usage, network connections
  - No application performance metrics

### What if we want application data?

- Use **Stackdriver Agents** – install and configure for the instance

#### Logging agent vs. Monitoring agent

- Logging Agent = Stackdriver Logging = Application Logs
  - Configure with **Fluentd**
- Monitoring Agent = Stackdriver Monitoring = Application performance/metrics/alerts
  - May require plugin configuration

[Return to Table of Contents](#)Choose a Lesson[Choosing a Managed Database](#)[Cloud SQL Basics](#)[Importing Data](#)[SQL Query Best Practices](#)[Next](#)

## *Choosing a Managed Database*

### Big picture perspective:

- At minimum, know which managed database is the best solution for any given use case:
  - Relational, non-relational?
  - Transactional, analytics?
  - Scalability?
  - Lift and shift?

#### Relational

#### Non-relational

#### Object - Unstructured

#### Data Warehouse

**Cloud SQL****Cloud Spanner****Cloud Datastore****Cloud Bigtable****Cloud Storage****BigQuery**Use Case  
e.g.Structured data  
Web frameworkRDBMS+scale  
High transactionsSemi-structured  
Key-value dataHigh throughput  
analyticsUnstructured data  
Holds everythingMission critical  
apps  
Scale+consistencyMedical records  
BlogsGlobal supply  
chain  
RetailProduct catalog  
Game stateGraphs  
IoT  
FinanceMultimedia  
Analytics  
Disaster recoveryLarge data  
analytics  
Processing using  
SQL

[Return to Table of Contents](#)

## Choose a Lesson

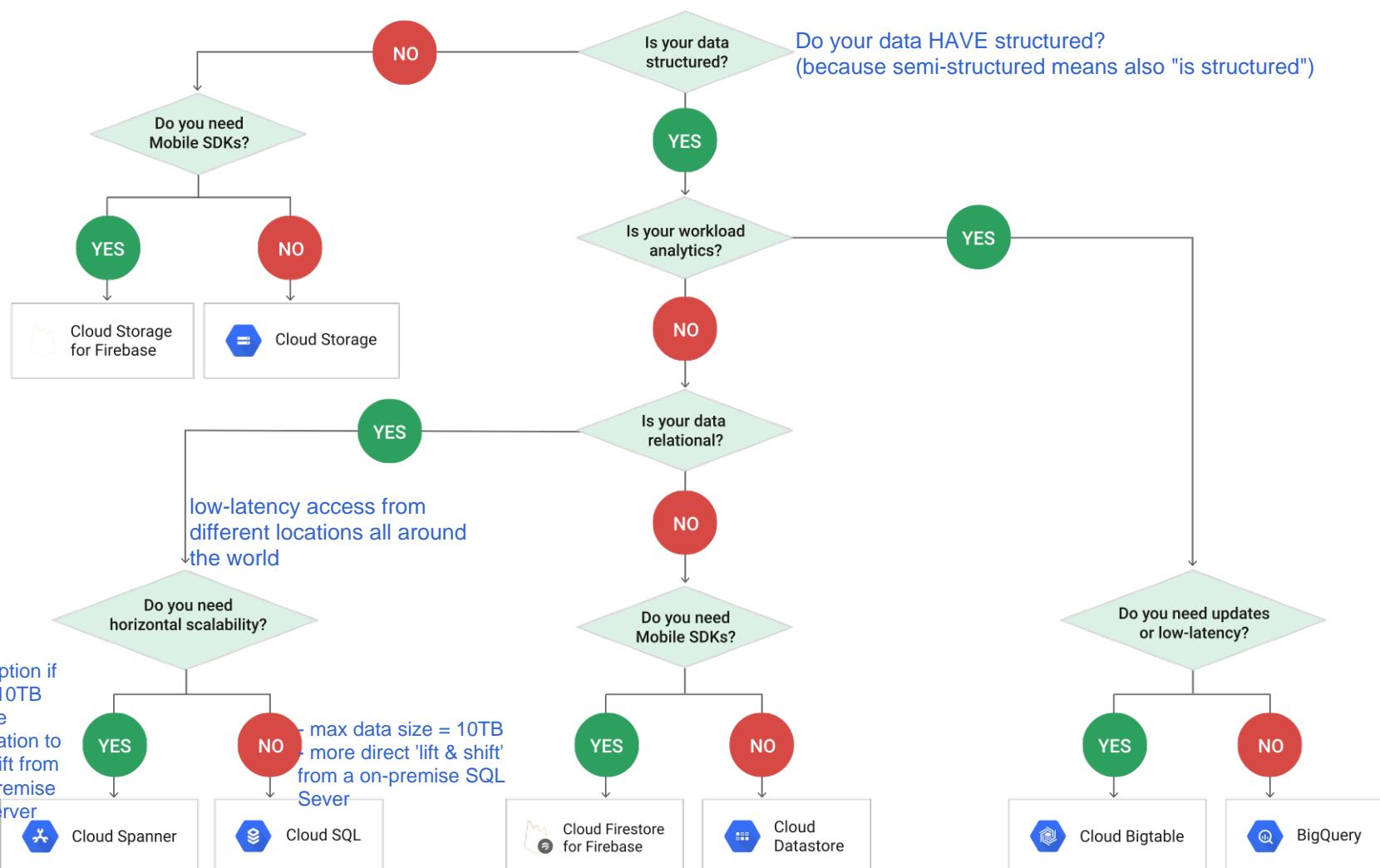
[Choosing a Managed Database](#)
[Cloud SQL Basics](#)
[Importing Data](#)
[SQL Query Best Practices](#)

## *Choosing a Managed Database*

[Previous](#)

### Decision tree criteria:

- Structured (database) or unstructured? [structured store or unstructured store](#)
- Analytical or transactional?
- Relational (SQL) or non-relational (NoSQL)?
- Scalability/availability/size requirements?



[Return to Table of Contents](#)

## Choose a Lesson

[Choosing a Managed Database](#)
[Cloud SQL Basics](#)
[Importing Data](#)
[SQL Query Best Practices](#)

## Cloud SQL Basics

**What is Cloud SQL?** (resume) is MySQL/PostgreSQL however it's in Managed Format

- Direct lift and shift of traditional MySQL/PostgreSQL workloads with the maintenance stack managed for you

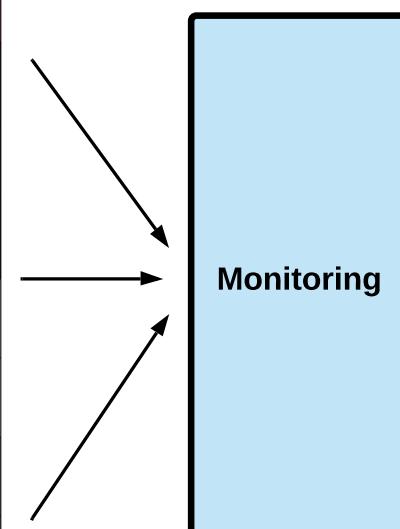
### What is managed?

- OS installation/management
- Database installation/management
- Backups
- Scaling - disk space (if we need to increase the disk size?)
- Availability:
  - Failover
  - Read replicas
- Monitoring
- Authorize network connections/proxy/use SSL

**Limitations:** that's not True anymore ???? because the Cross-region read replicas is available now !!!

- Read replicas limited to the same region as the master:
  - Limited global availability
- Max disk size of 10 TB
- If > 10 TB is needed, or global availability in RDBMS, use Spanner

failover: a method of protecting computer systems from failure, in which standby equipment automatically takes over when the main system fails.



Monitoring

In Cloud SQL:

- high memory machine -> high network throughput
- more storage capacity -> more disk throughput/IOPS
- read replicas
  - + near realtime replication from master instance
  - + offload read requests or analytic traffic from master

when you're working with even an UNmanaged Compute Engine Instance ON Google Cloud, these 3 things (in grey) are handled for you

The purpose of a Read Replica is to scale the use of a database without affecting its performance or needing to resize it. A Read Replica can handle additional reads from clients, thereby distributing the load. Note that you CAN'T send a write to a read-replica server.

[Return to Table of Contents](#)

## Choose a Lesson

[Choosing a Managed Database](#)[Cloud SQL Basics](#)[Importing Data](#)[SQL Query Best Practices](#)

## *Importing Data*

### Importing data into Cloud SQL:

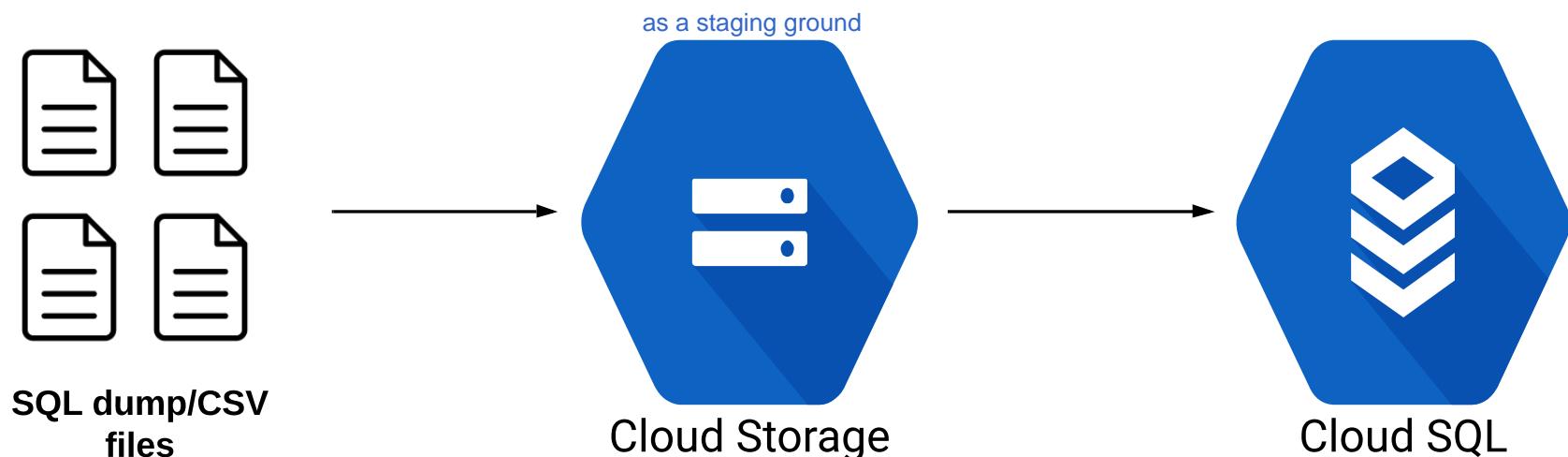
- **Cloud Storage as a staging ground**
- SQL dump/CSV file format

### Export/Import process:

- Export SQL dump/CSV file:
  - SQL dump file **cannot** contain triggers, views, stored procedures
- Get dump/CSV file into Cloud Storage
- Import from Cloud Storage into Cloud SQL instance

### Best Practices:

- Use correct flags for dump file (`--flag_name`):
  - Databases, hex-blob, skip-triggers, set-gtid-purged=OFF, ignore-table
- **Compress data** to reduce costs:
  - Cloud SQL can import compressed .gz files
- Use InnoDB for Second Generation instances



[Return to Table of Contents](#)

## Choose a Lesson

[Choosing a Managed Database](#)[Cloud SQL Basics](#)[Importing Data](#)[SQL Query Best Practices](#)

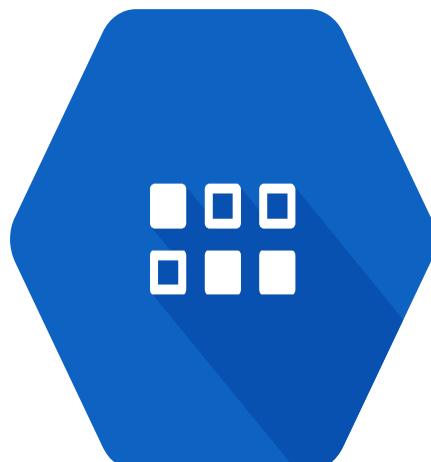
## ***SQL Query Best Practices***

### **General SQL efficiency best practices:**

- More, smaller tables better than fewer, large tables:
  - Normalization of tables
- Define your SELECT fields instead of using SELECT \*:
  - SELECT \* acts as a 'select all'
- When joining tables, use INNER JOIN instead of WHERE:
  - WHERE creates more variable combinations = more work

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)

Cloud Datastore

## Cloud Datastore Overview

[Next](#)

### What is Cloud Datastore?

- **No Ops:** just use, no set-up steps
  - No provisioning of instances, compute, storage, etc.
  - Compute layer is abstracted away
- **Highly scalable:**
  - Multi-region access available
  - Sharding/replication handled automatically
- **NoSQL/non-relational database:**
  - Flexible structure/relationship between objects

### Use Datastore for:

- Applications that need highly available structured data, at scale
- Product catalogs - real-time inventory
- User profiles - mobile apps
- Game save states
- ACID transactions - e.g., transferring funds between accounts

### Do not use Datastore for:

- Analytics (full SQL semantics):
  - Use BigQuery/Cloud Spanner
- Extreme scale (10M+ read/writes per second):
  - Use Bigtable
- Don't need ACID transactions/data not highly structured:
  - Use Bigtable
- Lift and shift (existing MySQL):
  - Use Cloud SQL
- Near zero latency (sub-10ms):
  - Use in-memory database (Redis)

In practice, now we use either Firestore in Native mode or in Datastore mode

Read this: <https://stackoverflow.com/questions/47689003/google-firebase-a-subset-or-superset-of-google-cloud-datastore>

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)

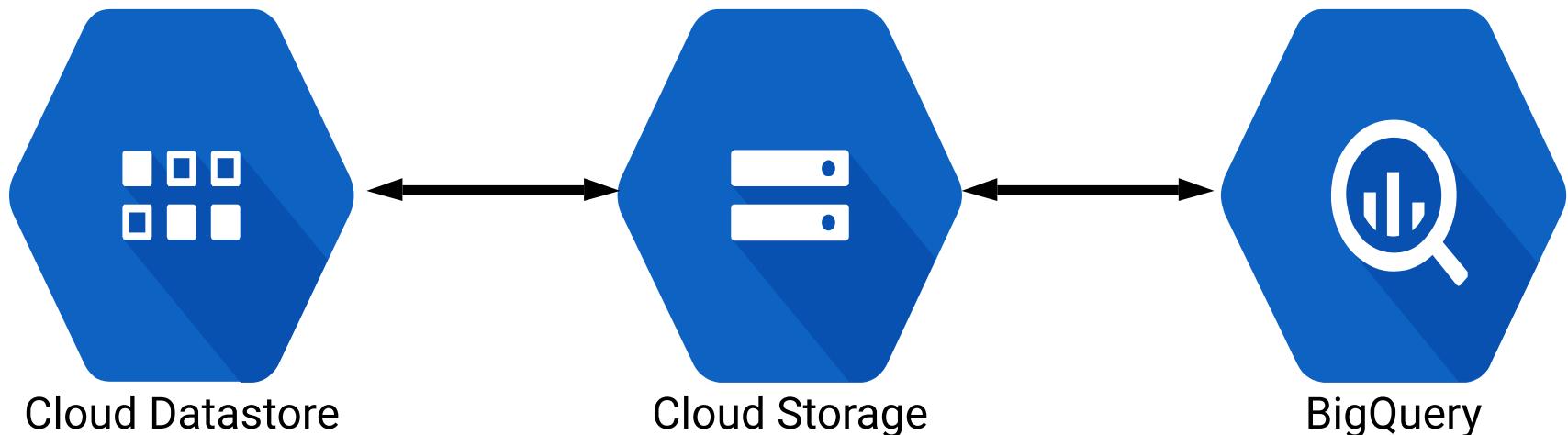
## Cloud Datastore Overview

[Previous](#)

### Other important facts:

- Single Datastore database per project 1 Datastore per project
- Multi-regional for wide access, single region for lower latency and for single location
- Datastore is a transactional database
- Bigtable is an analytical database
- IAM roles:
  - Primitive and predefined
  - Owner, user, viewer, import/export admin, index admin

## Backup/Export/Import/Analyze Managed export/import service



You can use BigQuery for your Datastore Analysis  
after you export from one to the other like the figure

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)

## Data Organization

[Next](#)

### Short version:

- Entities grouped by kind (category)
- Entities can be hierarchical (nested)
- Each entity has one or more properties
- Properties have a value assigned

Concept	Relational Database	Datastore
Category of object	Table	Kind
Single Object	Row	Entity
Individual data for an object	Column	Property
Unique ID for an object	Primary key	Key

[Return to Table of Contents](#)

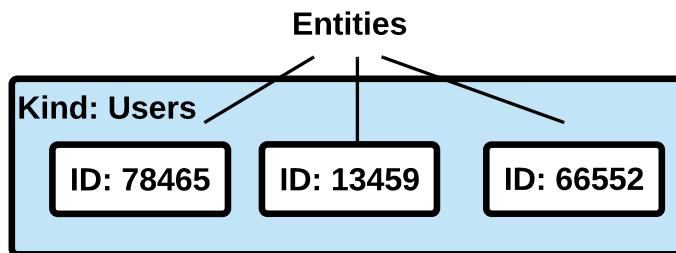
### Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)

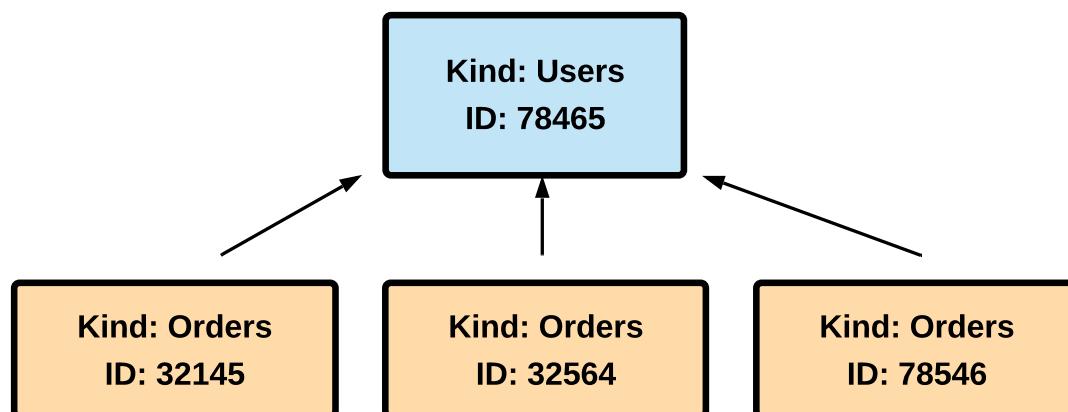
## *Data Organization*

[Previous](#)

### Simple Collections of Entities



### Hierarchies (Entity Groups)



[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)[Next](#)

## Queries and Indexing

### Query:

- Retrieve an **entity** from Datastore that meets a set of conditions
- Query includes:
  - Entity kind
  - Filters
  - Sort order
- Query methods:
  - Programmatic
  - Web console
  - Google Query Language (GQL)

### Indexing:

- Queries gets results from indexes:
  - Contain entity keys specified by index properties
  - Updated to reflect changes
  - Correct query results available with no additional computation needed

### Index types:

- **Built-in** - default option:
  - Allows single property queries
- **Composite** - specified with an index configuration file (`index.yaml`):
  - `gcloud datastore create-indexes index.yaml`

```
indexes:
- kind: Task
  properties:
    - name: tags
    - name: created
- kind: Task
  properties:
    - name: collaborators
    - name: created
```

When querying, make sure there are either simple or composite indexes for the properties you are searching

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)

## *Queries and Indexing*

[Previous](#)

### Danger - Exploding Indexes!

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

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)

## Data Consistency

[Next](#)

### What is data consistency in queries?

- "How up to date are these results?"
- "Does the order matter?"
- **Strongly consistent** = Parallel processes see changes in same order:
  - Query is guaranteed up to date but may take longer to complete
- **Eventually consistent** = Parallel process can see changes out of order, will eventually see accurate end state:
  - Faster query, but may \*sometimes\* return stale results
- **Performance vs. accuracy**
- **Ancestor query/key-value operations** = strong
- **Global queries/projections** = eventual

### Use cases:

- Strong - financial transaction:
  - Make deposit -- check balance
- Eventual - census population:
  - Order not as important, as long as you get eventual result

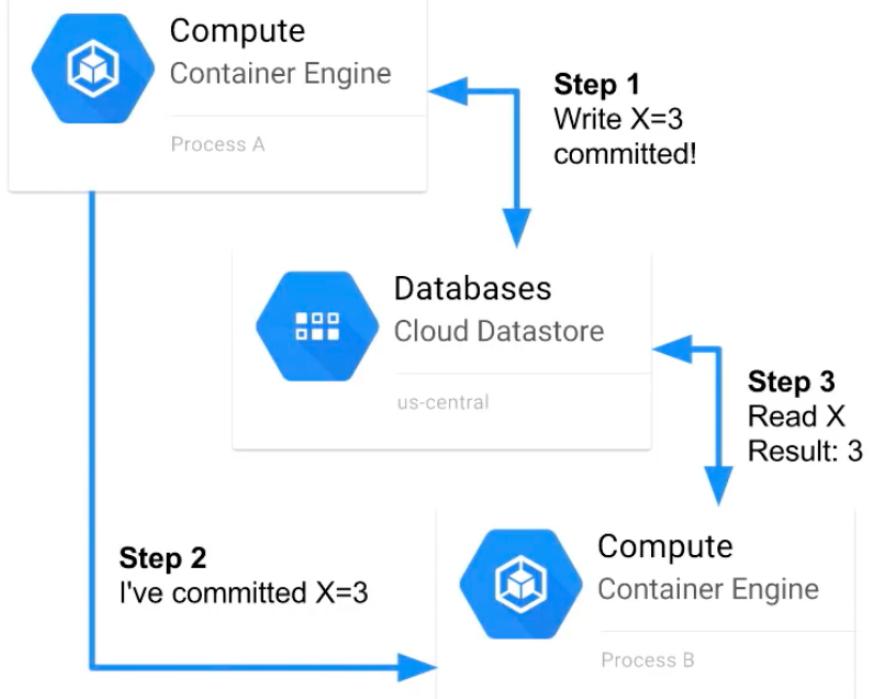
[Return to Table of Contents](#)

### Choose a Lesson

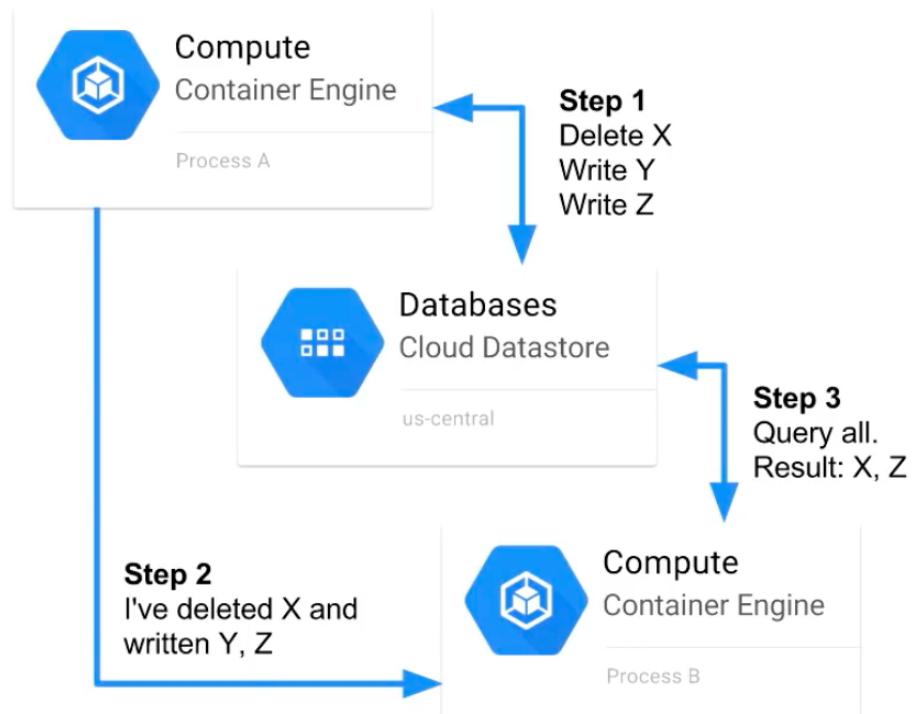
[Cloud Datastore Overview](#)[Data Organization](#)[Queries and Indexing](#)[Data Consistency](#)[Previous](#)

### Strong

## Data Consistency



### Eventual



[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

Cloud Bigtable

## *Cloud Bigtable Overview*

[Next](#)

### What is Cloud Bigtable?

- High performance, massively scalable **NoSQL database**
- Ideal for large **analytical workloads**

### History of Bigtable

- Considered one of the originators for a NoSQL industry
- Developed by Google in 2004
  - Existing database solutions were too slow
  - Needed real-time access to petabytes of data
- Powers Gmail, YouTube, Google Maps, and others

### What is it used for?

- High throughput analytics
- Huge datasets when you have > Terabytes data -> use Bigtable because it's really expensive

### Use Cases

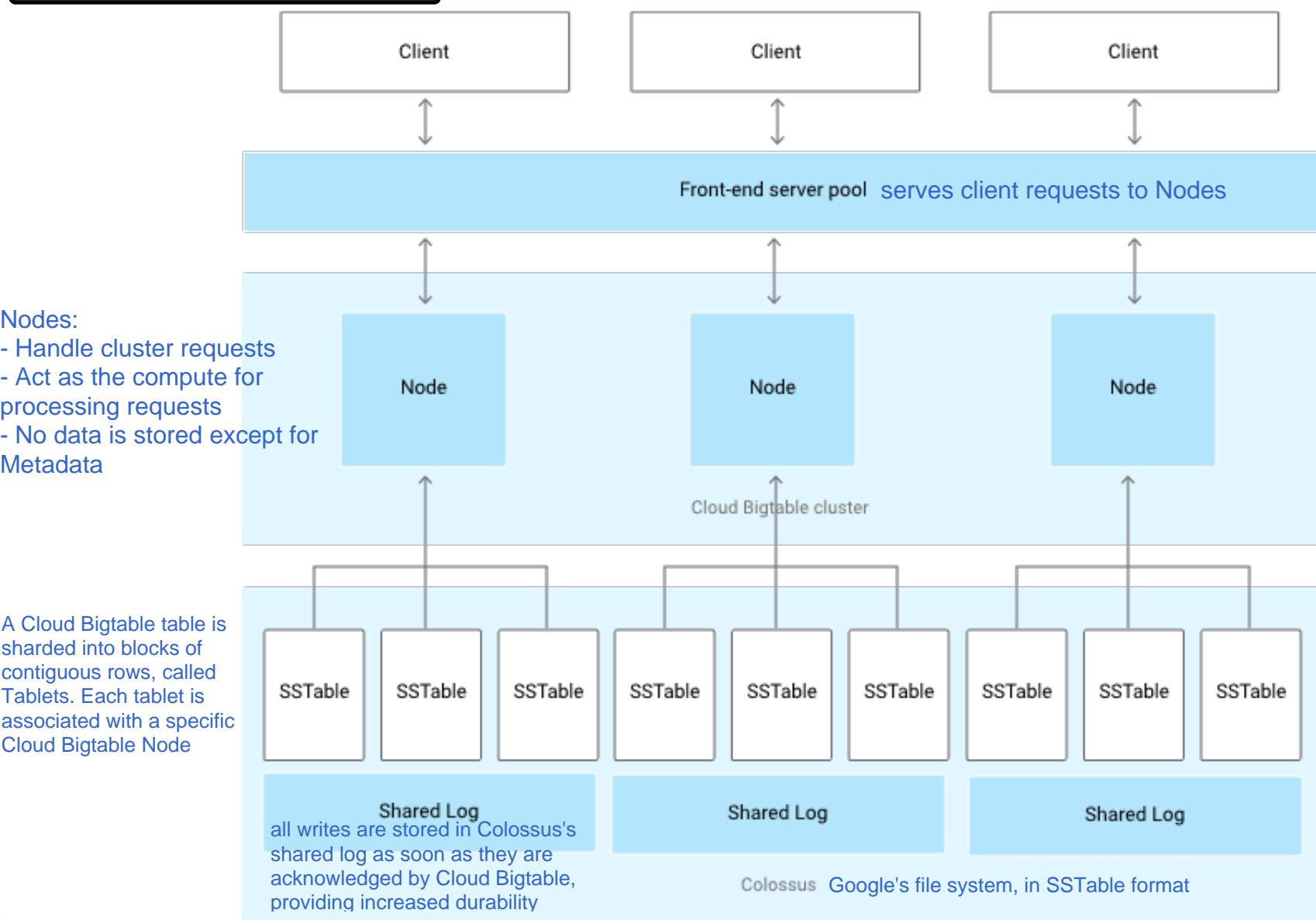
- Financial data – stock prices
- IoT data
- Marketing data – purchase histories

### Access Control

- Project wide or instance level
- Read/Write/Manage

[Return to Table of Contents](#)**Choose a Lesson**[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)[Previous](#)***Cloud Bigtable Overview*****Advantage:**

- Rebalancing tablets from one node to another is very fast, because the actual data is not copied.
- Recovery from the failure of a Cloud Bigtable node is very fast, because only metadata needs to be migrated to the replacement node.
- When a Cloud Bigtable node fails, no data is lost.

**Cloud Bigtable Infrastructure**

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

Cloud Bigtable

## *Instance Configuration*

### Instance basics

[Next](#)

- **Not no-ops**
  - Must configure nodes
- **Entire Bigtable project called 'instance'**
  - All nodes and clusters
- **Nodes grouped into clusters**
  - **1 or more clusters per instance**
- **Auto-scaling storage**
- **Instance types**
  - **Development** - low cost, single node
    - No replication
  - **Production** - 3+ nodes per cluster
    - Replication available, throughput guarantee

### Replication and Changes

- **Synchronize data between clusters** even they are in different zone
  - One additional cluster, total
  - (Beta) available cross-region
- **Resizing**
  - Add and remove nodes and clusters with **no downtime**
- **Changing disk type (e.g. HDD to SSD) requires new instance**

### Interacting with Bigtable

- **Command line - cbt tool or HBase shell**
  - cbt tool is simpler and preferred option

Cloud Bigtable instances with only 1 cluster DO NOT use Replication. If you add a second cluster to an instance, Cloud Bigtable automatically starts replicating your data by keeping separate copies of the data in each of the clusters' zones and synchronizing updates between the copies.

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

Cloud Bigtable

[Previous](#)

## *Instance Configuration*

### **Bigtable interaction using cbt**

- Install the cbt command in Google SDK
  - sudo gcloud components update
  - gcloud components install cbt
- Configure cbt to use your project and instance via .cbtrc file'
  - echo -e "project = [PROJECT\_ID]\ninstance = [INSTANCE\_ID]" > ~/.cbtrc
- **Create** table
  - cbt createtable my-table
- **List** table
  - cbt ls
- **Add column family**
  - cbt createfamily my-table cf1
- **List column family**
  - cbt ls my-table
- **Add value to row 1, using column family cf1 and column qualifier c1**
  - cbt set my-table r1 cf1:c1=test-value
- **Delete** table (if not deleting instance)
  - cbt deletetable my-table
- **Read** the contents of your table
  - cbt read my-table

Get help with cbt command using 'cbt --help'

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

## Data Organization

### Data Organization

- One big table (hence the name Bigtable)
- Table can be thousands of columns/billions of rows
- Table is sharded across tablets

### Table components

- Row Key
  - First column
- Columns grouped into column families

Each row/column intersection can contain multiple cells, or versions, at different timestamps, providing a record of how the stored data has been altered over time.

Cloud Bigtable stores data Lexicographically.

	Column-Family-1		Column-Family-2	
Row Key	Column-Qualifier-1	Column-Qualifier-2	Column-Qualifier-1	Column-Qualifier-2
r1	r1, cf1:cq1	r1, cf1:cq2	r1, cf1:cq1	r1, cf1:cq2
r2	r2, cf1:cq1	r2, cf1:cq2	r2, cf1:cq1	r2, cf1:cq2

### Indexing and Queries

- Only the row key is indexed
- Anything you want to do a search for has to be in your Row Key
- Schema design is necessary for efficient queries!
- Field promotion - move fields from column data to row key

Example Field promotion

Row key	Column data
BATTERY#Corrie#20150301124501001	METRIC:PERCENTAGE:98
BATTERY#Corrie#20150301124501003	METRIC:PERCENTAGE:96
BATTERY#Jo#20150301124501002	METRIC:PERCENTAGE:54
BATTERY#Sam#20150301124501004	METRIC:PERCENTAGE:43
BATTERY#Sam#20150301124501005	METRIC:PERCENTAGE:38

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

X

Front-end server pool serves client requests to nodes

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

X

Nodes handle cluster requests. It acts as the compute for processing requests.

No data is stored on the node except for metadata to direct requests to the correct tablet

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

**X**

Bigtable's table is sharded into block of rows, called tablets.  
Tablets are stored on Colossus, Google's file system, in SStable format  
Storage is separate from the compute nodes, though each tablet is associated with a node.  
As a result, replication and recovery of node data is very fast, as only metadata/pointers need to be updated.

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Bigtable Overview](#)[Instance Configuration](#)[Data Organization](#)[Schema Design](#)

### Row Key

memusage+user+timestamp

20-mattu-201805082048

To get the best write performance from Cloud Bigtable, it's important to distribute writes as evenly as possible across nodes. One way to achieve this goal is by using Row Keys that DO NOT follow a predictable order.

## Schema Design

- Per table – Row key is the only indexed item
- ~~Keep all entity info in a single row~~ Keep all information for an entity in a single row
- Related entities should be in adjacent rows Cloud Bigtable stores adjacent row keys on the same server node
- More efficient reads
- Tables are sparse – empty columns take no space empty cells

## Schema Efficiency

- Well-defined row keys = less work
  - Multiple values in row key
- Row key (or prefix) should be sufficient for a search
- Goal = spread loads over multiple nodes
  - All on one node = hotspotting

## Row Key Best Practices

- Good row keys = distributed load
  - Reverse domain names (com.linuxacademy.support)
  - String identifiers (mattu)
  - Timestamps (reverse, NOT at front/or only identifier)
- Poor row keys = hotspotting
  - Domain names (support.linuxacademy.com)
  - Sequential ID's
  - Timestamps alone/at front

## Table Design - Time Series Data

- For time series data, use tall and narrow tables (one event per row)
  - Easier to run queries against data

More example: <https://cloud.google.com/bigtable/docs/schema-design-time-series>

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

## *Cloud Spanner Overview*

[Next](#)

### What is Cloud Spanner?

- Fully managed, highly scalable/available, **relational database**
- **Similar architecture to Bigtable**
- "NewSQL"

### What is it used for?

- Mission critical, relational databases that need **strong transactional consistency (ACID compliance)**
- Wide scale availability
- Higher workloads than Cloud SQL can support
- Standard SQL format (ANSI 2011)

### Horizontal vs. vertical scaling

- **Vertical** = **more** compute on single instance (**CPU/RAM**) more resources
- **Horizontal** = **more instances** (nodes) sharing the load

### Compared to Cloud SQL

- Cloud SQL = Cloud incarnation of **on-premises MySQL** database
- Spanner = designed from the ground up for the cloud
- Spanner is not a 'drop in' replacement for MySQL
  - Not MySQL/PostgreSQL compatible
  - Work required to migrate
  - However, when making transition, don't need to choose between consistency and scalability

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

## **Cloud Spanner Overview**

[Previous](#)[Next](#)

**Transactional Consistency vs. Scalability**  
**Why not both?**

	Cloud Spanner	Traditional Relational	Traditional Non-relational
Schema	Yes	Yes	No
SQL	Yes	Yes	No
Consistency	Strong	Strong	Eventual
Availability	High	Failover	High
Scalability	Horizontal	Vertical	Horizontal
Replication	Automatic	Configurable	Configurable

**Primary purpose of Cloud Spanner:**  
**No compromises relational database**

[Return to Table of Contents](#)

## Cloud Spanner Overview

### Choose a Lesson

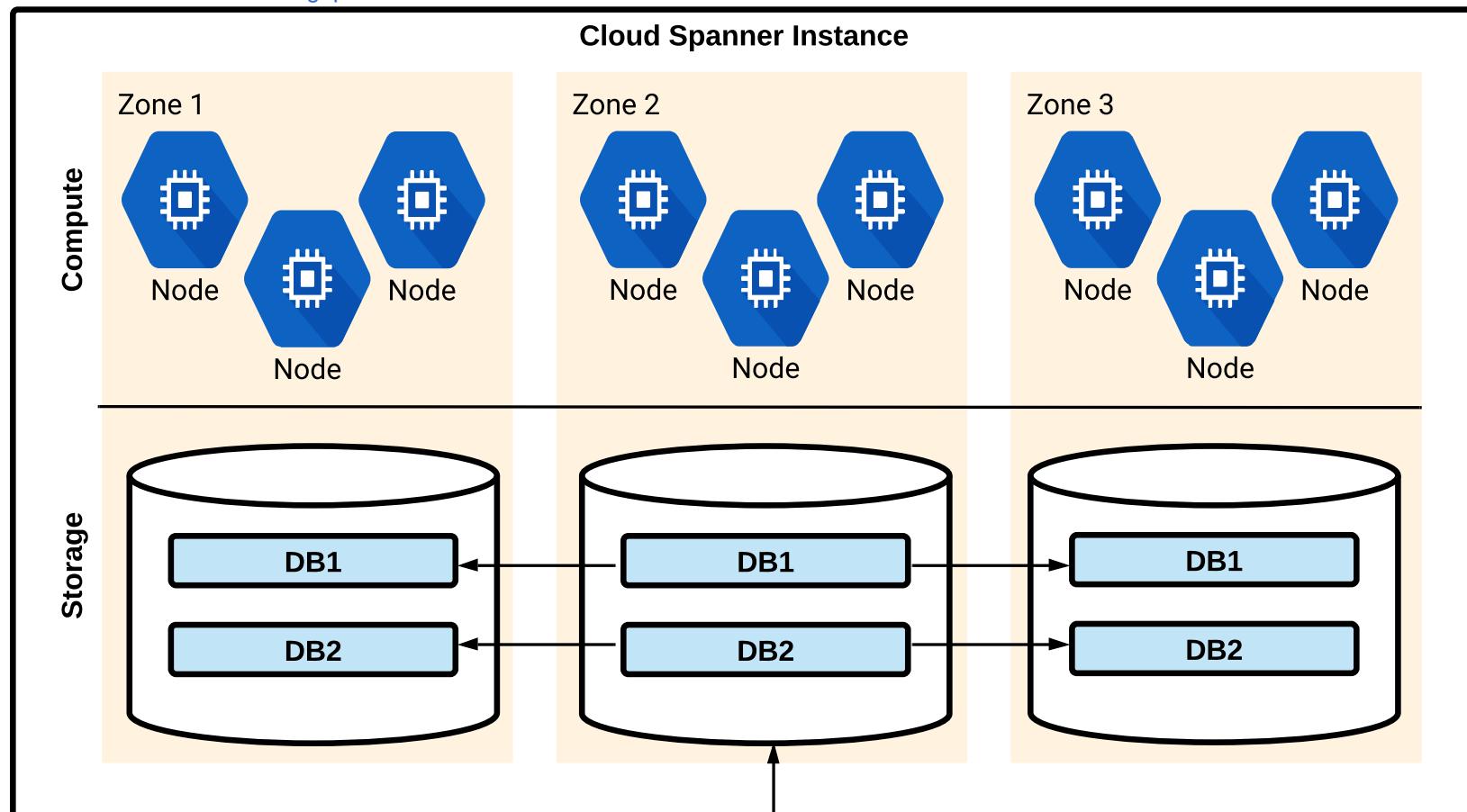
[Cloud Spanner Overview](#)[Previous](#)[Next](#)[Data Organization and Schema](#)

## Cloud Spanner Architecture (similar to Bigtable)

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



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 replication

- When data is written, you know it is been written
- Any reads guarantee data accuracy

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

## ***Cloud Spanner Overview***

[Previous](#)

### **Identity and Access Management (IAM)**

- Project, Instance, or Database level
- roles/spanner.
- Admin - Full access to all Spanner resources
- Database Admin - Create/edit/delete databases, grant access to databases
- Database Reader - read/execute database/schema
- Viewer - view instances and databases
  - Cannot modify or read from database

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

X

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

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

X

Storage is replicated across zones (and regions, where applicable).  
Like Bigtable, storage is separate from computing nodes

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

X

Whenever an update is made to a database in one zone/region, it is automatically replicated across zones/regions.

Automatic synchronous replication

- When data is written, you know it has been written
- Any reads guarantee data accuracy

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)
[Data Organization and Schema](#)
[Next](#)

## Data Organization and Schema

### Organization

- RDBMS = tables
- Supports SQL joins, queries, etc
- Same SQL dialect as BigQuery
- Tables are handled differently
  - Parent/child tables
  - Interleave Data Layout

- Data in Cloud Spanner is strongly typed
- There are two ways to define parent-child relationships in Cloud Spanner:
  - + table interleaving
  - + foreign keys.

Classical  
Typical Relational Database

Two sets of related data = Two tables

SingerId	SingerName
1	Beatles
2	U2
3	Pink Floyd

SingerId	AlbumId	AlbumName
1	1	Help!
1	2	Abbey Road
3	1	The Wall

Image below: Physical layout of rows of Singers and its child table Albums.

Singers(1)	"Marc"	"Richards"	<Bytes>		
Albums(1, 1)				"Total Junk"	
Albums(1, 2)				"Go, Go, Go"	
Songs(1, 2, 1)					"42"
Songs(1, 2, 2)					"Nothing Is The Same"
Singers(2)	"Catalina"	"Smith"	<Bytes>		
Albums(2, 1)				"Green"	
Songs(2, 1, 1)					"Let's Get Back Together"
Songs(2, 1, 2)					"Starting Again"
Songs(2, 1, 3)					"I Knew You Were Magic"
Albums(2, 2)				"Forever Hold Your Peace"	
Albums(2, 3)				"Terrified"	
Songs(2, 3, 1)					"Fight Story"

### Spanner

#### Interleave Tables

- Interleave is a good choice where the child table's primary key includes the parent table's primary key columns

- If a parent table's primary key is composed of N columns, the primary key of each of its child tables must also have those same N columns, in the same order and starting with the same column, as the prefix.

- Cloud Spanner physically co-locate the child table(s) with their parent

+ Cloud Spanner stores rows in sorted order by primary key values, with child rows inserted between parent rows

+ child tables are also called interleaved tables

+ The parent row must exist before you can insert child rows

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Spanner Overview](#)[Data Organization and Schema](#)

### Choosing a primary key

- Hash the key
- Swap the order of the columns in the PK
- Use a UUID (version 4 is recommended)
- Bit-reverse

## **Data Organization and Schema**

[Previous](#)

### Recall

- primary key can be composed of zero or more columns of that table
- the idea of **Avoiding Hotspotting** is Spreading the load across multiple servers

### Primary keys and Schema

- How to tell which child tables to store with which parent tables
- Usually a natural fit
  - 'Customer ID'
  - 'Invoice ID'
- **Avoid hotspotting**
  - No sequential numbers if you insert records with a monotonically increasing integer as the key, your inserts will be directed at a single server, creating a hotspot
  - No timestamps (also sequential)
    - Use descending order if timestamps required

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

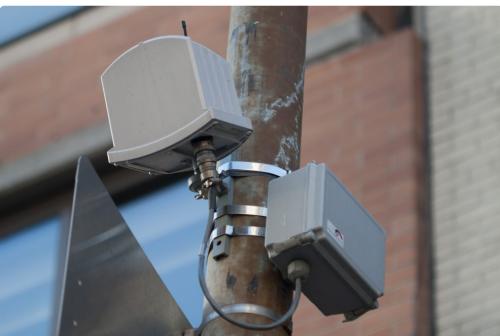
## *Streaming Data Challenges*

### What is Streaming Data?

[Next](#)

- "Unbounded" data
- Infinite, never completes, always flowing

### Examples



Traffic Sensors



Credit Card Transactions



Mobile Gaming

### Fast action is often necessary

- Quickly collect data, gain insights, and take action
- Sending to storage can add latency
- Use cases:
  - Credit card fraud detection
  - Predicting highway traffic

[Return to Table of Contents](#)

## Choose a Lesson

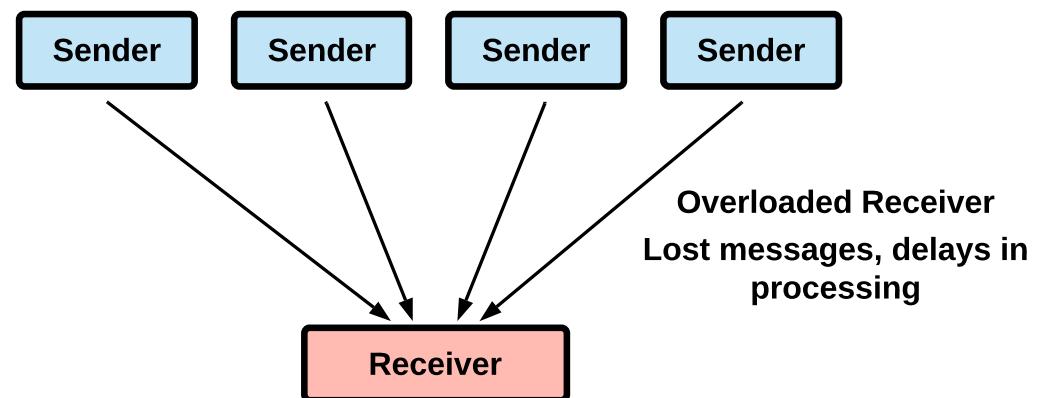
[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)[Previous](#)

## Streaming Data Challenges

### Tight vs. Loose Coupling in Systems

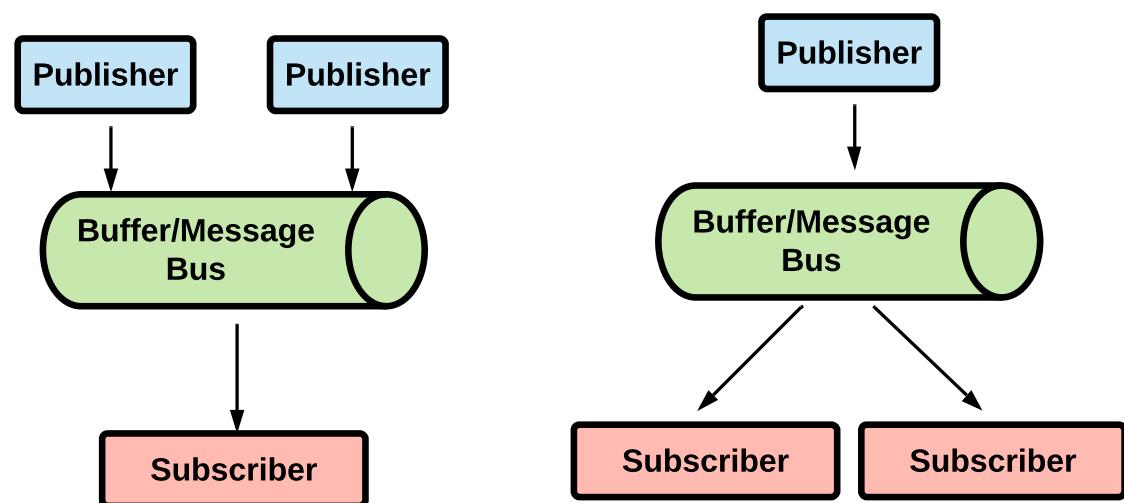
- Tightly (direct) coupled systems more likely to fail
- Loosely coupled systems with 'buffer' scale have better fault tolerance

#### Tightly-Coupled System



#### Loosely-Coupled System

- Fault tolerance
- Scalability
- Message queuing



[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

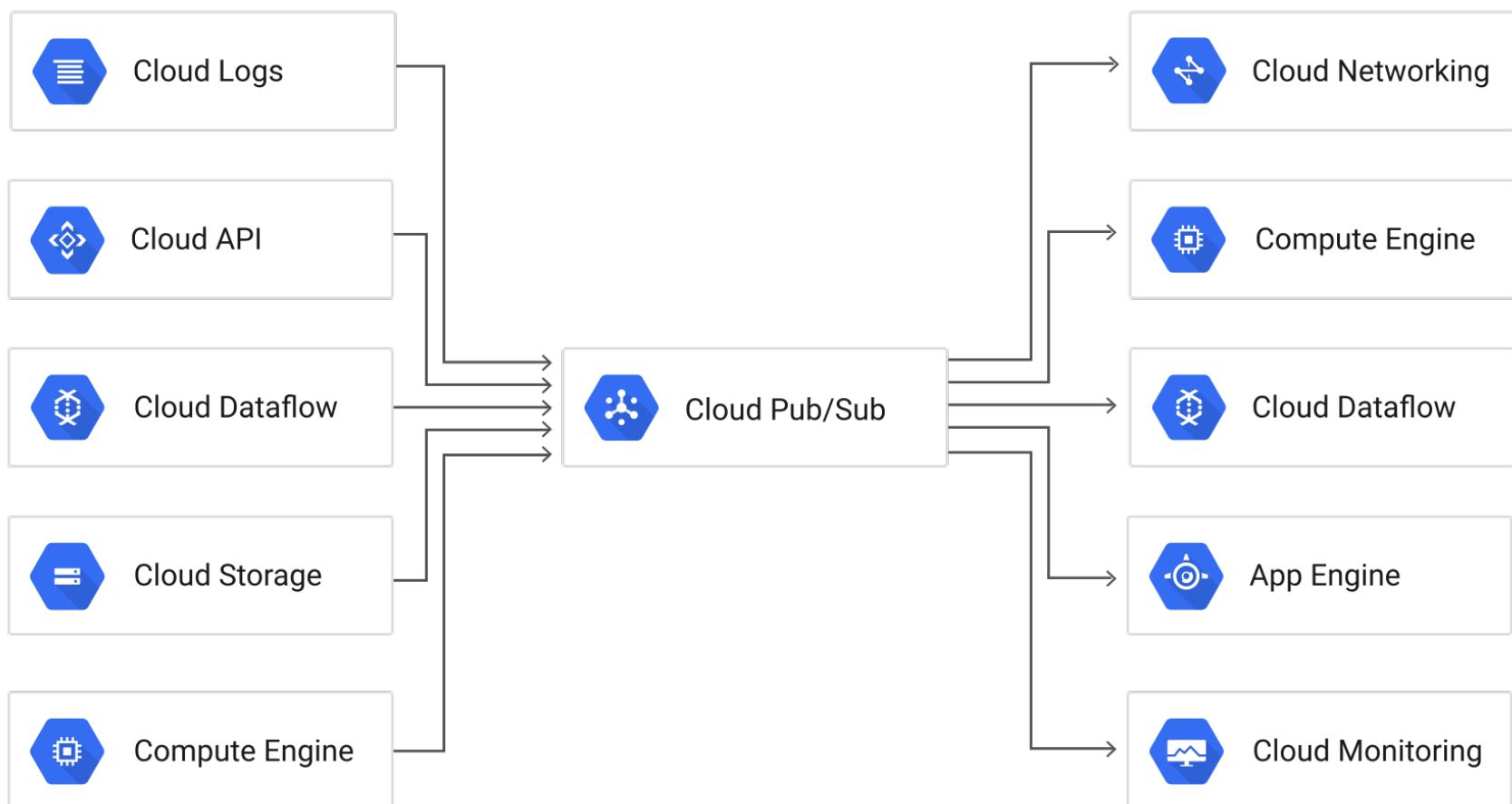
## **Cloud Pub/Sub Overview**

### What is Cloud Pub/Sub?

[Next](#)

- Global-scale messaging buffer/coupler
- NoOps, global availability, auto-scaling
- Decouples senders and receivers
- Streaming data ingest:
  - Also connects other data pipeline services
- Equivalent to Apache Kafka (open source)
- Guaranteed at-least-once delivery

Asynchronous messaging - many to many (or any other combination)



[Return to Table of Contents](#)

## Choose a Lesson

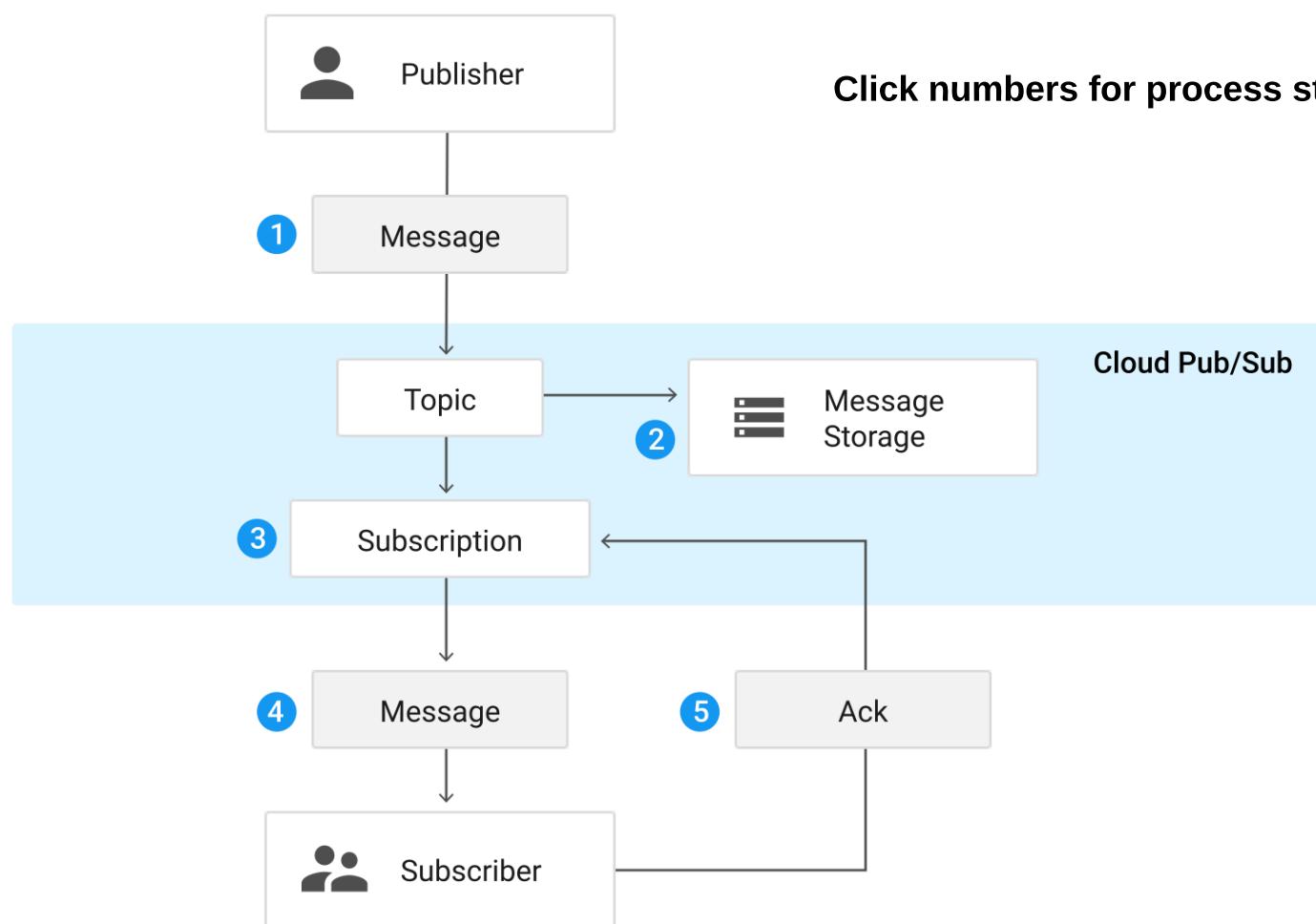
[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

## Cloud Pub/Sub Overview

[Previous](#)[Next](#)

### How It Works: Terminology

- Topics, Messages, Publishers, Subscribers, Message Store



[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

Publisher application creates a *topic* in the Cloud Pub/Sub service and sends *messages* to the topic. A message contains a payload and optional *attributes* that describe the payload content.



[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

X

Messages are stored in a *message store* until they are delivered and acknowledged by subscribers.

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

X

Pub/Sub forwards messages from a topic to all subscribers, individually.

Messages can be either *pushed* by Pub/Sub to subscribers, or *pulled* by subscribers from Pub/Sub.

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

A subscriber receives pending messages from its subscription, and acknowledges each one to the Pub/Sub service.

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

The after message is acknowledged by the subscriber, then it is removed from the subscription's queue of messages.

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

## Cloud Pub/Sub Overview

[Previous](#)[Next](#)

### Push and Pull

- Pub/Sub can either **push** messages to subscribers, or subscribers can **pull** messages from Pub/Sub.
- Push = lower latency, more real-time.
- Push subscribers must be Webhook endpoints that accept POST over HTTPS.
- Pull is ideal for large volume of messages, and uses batch delivery

### IAM

- Allows for controlling access at project, topic, or subscription level
- Admin, Editor, Publisher, Subscriber
- Service accounts are best practice

### Pricing

- Data volume used per month (per GB)

### 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.
- It's possible to add message attributes to help with ordering.

Monthly data	Price Per GB
First 10 GB	\$0.00
Next 50 TB	\$0.06
Next 100 TB	\$0.05
Beyond 150 TB	\$0.04

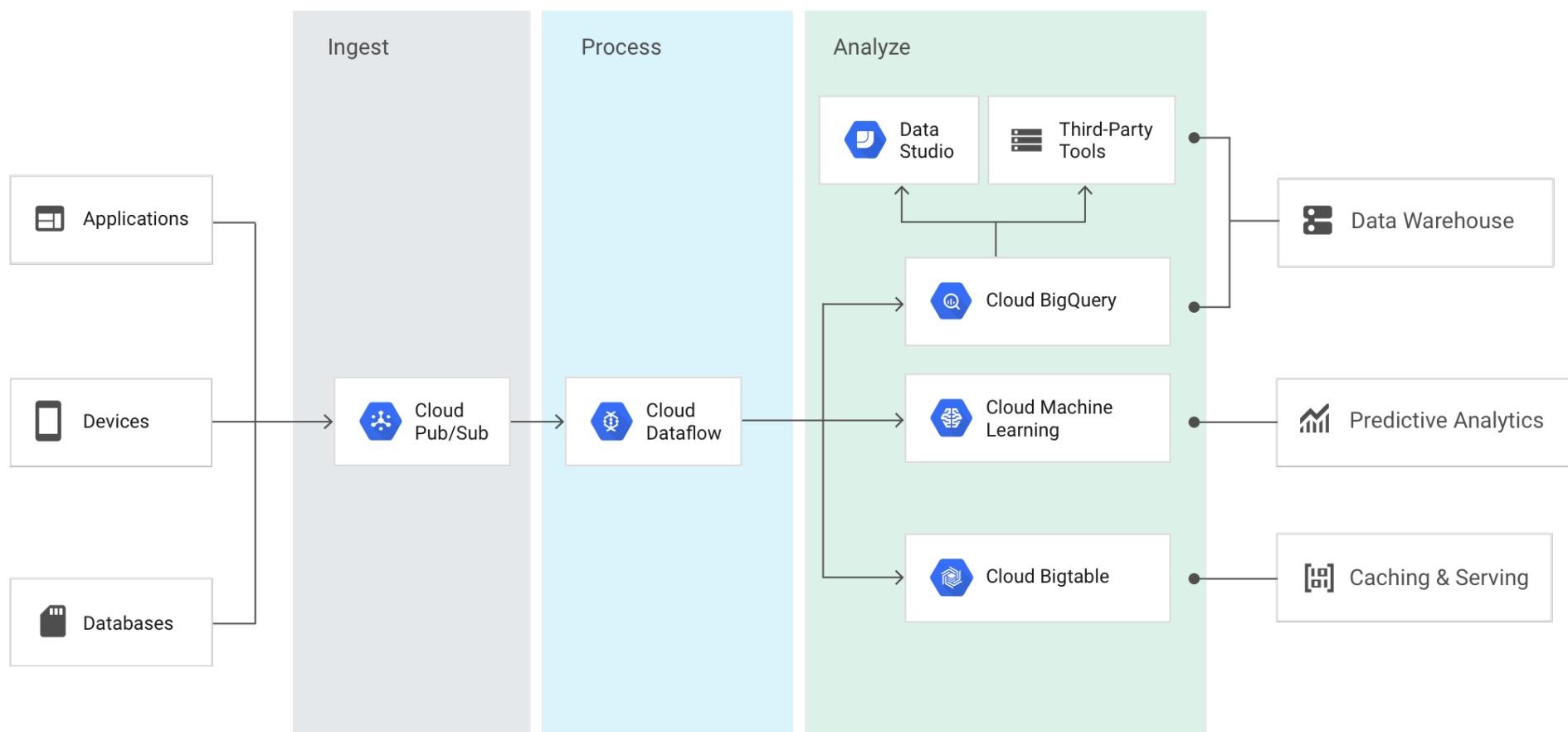
[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)[Previous](#)

## Cloud Pub/Sub Overview

### Big Picture: Data Lifecycle for Streaming Data Ingest



[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)[Next](#)

## **Pub/Sub Hands On**

### **The Steps**

- Create a topic
- Create a subscription
- Publish messages
- Retrieve messages

## **Simple topic/subscription/publish via gcloud**

### **Create a topic called *my-topic*:**

- `gcloud pubsub topics create my-topic`

### **Create subscription to topic *my-topic*:**

- `gcloud pubsub subscriptions create --topic my-topic mySub1`

### **Publish a message to your topic:**

- `gcloud pubsub topics publish my-topic --message "hello"`

### **Retrieve message with your subscription, acknowledge receipt, and remove message from queue:**

- `gcloud pubsub subscriptions pull --auto-ack mySub1`

### **Cancel subscription:**

- `gcloud pubsub subscriptions delete mySub1`

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

## **Pub/Sub Hands On**

[Previous](#)

### Traffic Data Exercise

- Clone GitHub
- Copy data points
- Simulate traffic data
- Pull messages

Clone GitHub data to Cloud Shell (or other SDK environment), and browse to publish folder:

```
cd ~  
git clone https://github.com/linuxacademy/googledataengineer  
cd ~/googledataengineer/courses/streaming/publish
```

Create a topic called **sandiego**:

```
gcloud pubsub topics create sandiego
```

Create subscription to topic **sandiego**:

```
gcloud pubsub subscriptions create --topic sandiego mySub1
```

Run script to download sensor data:

```
./download_data.sh
```

May need to authenticate shell to ensure we have the right permissions:

```
gcloud auth application-default login
```

View script info:

```
vim ./send_sensor_data.py or use viewer of your choice
```

Run python script to simulate one hour of data per minute:

```
./send_sensor_data.py --speedFactor=60 \  
--project=YOUR-PROJECT-ID
```

If you receive error: **google.cloud.pubsub can not be found** or an **ImportError: No module named iterator**, run this **pip** command to install components, then try again:

```
sudo pip install -U google-cloud-pubsub
```

Open new Cloud Shell tab (using + symbol)

Pull message using subscription **mySub1**:

```
gcloud pubsub subscriptions pull --auto-ack mySub1
```

Create a new subscription and pull messages with it:

```
gcloud pubsub subscriptions create --topic sandiego mySub2
```

```
gcloud pubsub subscriptions pull --auto-ack mySub2
```

[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

## *Connecting Kafka to GCP*

### Does Pub/Sub Replace Kafka?

[Next](#)

- Not always
- Hybrid workloads:
  - Interact with existing tools and frameworks
  - Don't need global/scaling capabilities with pub/sub
- Can use *both*: Kafka for on-premises and pub/sub for GCP in same data pipeline

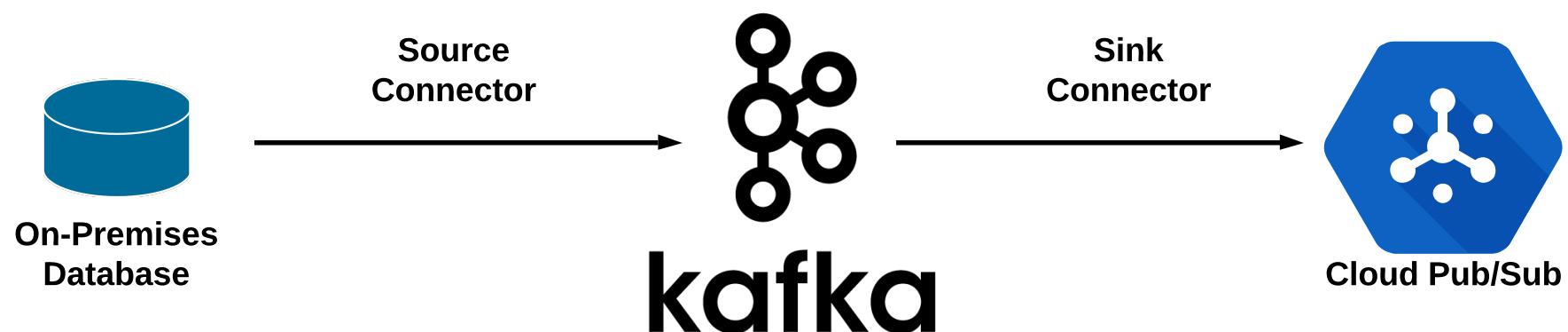
### How do we connect Kafka to GCP?

#### Overview on Connectors:

- Open-source plugins that connect Kafka to GCP
- Kafka Connect: One optional "connector service"
- Exist to connect Kafka directly to pub/sub, Dataflow, and BigQuery (among others)

#### Additional Terms

- **Source connector:** An upstream connector:
  - Streams *from* something *to* Kafka
- **Sink connector:** A downstream connector:
  - Streams *from* Kafka *to* something else



[Return to Table of Contents](#)

## Choose a Lesson

[Streaming Data Challenges](#)[Cloud Pub/Sub Overview](#)[Pub/Sub Hands On](#)[Connecting Kafka to GCP](#)[Monitoring Subscriber Health](#)

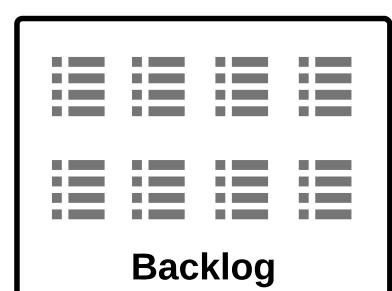
## **Monitoring Subscriber Health**

### In a Perfect World...

- Subscribers and Publishers work in perfect harmony:
  - Example:
    - 1 million messages/second published
    - 1 million messages/second successfully pulled/pushed
    - Result: No backlog in Pub/Sub queue
- But we don't live in a perfect world...
  - Subscriber cannot keep up with publish rate
  - Result: Backlog in Pub/Sub queue

### Troubleshooting Subscriber Health (Backlog)

- Create alerts for (x) backlog threshold
- Subscriber not able to keep up:
  - Under-provisioned
  - Code not optimized
- Not acknowledging message receipt:
  - Pub/Sub doesn't know it's delivered, and keeps trying
  - Subscriber code not properly acknowledging pulled messages
- Check publishers for excessive re-transmits



Sensors



1 million/sec



Cloud Pub/Sub



10,000/sec



Cloud  
Dataflow

[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## Data Processing Challenges

### What is Data Processing?

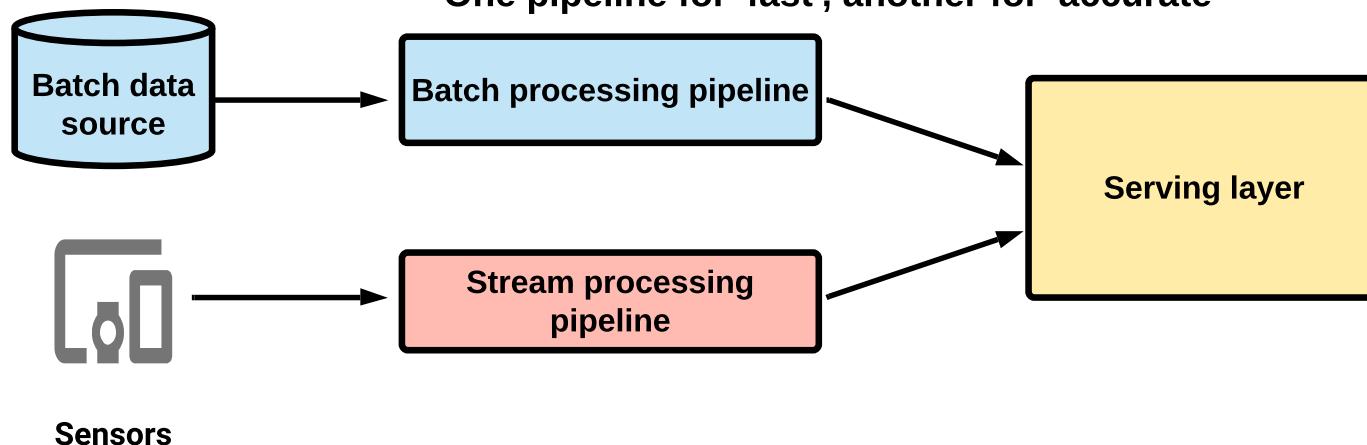
[Next](#)

- Read Data (Input)
- Transform it to be relevant - Extract, Transform, and Load (ETL)
- Create output



### Challenge: Streaming and Batch data pipelines:

- Until recently, separate pipelines are required for each
- Difficult to compare recent and historical data
- One pipeline for 'fast', another for 'accurate'



### Why both?

- Credit card monitoring
- Compare streaming transactions to historical batch data to detect fraud

[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)[Previous](#)

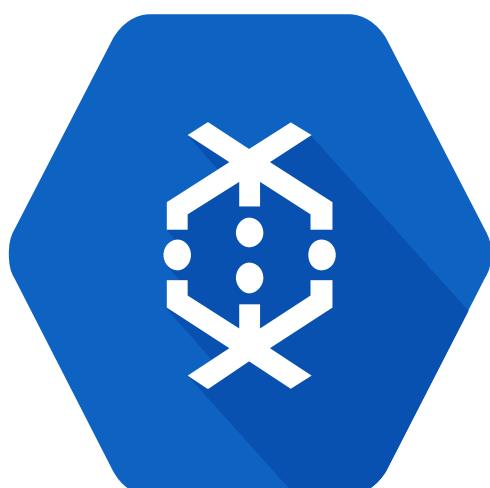
## *Data Processing Challenges*

- Challenge: Complex element processing:**
- Element = single data input
  - One at a time element ingest from single source = easy
  - Combining elements (aggregation) = hard
  - Processing data from different sources, streaming, and out of order (composite) = REALLY hard

## **Solution: Apache Beam + Cloud Dataflow**



beam +



Cloud Dataflow

[Return to Table of Contents](#)

## Choose a Lesson

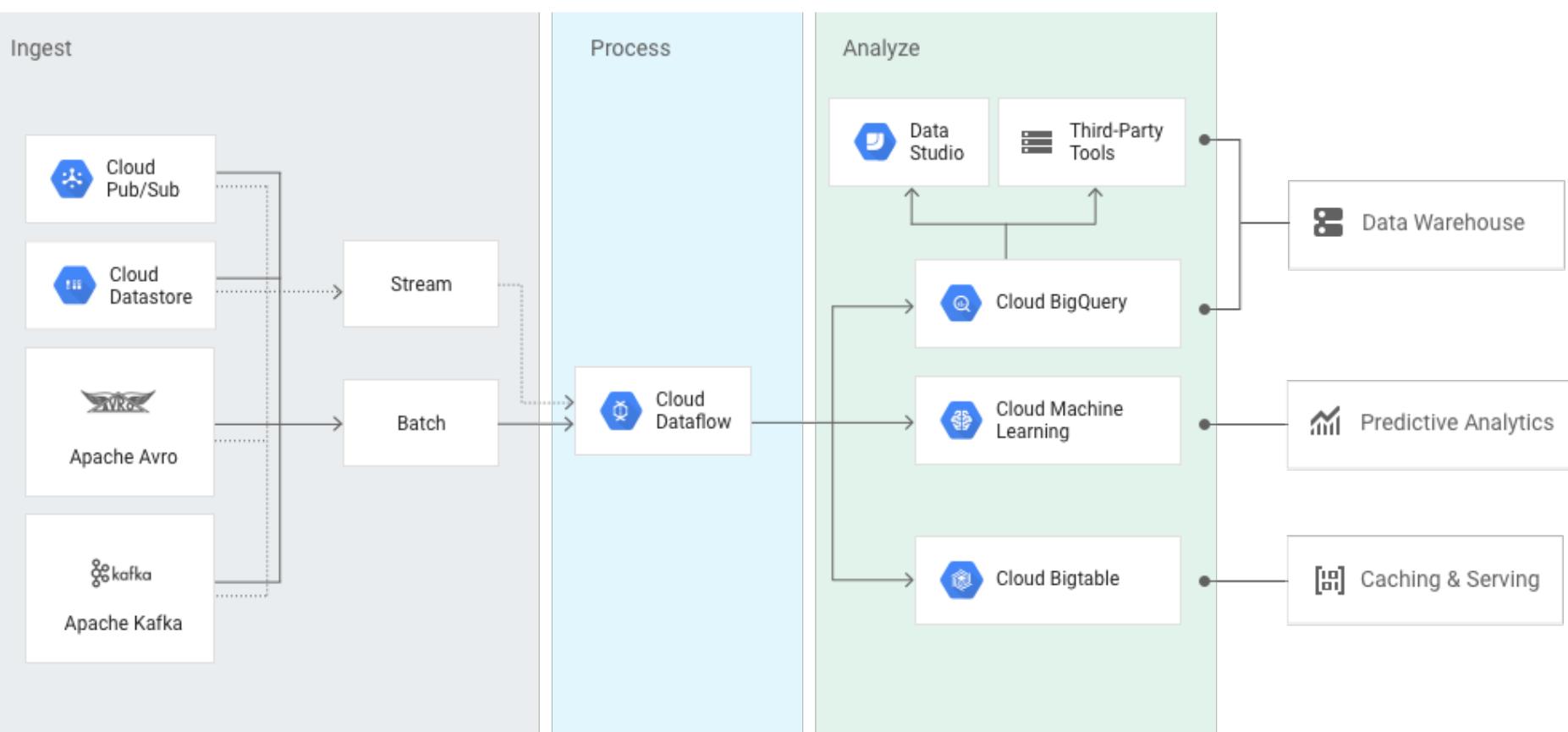
[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)[Next](#)

## Cloud Dataflow Overview

### What is it?

- Auto-scaling, No-Ops, Stream, and Batch Processing
- Built on Apache Beam:
  - Documentation refers to Apache Beam site
  - Configuration is 100% code-based
- Integrates with other tools (GCP and external):
  - Natively - Pub/Sub, BigQuery, Cloud ML Engine
  - Connectors - Bigtable, Apache Kafka
- Pipelines are regional-based

### Big Picture - Data Transformation



[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## *Cloud Dataflow Overview*

[Previous](#)[Next](#)

### IAM:

- Project-level only - all pipelines in the project (or none)
- Pipeline data access separate from pipeline access
- Dataflow Admin - Full pipeline access plus machine type/storage bucket config access
- Dataflow Developer - Full pipeline access, no machine type/storage bucket access
- Dataflow Viewer - view permissions only
- Dataflow Worker - Specifically for service accounts

### Dataflow vs Dataproc?

### Beam vs. Hadoop/Spark?

#### Dataproc:

- Familiar tools/packages
- Employee skill sets
- Existing pipelines

#### Dataflow:

- Less Overhead
- Unified batch and stream processing
- Pipeline portability across Dataflow, Spark, and Flink as runtimes

WORKLOADS	CLOUD DATAPROC	CLOUD DATAFLOW
Stream processing (ETL)		X
Batch processing (ETL)	X	X
Iterative processing and notebooks	X	
Machine learning with Spark ML	X	
Preprocessing for machine learning		X (with Cloud ML Engine)

[Return to Table of Contents](#)

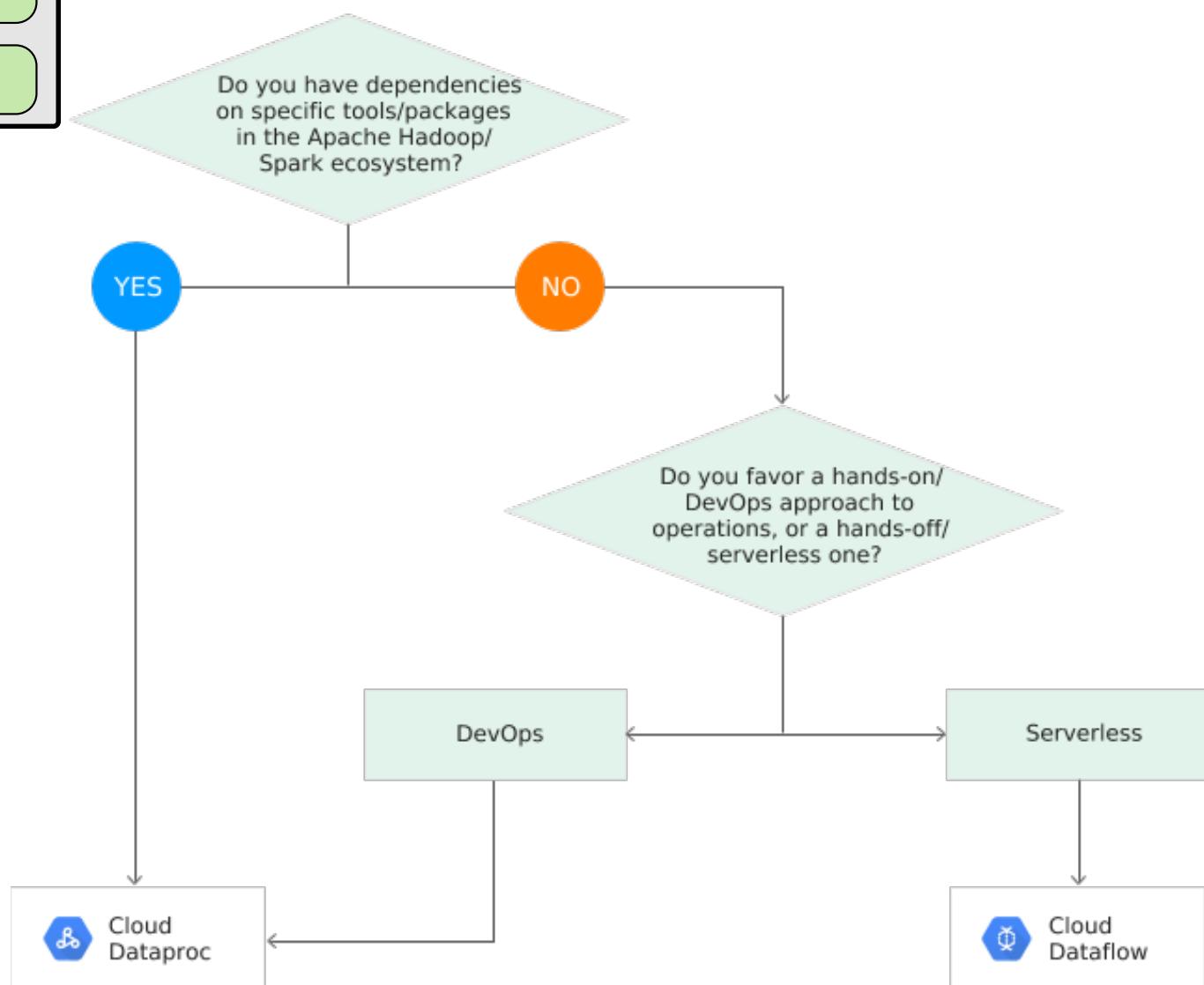
## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## Cloud Dataflow Overview

[Previous](#)

### Dataflow vs. Dataproc decision tree



[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## Key Concepts

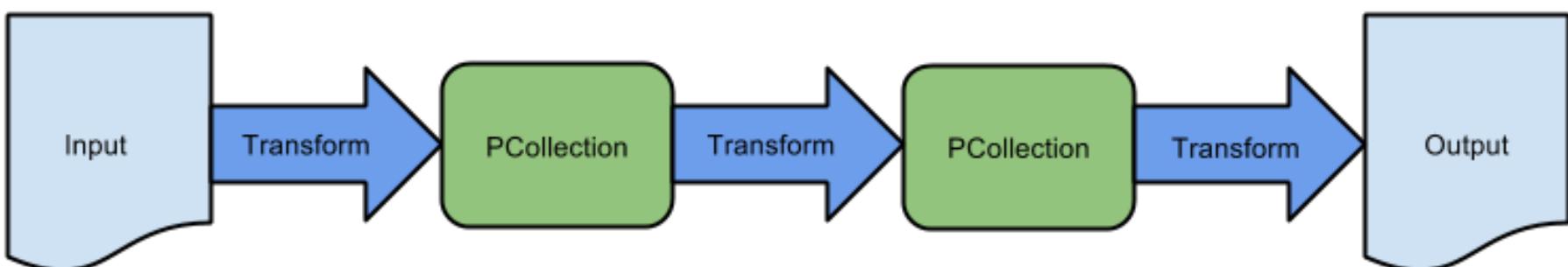
### Course/exam perspective:

[Next](#)

- Dataflow is very code-heavy
- Exam does not go deep into coding questions
- Some key concepts/terminology will be tested

### Key terms:

- Element - single entry of data (e.g., table row)
- PCollection - Distributed data set, data input and output
- Transform - Data processing operation (or step) in pipeline:
  - Uses programming conditionals (for/while loops, etc.)
- ParDo - Type of transform applied to individual elements:
  - Filter out/extract elements from a large group of data



**PCollection and ParDo in example Java code.**

**One step in a multi-step transformation process.**

```

PCollection<LaneInfo> currentConditions = p //
    .apply("GetMessages", PubsubIO.readStrings().fromTopic(topic)) //
    .apply("ExtractData", ParDo.of(new DoFn<String, LaneInfo>() {
        @ProcessElement
        public void processElement(ProcessContext c) throws Exception {
            String line = c.element();
            c.output(LaneInfo.newLineInfo(line));
        }
    }));
  
```

[Return to Table of Contents](#)

## Choose a Lesson

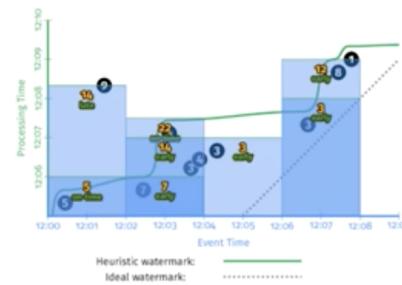
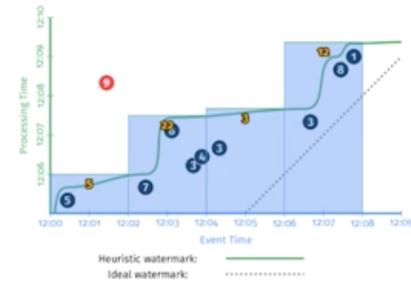
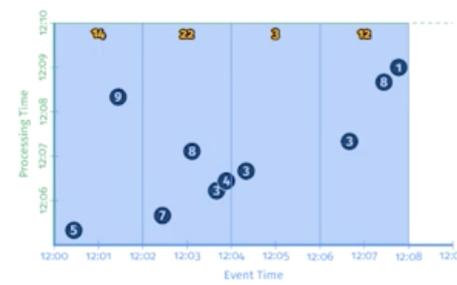
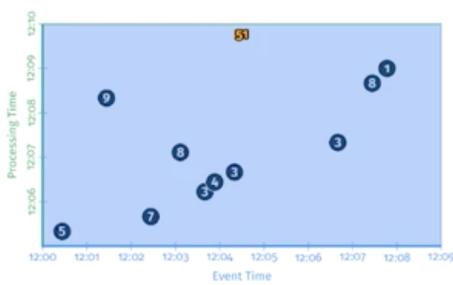
[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## **Cloud Dataflow Overview**

[Previous](#)

### Dealing with late/out of order data:

- Latency is to be expected (network latency, processing time, etc.)
- Pub/Sub 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 the processing 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

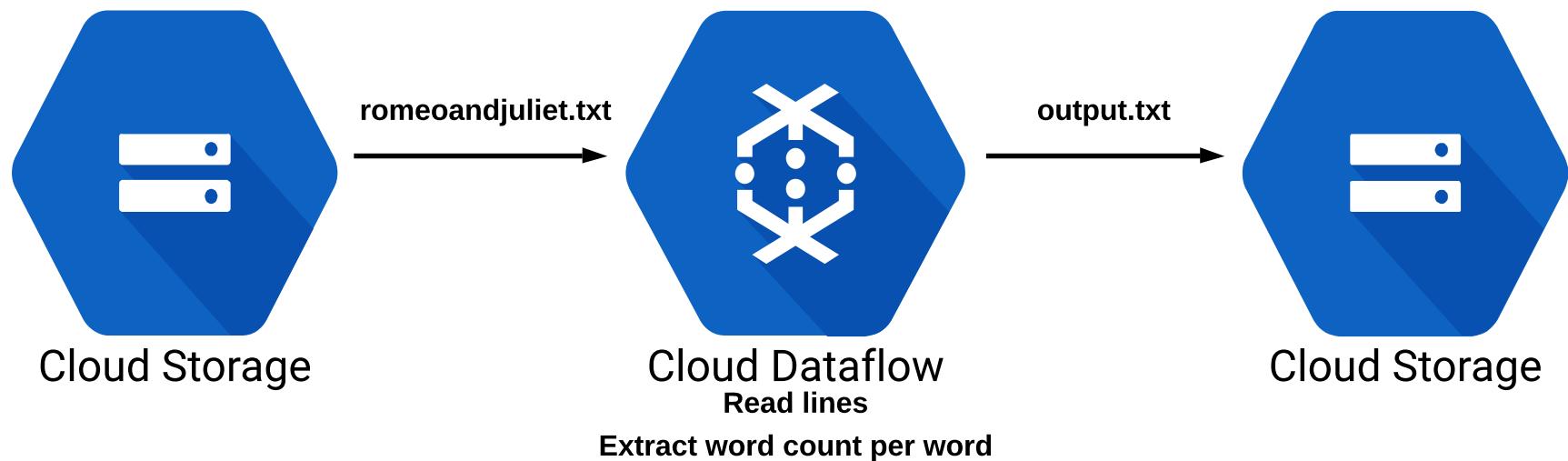


[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

- Google-provided templates
- Simple word count extraction



[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

Traffic data

## ***Streaming Ingest Pipeline Hands On***

- Take San Diego traffic data
- Ingest through Pub/Sub
- Process with Dataflow
- Analyze results with BigQuery
- First: Enable Dataflow API from API's and Services

[Next](#)

Data ingest

Published  
streaming  
sensor data

Topic: sandiego

Subscription  
pulls messagesCloud Dataflow  
Transform data  
to calculate  
average speed.Output to  
BigQuery.

BigQuery

[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## ***Streaming Ingest Pipeline Hands On***

[Previous](#)[Next](#)

### Quick command line setup (Cloud Shell)

- Create BigQuery dataset for processing pipeline output:
  - bq mk --dataset \$DEVSHELL\_PROJECT\_ID:demos
- Create Cloud Storage bucket for Dataflow staging:
  - gsutil mb gs://\$DEVSHELL\_PROJECT\_ID
- Create Pub/Sub topic and stream data:
  - cd ~/googledataengineer/courses/streaming/publish
  - gcloud pubsub topics create sandiego
  - ./download\_data.sh
  - sudo pip install -U google-cloud-pubsub
  - ./send\_sensor\_data.py --speedFactor=60  
--project=\$DEVSHELL\_PROJECT\_ID

### Open a new Cloud Shell tab:

- Execute Dataflow pipeline for calculating average speed:
  - cd ~/googledataengineer/courses/streaming/process/sandiego
  - ./run\_oncloud.sh \$DEVSHELL\_PROJECT\_ID \$DEVSHELL\_PROJECT\_ID AverageSpeeds
- Error resolution:
  - Pub/Sub permission denied, re-authenticate
    - gcloud auth application-default login
  - Dataflow workflow failed - enable Dataflow API

[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## *Streaming Ingest Pipeline Hands On*

[Previous](#)

### **View results in BigQuery:**

- List first 100 rows:
  - `SELECT * FROM [<PROJECTID>:demos.average_speeds]  
ORDER BY timestamp DESC LIMIT 100`
- Show last update to table:
  - `SELECT MAX(timestamp) FROM  
[<PROJECTID>:demos.average_speeds]`
- Look at results from the last minute:
  - `SELECT * FROM  
[<PROJECTID>:demos.average_speeds@-60000] ORDER BY  
timestamp DESC`

### **Shut down pipeline:**

- Drain - finishing processing buffered jobs before shutting down
- Cancel - full stop, cancels existing buffered jobs

[Return to Table of Contents](#)

### Choose a Lesson

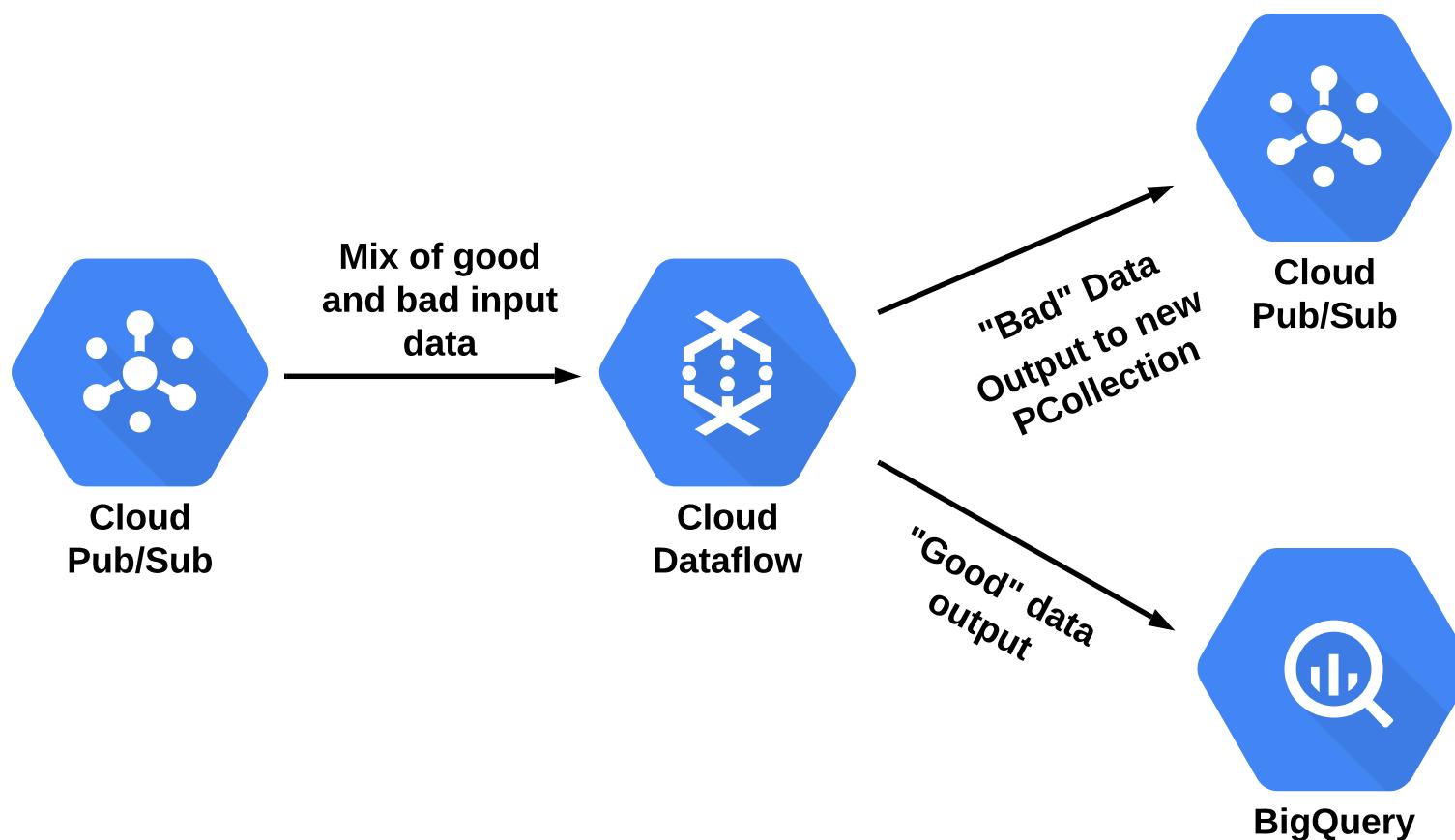
[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## Additional Best Practices

[Next](#)

### Handling Pipeline Errors

- If you do not have a mechanism in place to handle input data errors in your pipeline, the job can fail. How can we account for this?
- Gracefully catch errors:
  - Create separate output:
    - **Try-catch** block handles errors
    - Output errors to new **PCollection** - Send to **collector** for later analysis (Pub/Sub is a good target)
    - Think of it as *recycling* the *bad* data
- Technique is also valid for troubleshooting missing messages:
  - Scenario: Streaming pipeline missing some messages
  - Solution: Run a batch of the streaming data, and check output:
    - Create additional output to capturing and processing error data.



[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

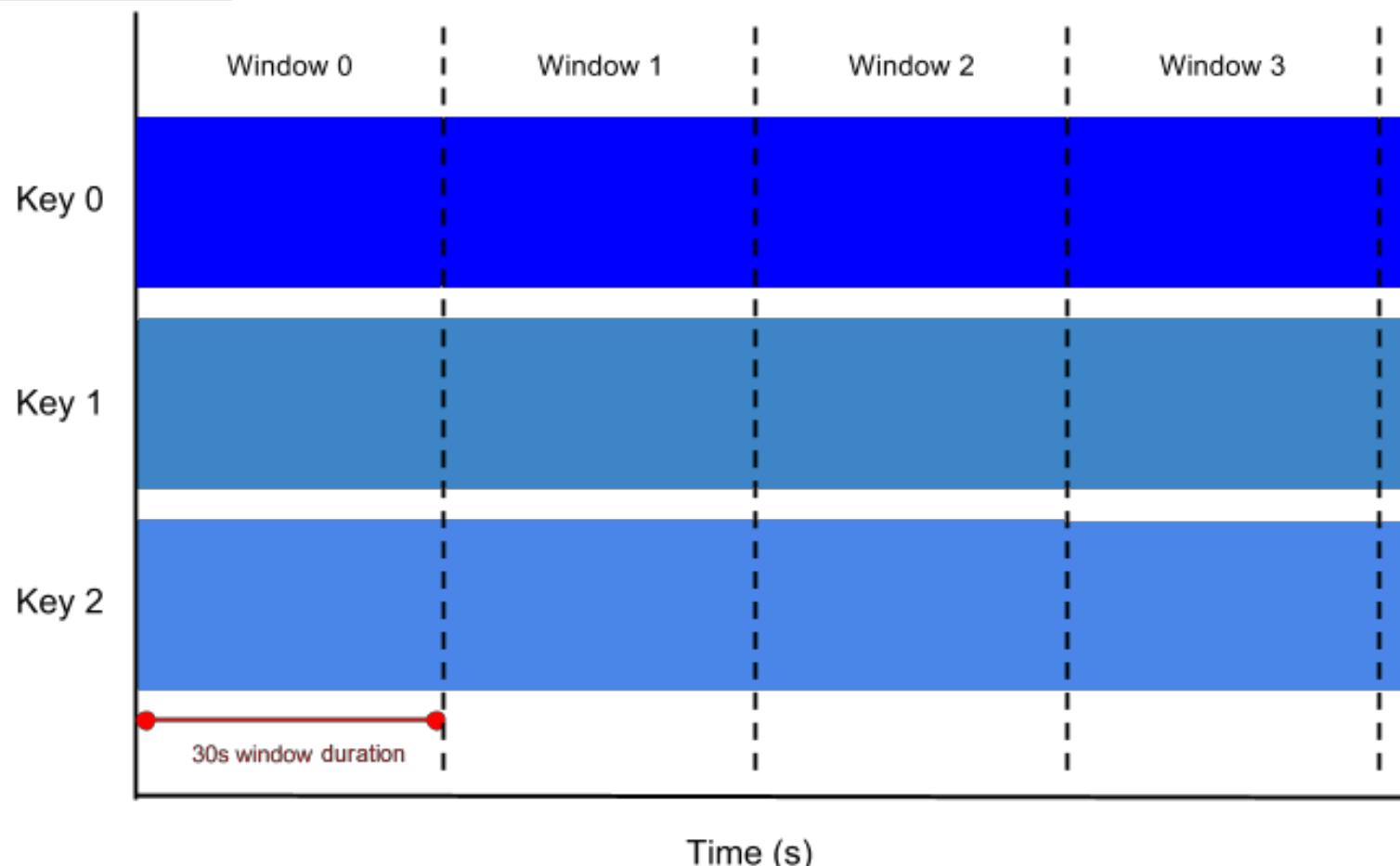
## Additional Best Practices

[Previous](#)[Next](#)

### Know your window types

- **Global, Fixed, Sliding, Session**
- **Global** - The default, uses a single window for entire pipeline
- **Fixed time** - Every (x) period of time
  - Every 5 seconds, 10 minutes, etc.
- **Sliding time** - Overlapping time windows
- **Session** - Within certain time of certain elements:
  - For example, *Time since last user/mouse activity*

#### Fixed time Window



[Return to Table of Contents](#)

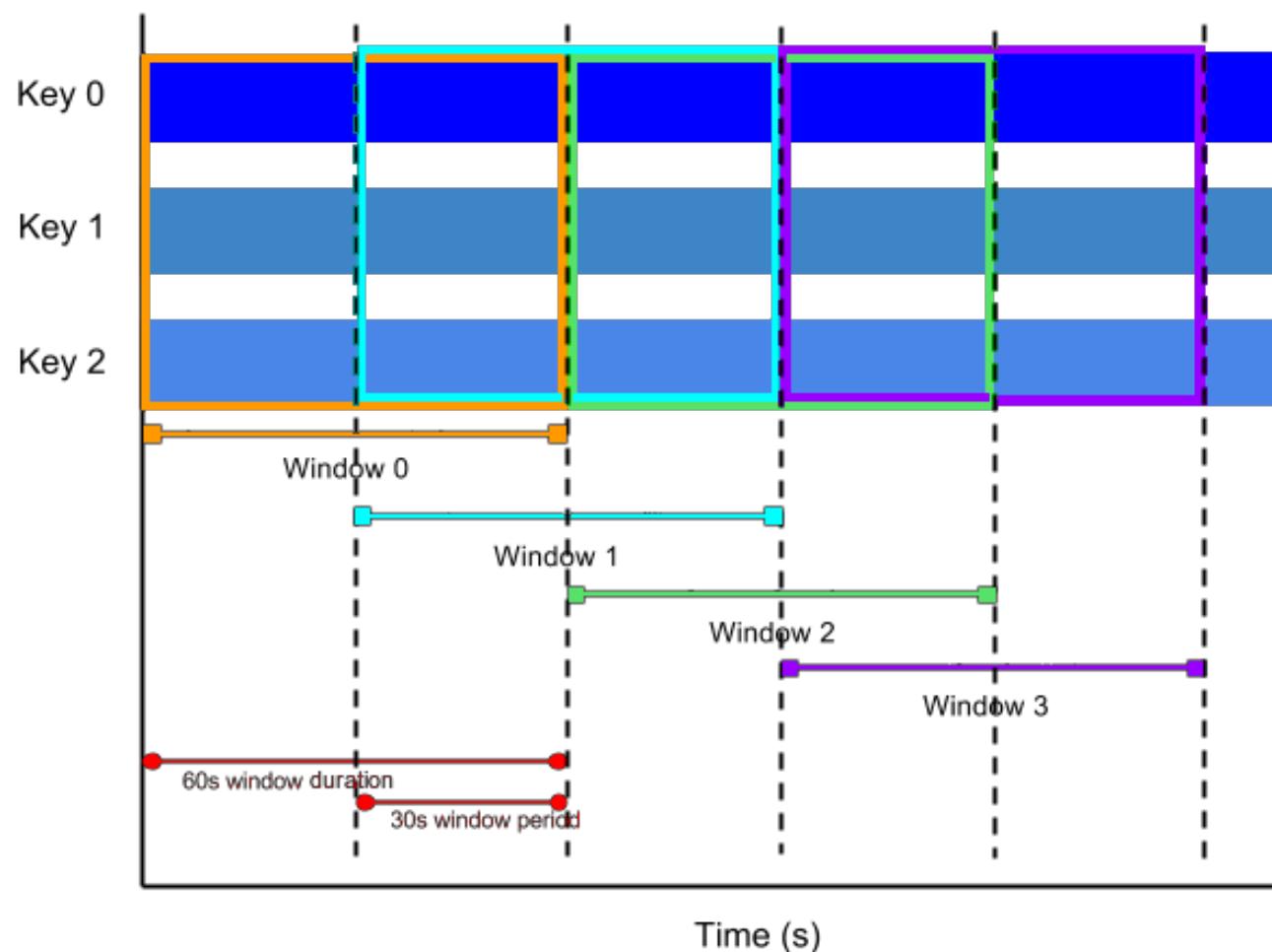
## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## Additional Best Practices

[Previous](#)[Next](#)

### Sliding Time Window



[Return to Table of Contents](#)

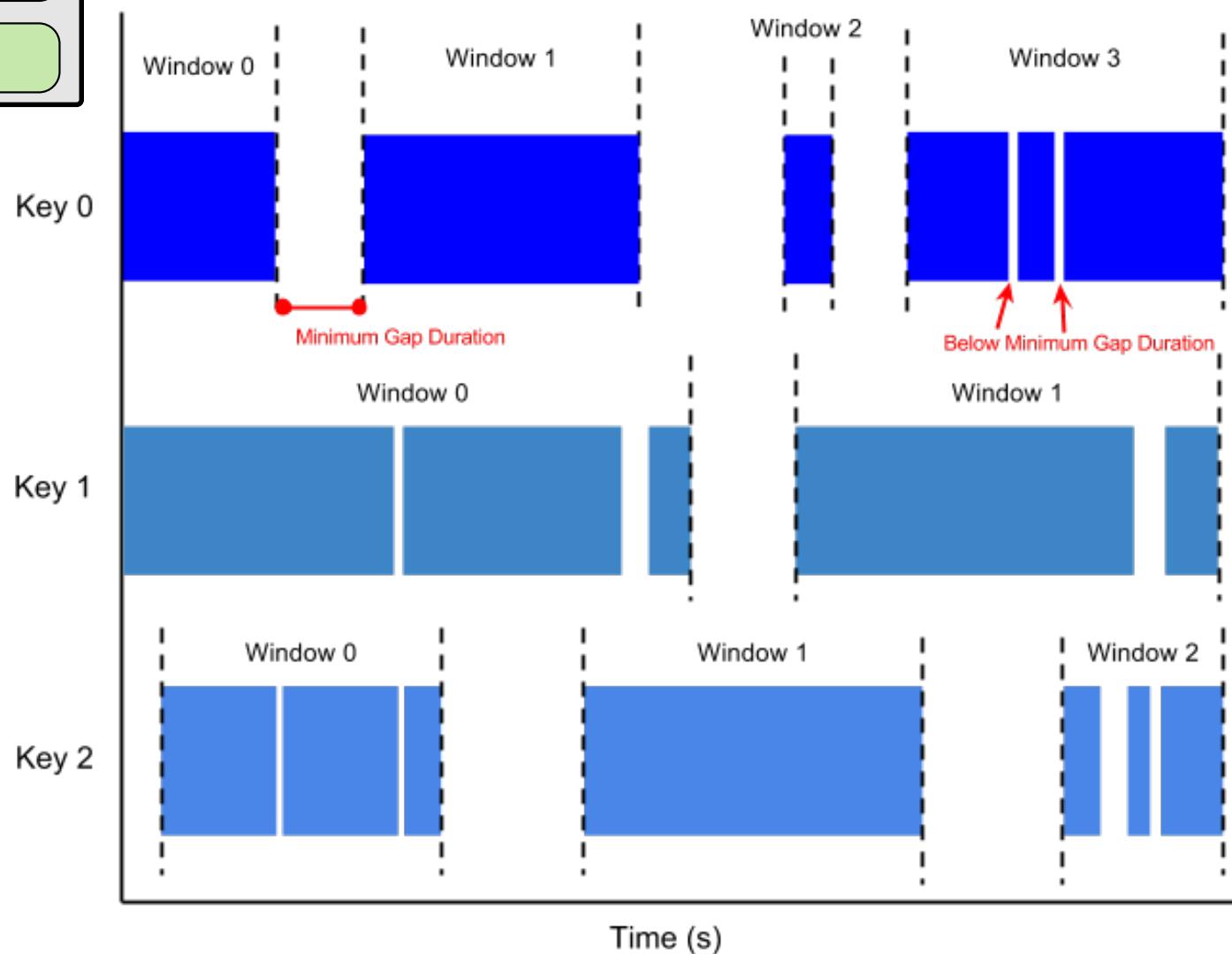
## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## Additional Best Practices

[Previous](#)[Next](#)

### Session Window



[Return to Table of Contents](#)

## Choose a Lesson

[Data Processing Challenges](#)[Cloud Dataflow Overview](#)[Key Concepts](#)[Template Hands On](#)[Streaming Ingest Pipeline Hands On](#)[Additional Best Practices](#)

## ***Additional Best Practices***

[Previous](#)

### Updating Dataflow Pipelines

- **Scenario:** Update streaming Dataflow pipeline with new code:
  - New code = new pipeline not compatible with current version
  - Need data to *switch over* to new job/pipeline without losing anything in the process
- **Solution:** Update job:
  - Creates new job with same name/new *jobID*
- Compatibility between old/new jobs:
  - Map old to new job transforms with **transform mapping**
    - "Bridge" between old and new code base
  - After compatibility check:
    - Buffered data transferred to new job, using transform mapping to translate changes

[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## Managed Hadoop/Spark Stack

Custom Code

Monitoring/Health

Dev Integration

Manual Scaling

Job Submission

Google Cloud Connectivity

Deployment

Creation

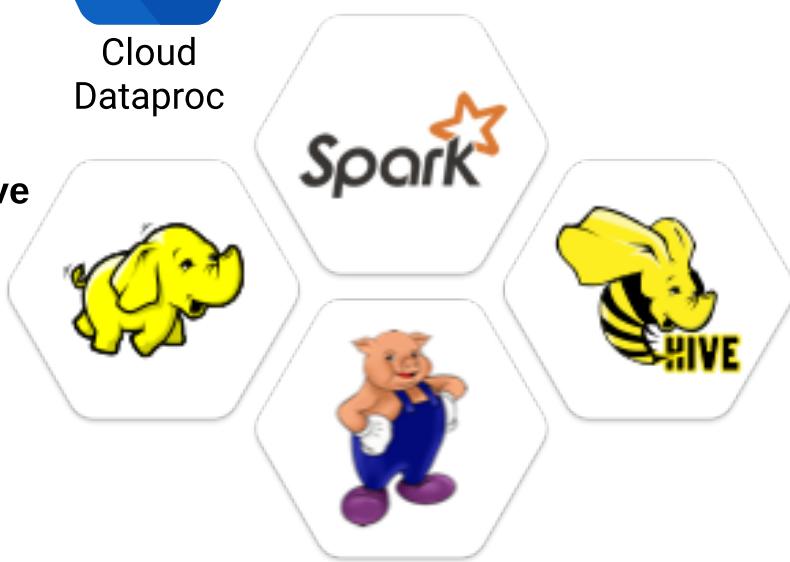
## Dataproc Overview

### What is Cloud Dataproc?

[Next](#)

### Hadoop ecosystem:

- Hadoop, Spark, Pig, Hive
- Lift and shift to GCP



### Dataproc facts:

- On-demand, managed Hadoop and Spark clusters
- Managed, but not no-ops:
  - Must configure cluster, not auto-scaling
  - Greatly reduces administrative overhead
- Integrates with other Google Cloud services:
  - Separate data from the cluster - save costs
- Familiar Hadoop/Spark ecosystem environment:
  - Easy to move existing projects
- Based on Apache Bigtop distribution:
  - Hadoop, Spark, Hive, Pig
- HDFS available (but maybe not optimal)
- Other ecosystem tools can be installed as well via initialization actions

[Return to Table of Contents](#)

## Choose a Lesson

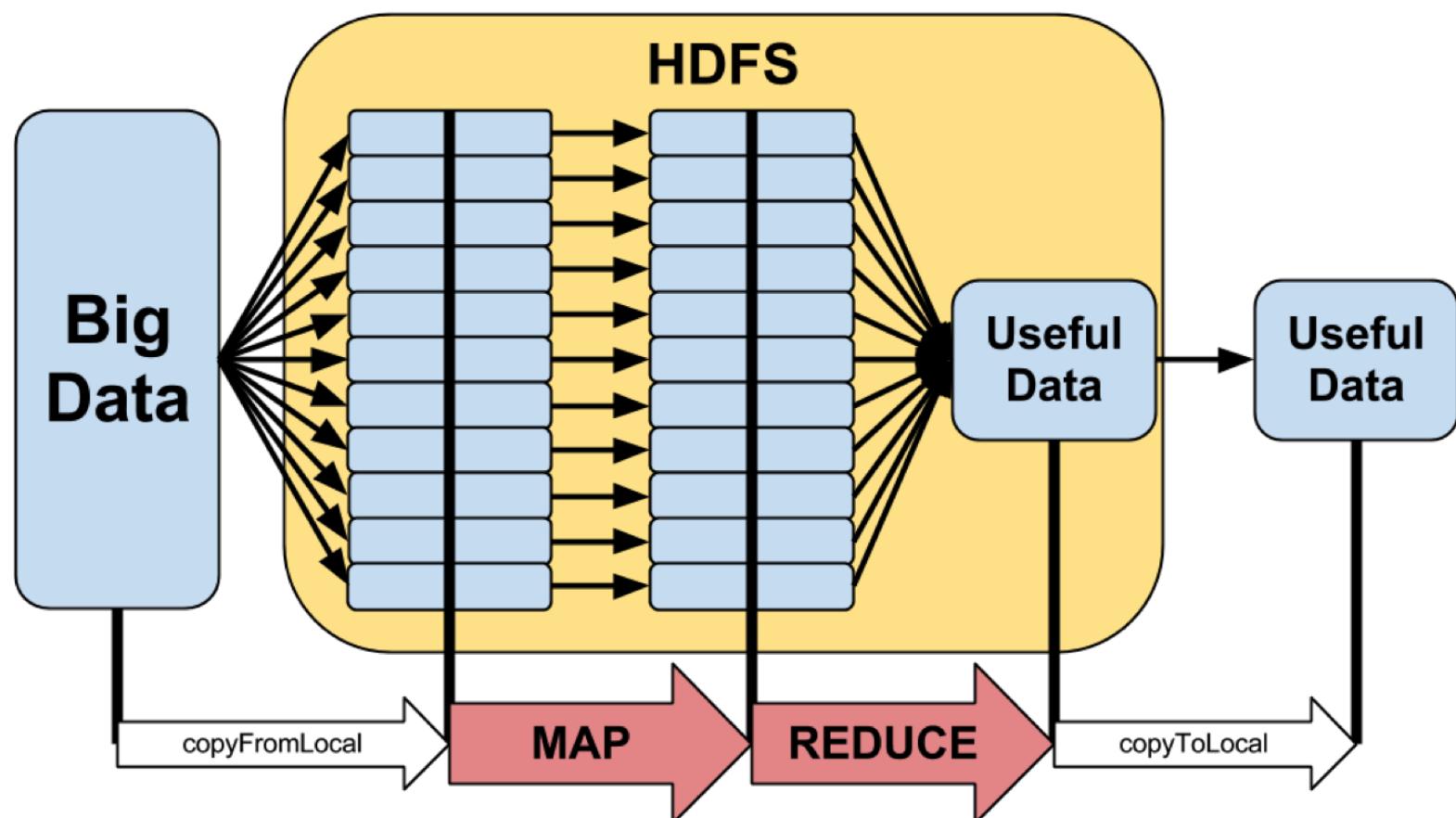
[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## Dataproc Overview

[Previous](#)[Next](#)

### What is MapReduce?

- Simple definition:
  - Take big data, distribute it to many workers (map)
  - Combine results of many pieces (reduce)
- Distributed/parallel computing



[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## Dataproc Overview

[Previous](#)[Next](#)

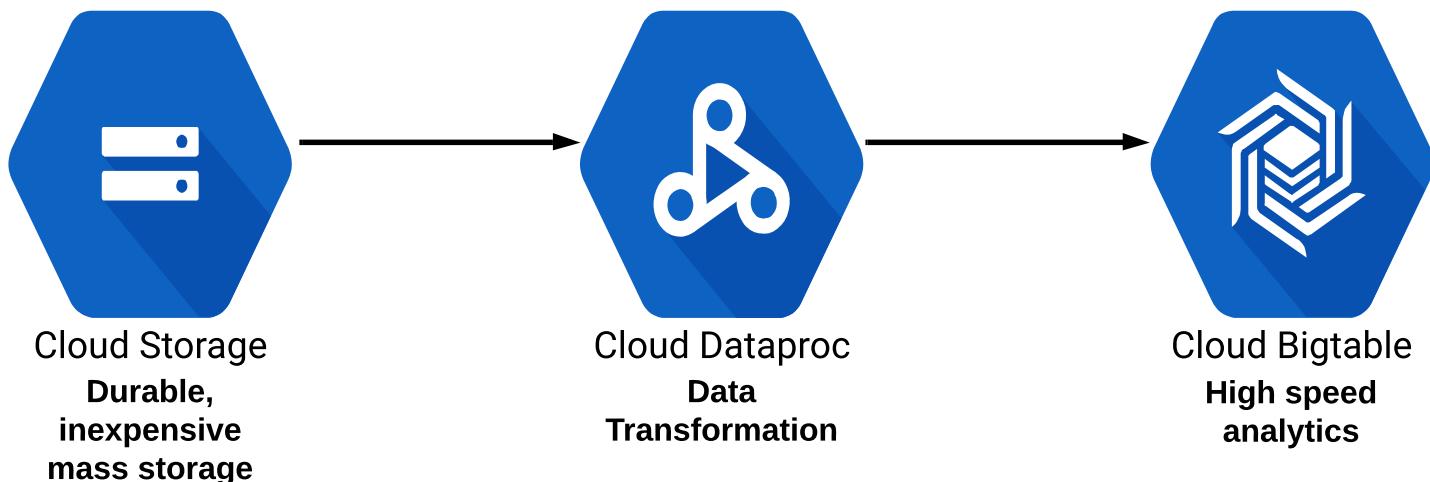
### Pricing:

- Standard Compute Engine machine type pricing + managed Dataproc premium
- Premium = \$0.01 per vCPU core/hour

Machine type	Virtual CPUs	Memory	Dataproc
n1-highcpu-2	2	1.80GB	\$0.020
n1-highcpu-4	4	3.60GB	\$0.040
n1-highcpu-8	8	7.20GB	\$0.080
n1-highcpu-16	16	14.40GB	\$0.160
n1-highcpu-32	32	28.80GB	\$0.320
n1-highcpu-64	64	57.60GB	\$0.640

### Data Lifecycle Scenario

#### Data Ingest, Transformation, and Analysis



[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## Dataproc Overview

[Previous](#)

### **Identity and Access Management (IAM):**

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

[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## Configure Dataproc Cluster

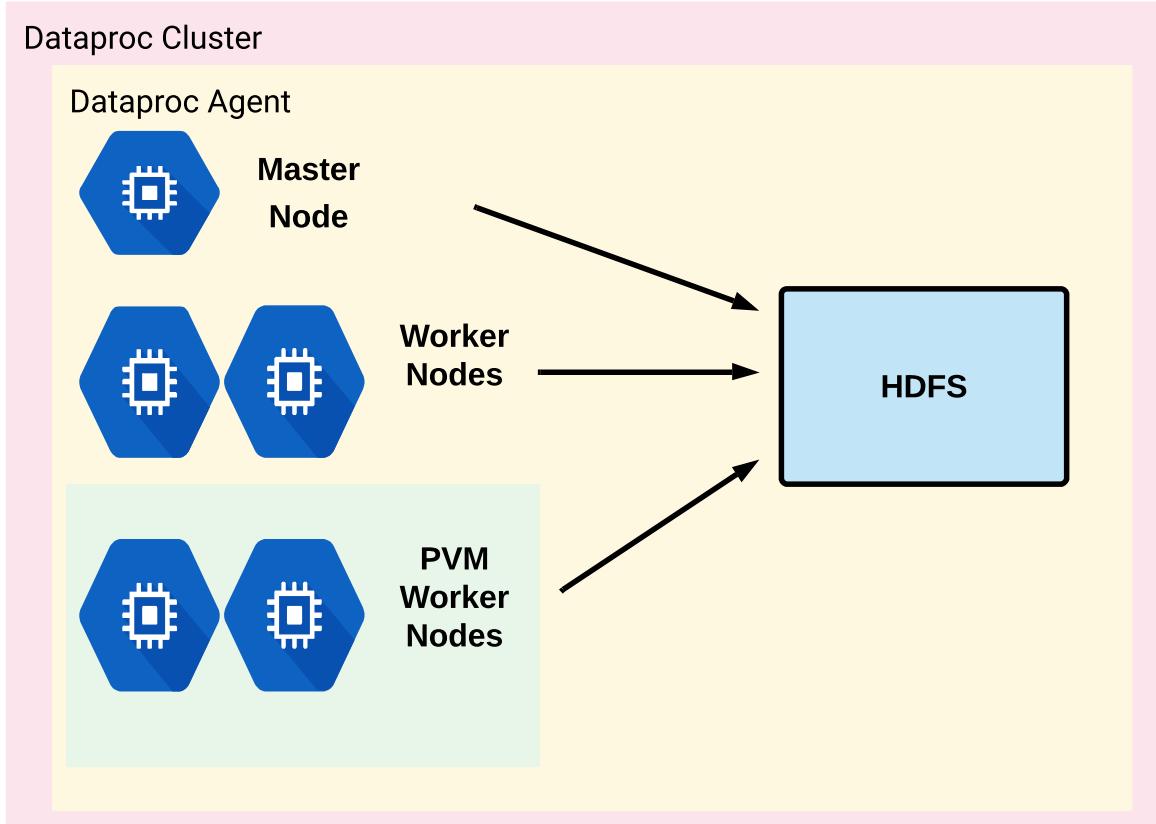
[Next](#)

### Create cluster:

- `gcloud dataproc clusters create [cluster_name] --zone [zone_name]`
- **Configure master node, worker nodes:**
  - Master contains YARN resource manager
  - YARN = Yet Another Resource Negotiator

### Updating clusters:

- Can only change # workers/preemptible VM's/labels/toggle graceful decommission
- Automatically reshards data for you
- `gcloud dataproc clusters update [cluster_name] --num-workers [#] --num-preemptible-workers [#]`



[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Previous](#)

## Preemptible VM's on Dataproc:

- Excellent low-cost worker nodes
- Dataproc manages the entire leave/join process:
  - No need to configure startup/shutdown scripts
  - Just add PVM's...and that's it
- No assigned disks for HDFS (only disk for caching)
- Want a mix of standard + PVM worker nodes

## Access your cluster:

- SSH into master - same as any compute engine instance
- `gcloud compute ssh [master_node_name]`

## Access via web - 2 options:

- Open firewall ports to your network (8088, 9870)
- Use SOCKS proxy - does not expose firewall ports

## SOCKS proxy configuration:

- SSH to master to enable port forwarding:
  - `gcloud compute ssh master-host-name --project=project-id --zone=master-host-zone -- -D 1080 -N`
- Open new terminal window - launch web browser with parameters (varies by OS/browser):
  - `"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome"`
  - `--proxy-server="socks5://localhost:1080" --host-resolver-rules="MAP * 0.0.0.0 , EXCLUDE localhost"`
  - `--user-data-dir=/tmp/cluster1-m`
- Browse to `http://[master]:port`:
  - 8088 - Hadoop
  - 9870 - HDFS

## Using Cloud Shell (must use for each port):

- `gcloud compute ssh master-host-name --project=project-id --zone master-host-zone -- -4 -N -L port1:master-host-name:port2`
- Use Web Preview to choose port (8088/9870)

[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

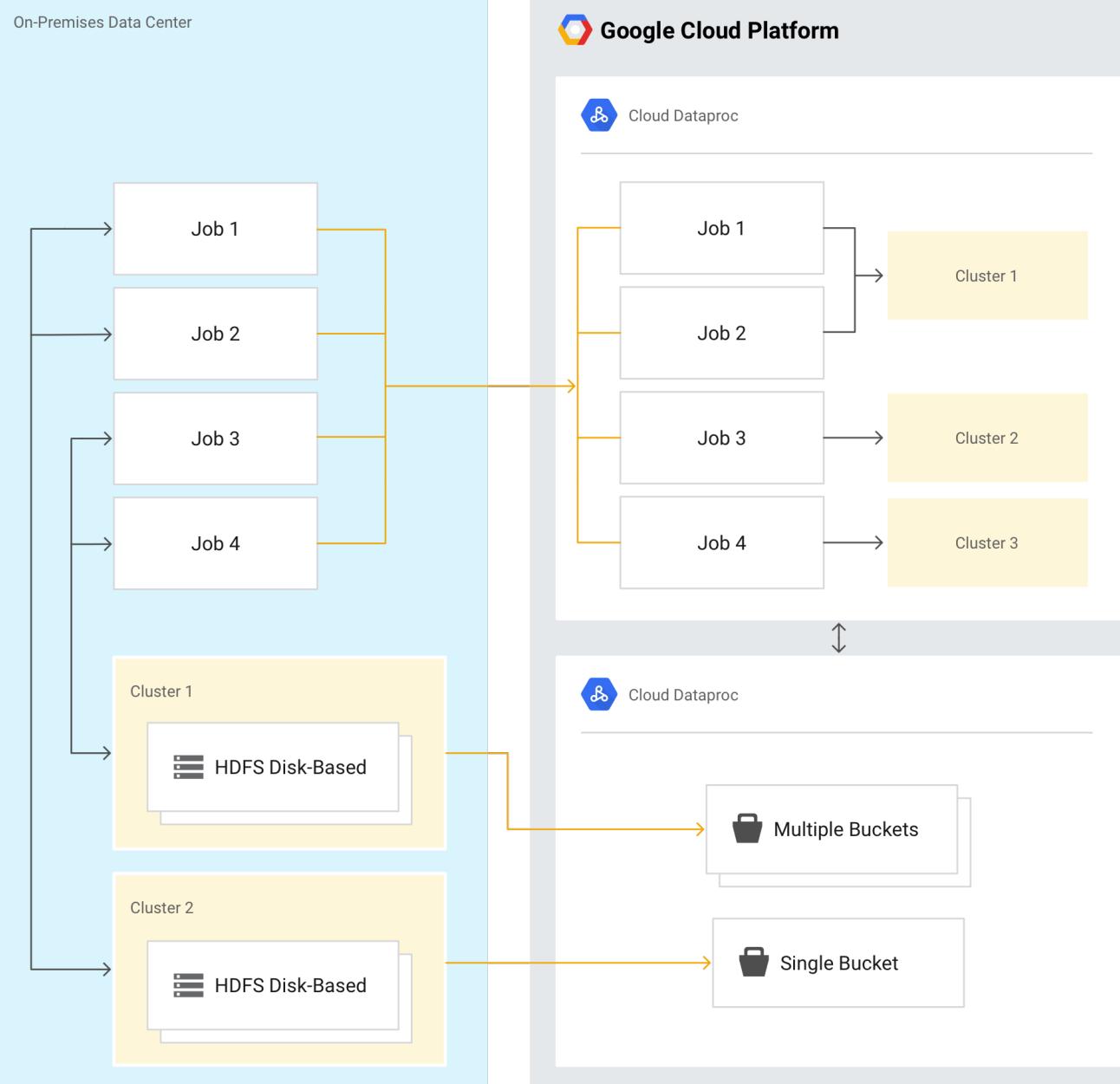
## Migrating and Optimizing for Google Cloud

### Migrating to Cloud Dataproc

[Next](#)

#### What are we moving/optimizing?

- Data (from HDFS)
- Jobs (pointing to Google Cloud locations)
- Treating clusters as ephemeral (temporary) rather than permanent entities



Install Cloud Storage connector to connect to GCS (Google Cloud Storage).

[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## *Migrating and Optimizing for Google Cloud*

[Previous](#)[Next](#)

### Migration Best Practices:

- Move data first (generally Cloud Storage buckets):
  - Possible exceptions:
    - Apache HBase data to Bigtable
    - Apache Impala to BigQuery
    - Can still choose to move to GCS if Bigtable/BQ features not needed
- Small-scale experimentation (proof of concept):
  - Use a subset of data to test
- Think of it in terms of ephemeral clusters
- Use GCP tools to optimize and save costs

[Return to Table of Contents](#)

## Migrating and Optimizing for Google Cloud

### Choose a Lesson

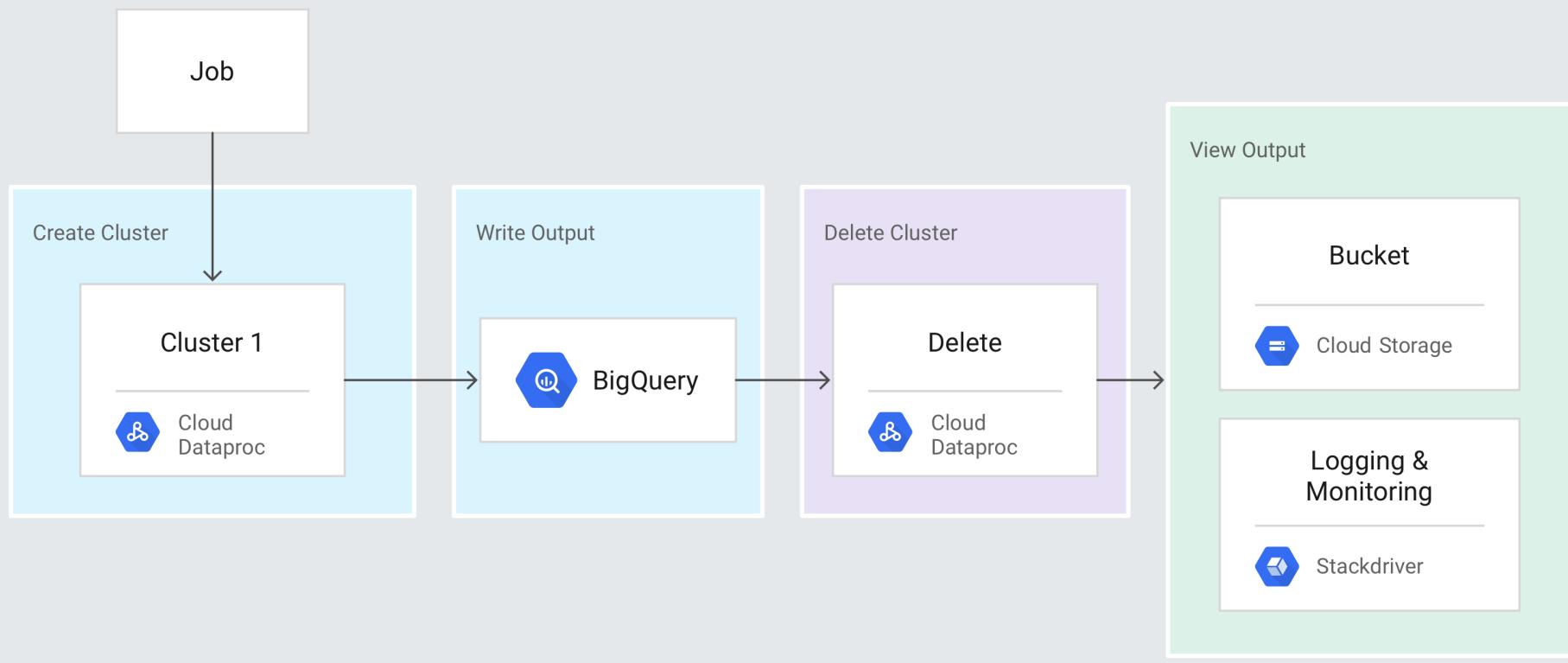
[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)[Previous](#)[Next](#)

### Optimize for the Cloud ("Lift and Leverage")

#### Separate storage and compute (cluster):

- **Save on costs:**
  - No need to keep clusters to keep/access data
- **Simplify workloads:**
  - No shaping workloads to fit hardware
  - Simplify storage capacity
- **HDFS --> Google Cloud Storage**
- **Hive --> BigQuery**
- **HBase --> Bigtable**

### Google Cloud Platform



[Return to Table of Contents](#)

## Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

## *Migrating and Optimizing for Google Cloud*

[Previous](#)

### Converting from HDFS to Google Cloud Storage:

#### 1. Copy data to GCS:

- Install connector or copy manually

#### 2. Update file prefix in scripts:

- From `hdfs://` to `gs://`

#### 3. Use Dataproc, and run against/output to GCS

The end goal should be to eventually move toward a cloud-native and serverless architecture (Dataflow, BigQuery, etc.).

[Return to Table of Contents](#)

### Choose a Lesson

[Dataproc Overview](#)[Configure Dataproc Cluster and Submit Job](#)[Migrating and Optimizing for Google Cloud](#)[Best Practices for Cluster Performance](#)

# The Data Dossier

## ***Best Practices for Cluster Performance***

### Dataproc Performance Optimization

(GCP-specific)

- Keep data close to your cluster
  - Place Dataproc cluster in the same region as storage bucket
- Larger persistent disk = better performance
  - Consider using SSD over HDD – slightly higher cost
- Allocate more VM's
  - Use preemptible VM's to save on costs
  - More VM's will come at a higher cost than larger disks if more disk throughput is needed

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## BigQuery Overview

[Next](#)

### What is BigQuery?

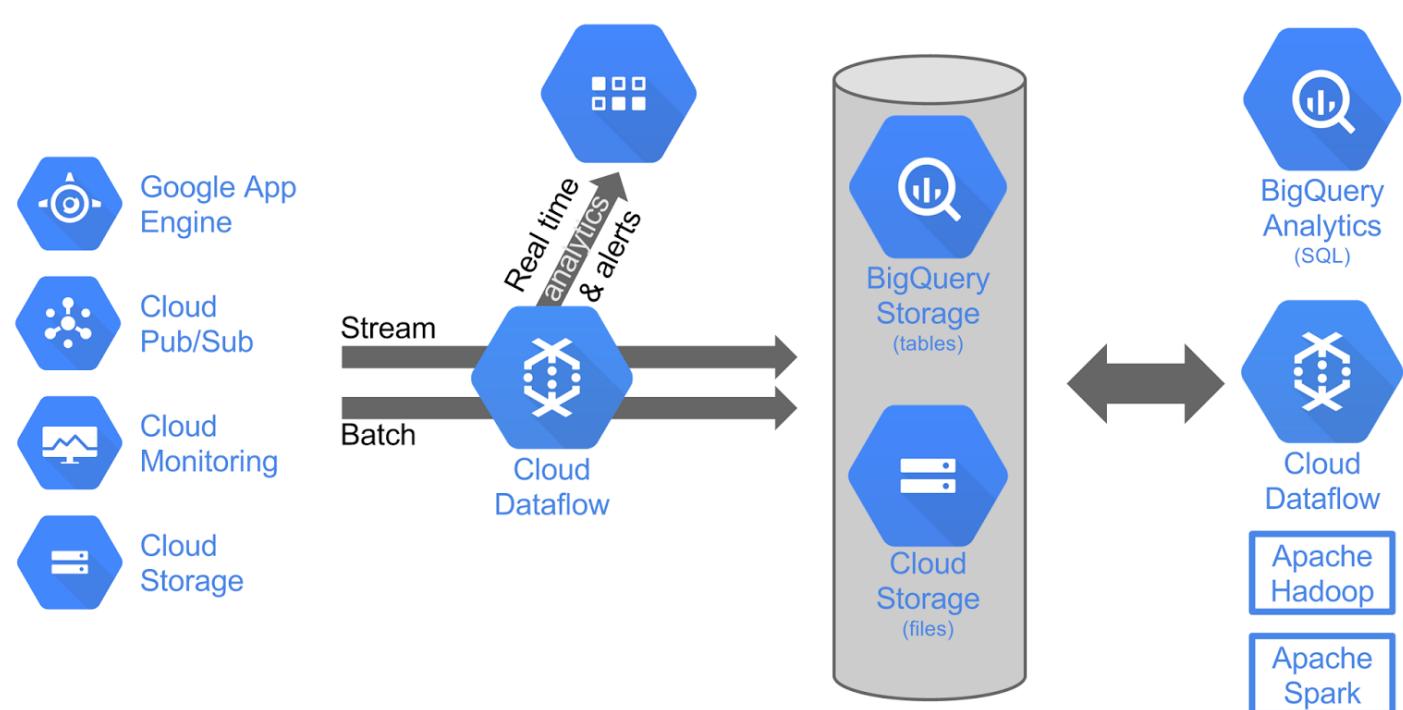
- Fully Managed Data warehousing
  - Near-real time analysis of petabyte scale databases
- Serverless (no-ops)
- Auto-scaling to petabyte range
- Both storage and analysis
- Accepts batch and streaming loads
- Locations = multi-regional (US, EU), Regional (asia-northeast1)
- Replicated, durable
- Interact primarily with standard SQL (also Legacy SQL)
  - [SQL Primer course](#)

Ingest

Process

Store

Analyze



[Return to Table of Contents](#)

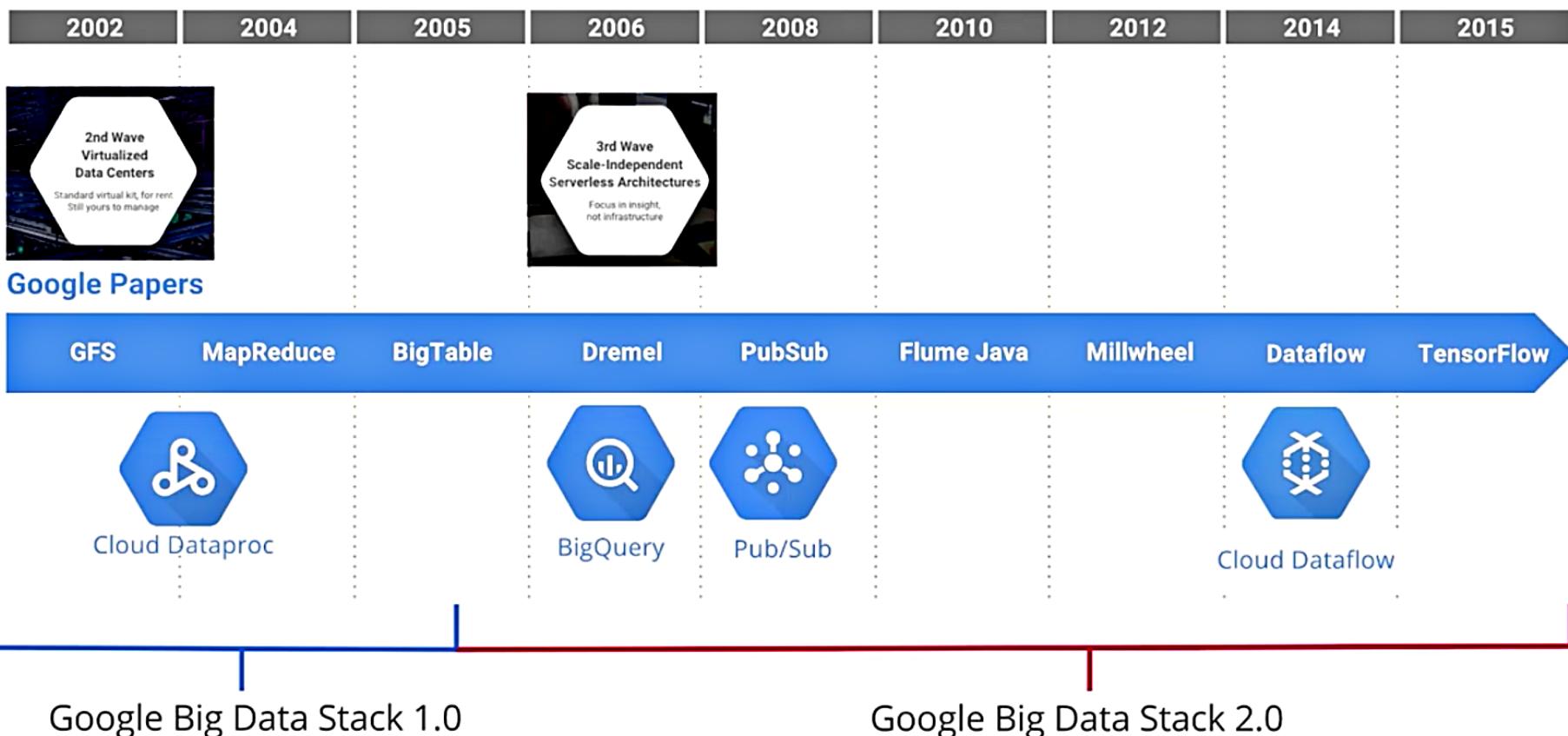
## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)[Previous](#)[Next](#)

## BigQuery Overview

### How BigQuery works

- Part of the "3rd wave" of cloud computing
  - Google Big Data Stack 2.0
- Focus on serverless compute, real time insights, machine learning...
  - ...instead of data placement, cluster configuration
  - No managing of infrastructure, nodes, clusters, etc



[Return to Table of Contents](#)

## Choose a Lesson

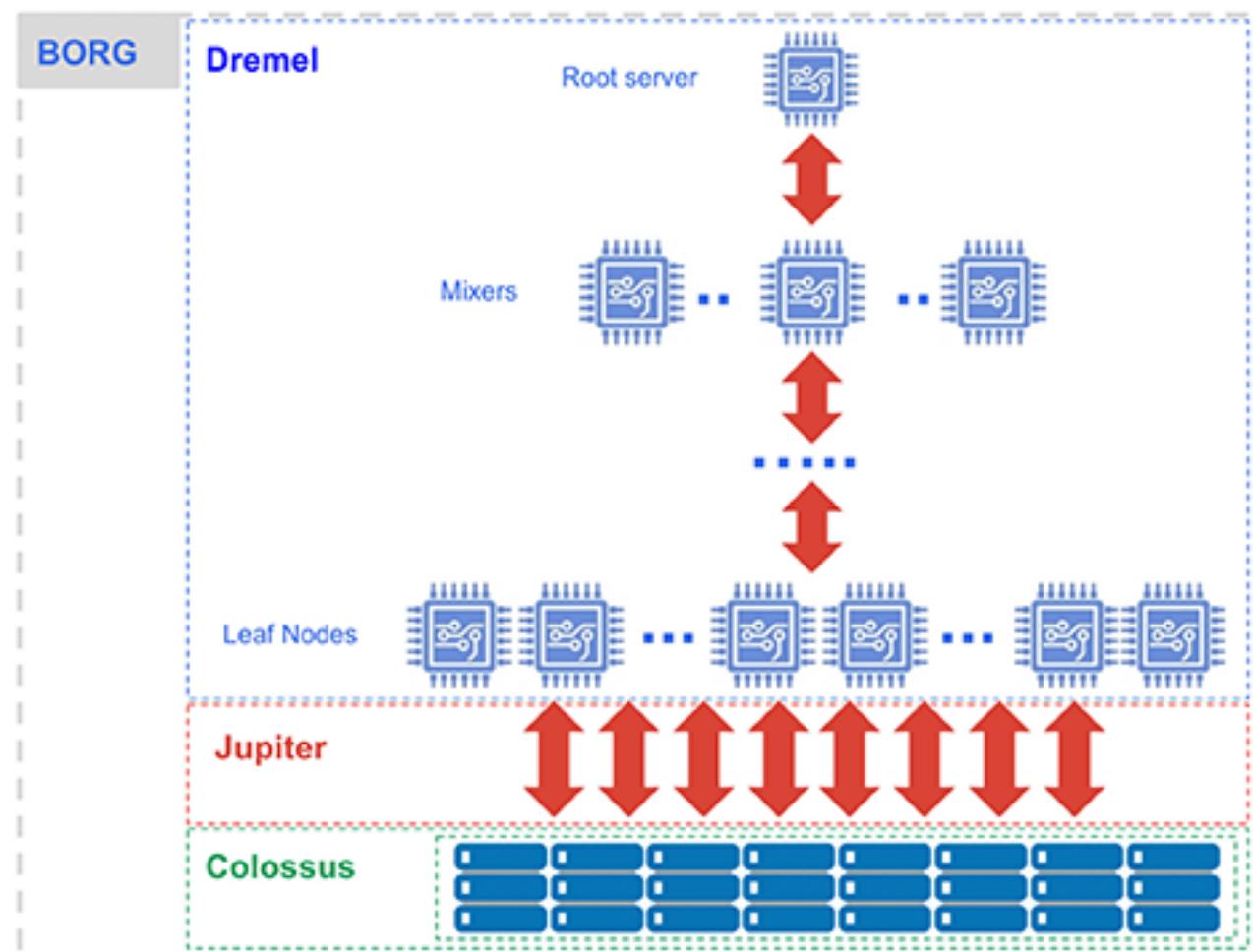
[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## BigQuery Overview

[Previous](#)[Next](#)

### How BigQuery works (cont)

- 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



[Return to Table of Contents](#)

## Choose a Lesson

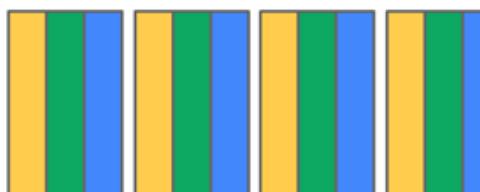
[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## BigQuery Overview

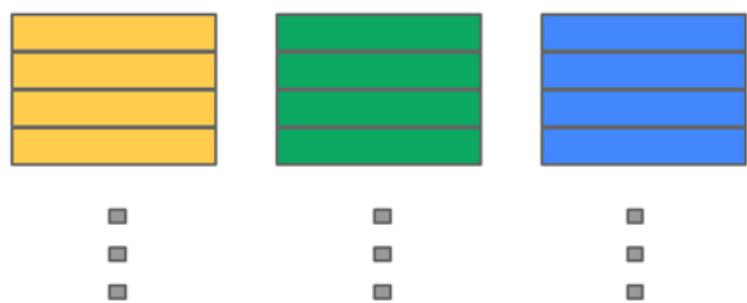
[Previous](#)[Next](#)

### How BigQuery works (cont)

- Columnar data store
  - Separates records into column values, stores each value on different storage volume
  - Traditional RDBMS stores whole record on one volume
  - Extremely fast read performance, poor write (update) performance - BigQuery does not update existing records
  - Not transactional



...



Record Oriented Storage

Column Oriented Storage

[Return to Table of Contents](#)

## Choose a Lesson

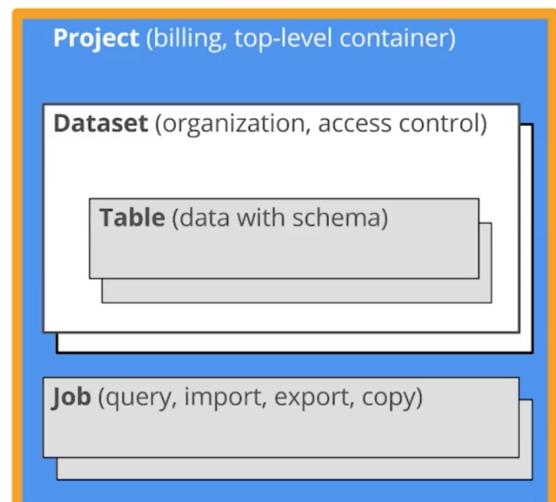
[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *BigQuery Overview*

[Previous](#)[Next](#)

### BigQuery structure

- **Dataset** - contains tables/views
- **Table** = collection of columns
- **Job** = long running action/query



## Identity and Access Management (IAM)

- Control by project, dataset, view
- Cannot control at table level
  - But can control by views via datasets as alternative (virtual table defined by SQL query)
- Predefined roles - BigQuery...
  - 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)
- [Roles comparison matrix](#)
- Sharing datasets
  - Make public with All Authenticated Users

[Return to Table of Contents](#)

### Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *BigQuery Overview*

[Previous](#)

### Pricing

- Storage, Queries, Streaming insert
- Storage = \$0.02/GB/mo (first 10GB/mo free)
  - Long term storage (not edited for 90 days) = \$0.01/GB/mo
- Queries = \$5/TB (first TB/mo free)
- Streaming = \$0.01/200 MB
- Pay as you go, with high end flat-rate query pricing
- Flat rate - starts at \$40K per month with 2000 slots

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *Interacting with BigQuery*

### Interaction methods

[Next](#)

- Web UI
- Command line (bq commands)
  - bq query --arguments 'QUERY'
- Programmatic (REST API, client libraries)
- Interact via queries

### Querying tables

- FROM `project.dataset.table` (Standard SQL)
- FROM [project:dataset.table] (Legacy SQL)

### Searching multiple tables with wildcards

Query across multiple, similarly named tables

- FROM `project.dataset.table\_prefix\*`

Filter further in WHERE clause

- AND \_TABLE\_SUFFIX BETWEEN 'table003' and 'table050'

### Advanced SQL queries are allowed

- JOINS, sub queries, CONCAT

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *Interacting with BigQuery*

[Previous](#)

### Views

- Virtual table defined by query
- 'Querying a query'
- Contains data only from query that contains view
- Useful for limiting table data to others

### Cached queries

- Queries cost money
- Previous queries are cached to avoid charges if ran again
- command line to disable cached results
  - `bq query --no_use_cache '(QUERY)'`
- Caching is per user only

### User Defined Functions (UDF)

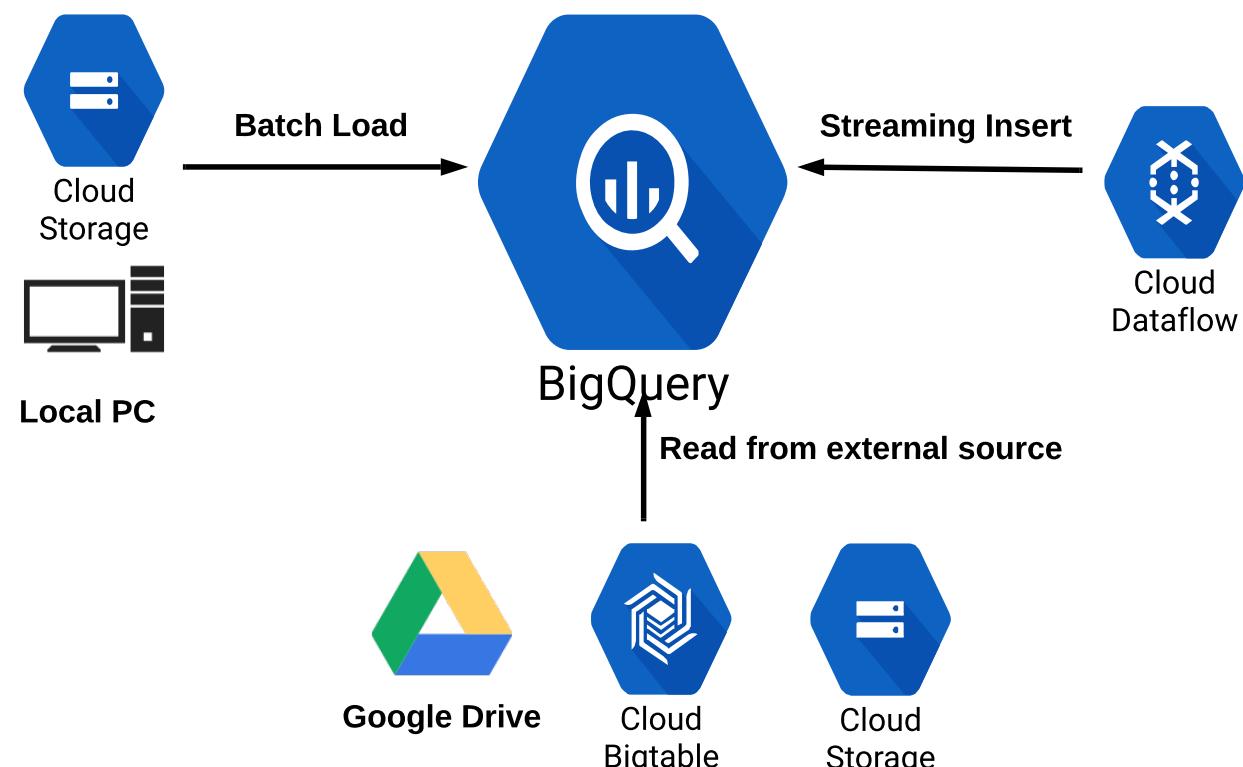
- Combine SQL code with JavaScript/SQL functions
- Combine SQL queries with programming logic
- Allow much more complex operations (loops, complex conditionals)
- WebUI only usable with Legacy SQL

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)[Next](#)

## Loading and reading sources



### Data formats:

#### Load

- CSV
- JSON (Newline delimited)
- Avro - best for compressed files
- Parquet
- Datastore backups

#### Read

- CSV
- JSON (Newline delimited)
- Avro
- Parquet

### Why use external sources?

- Load and clean data in one pass from external, then write to BigQuery
- Small amount of frequently changing data to join to other tables

### Loading data with command line

- `bq load --source_format=[format] [dataset].[table] [source_path] [schema]`
- Can load multiple files with command line (not WebUI)

[Return to Table of Contents](#)

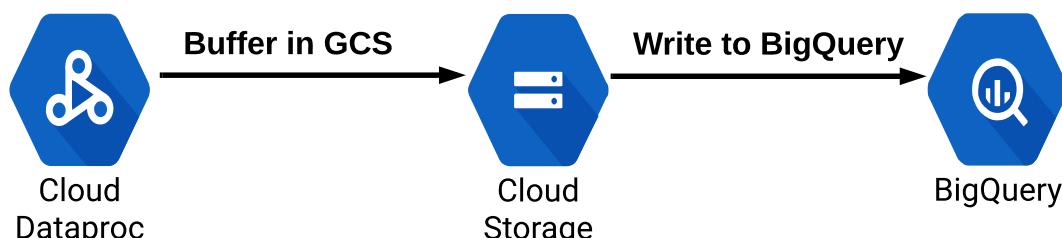
## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)[Previous](#)

## Load and Export Data

### Connecting to/from other Google Cloud services

- Dataproc - Use BigQuery connector (installed by default), job uses Cloud Storage for staging



### Exporting tables

- Can only export to Cloud Storage
- Can copy table to another BigQuery dataset
- Export formats: CSV, JSON, Avro
- Can export multiple tables with command line
- Can only export up to 1GB per file, but can split into multiple files with wildcards
- Command line
  - `bq extract 'projectid:dataset.table' gs://bucket_name/folder/object_name`
  - Can drop 'project' if exporting from same project
  - Default is CSV, specify other format with `--destination_format`
  - `--destination_format=NEWLINE_DELIMITED_JSON`

### BigQuery Transfer Service

- Import data to BigQuery from other Google advertising SaaS applications
- Google AdWords
- DoubleClick
- YouTube reports

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## ***Optimize for Performance and Costs***

### Performance and costs are complementary

[Next](#)

- Less work = faster query = less costs
- What is 'work'?
  - I/O - how many bytes read?
  - Shuffle - how much passed to next stage
  - How many bytes written?
  - CPU work in functions

### General best practices

- Avoid using **SELECT \***
- Denormalize data when possible
  - Grouping data into single table
  - Often with nested/repeated data
  - Good for read performance, not for write (transactional) performance
- Filter early and big with **WHERE** clause
- Do biggest joins first, and filter pre-JOIN
- **LIMIT** does not affect cost
- Partition data by date
  - Partition by ingest time
  - Parition by specified data columns

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

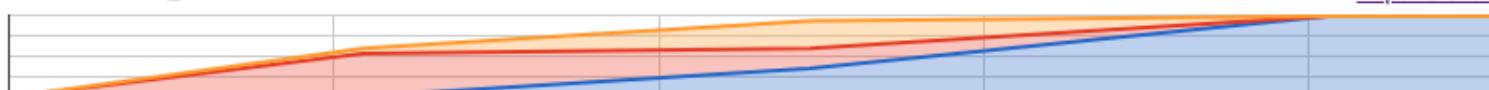
## Optimize for Performance and Costs

[Previous](#)

### Monitoring query performance

- Understand color codes
- Understand 'skew' in difference between average and max time

#### Timeline

[Expand chart](#)

#### Execution Plan

	S00: Input	Stage timing				Parallel Inputs	Rows	
		Wait	Read	Compute	Write		Input	Output
▶	S00: Input	✓	█	███████	█	105	122 M	5.34 K (78.2 KB)
▶	S02: Coalesce	✓	██	██	███	100	5.34 K	5.34 K (78.2 KB)
▶	S03: Join+	✓	██	██	███	74	22.1 M	37 (925 B)
▶	S04: Aggregate+	✓	██	██	███	17	37	19 (646 B)
▶	S05: Output	✓	█	██	███	1	19	19 (532 B)

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

# The Data Dossier

## *Streaming Insert Example*

### Quick setup

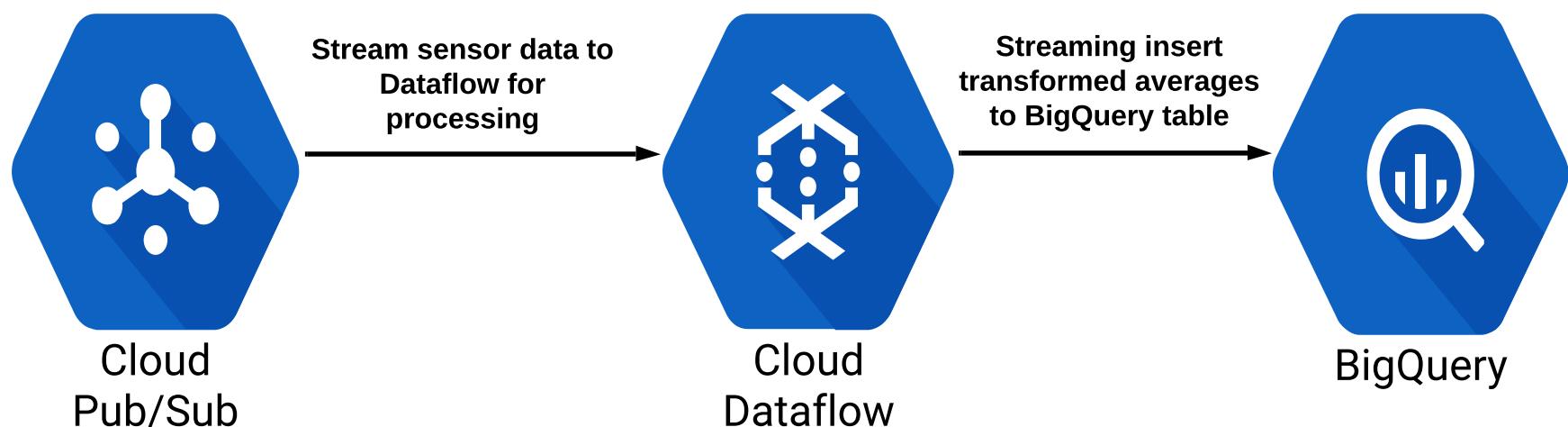
```
cd
```

```
gsutil cp -r gs://gcp-course-exercise-scripts/data-engineer/* .  
bash streaming-insert.sh
```

### Clean up

```
bash streaming-cleanup.sh
```

Manually stop Dataflow job



[Return to Table of Contents](#)

### Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

# The Data Dossier

## *BigQuery Logging and Monitoring*

### Stackdriver Monitoring and Logging Differences

- Monitoring = performance/resources
- Logging = who is doing what
  - History of actions

### Monitoring BigQuery Performance/Resources

- Monitoring = metrics, performance, resource capacity/usage (slots)
  - Query count, query times, slot utilization
  - Number of tables, stored and uploaded bytes over time
  - Alerts on metrics e.g., long query times
    - Example: Alerts when queries take more than one minute
- No data on who is doing what, or query details

### Stackdriver Logging: "A Paper Trail"

- Logging = who is doing what
- Record of jobs and queries associated with accounts

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *BigQuery Best Practices*

### Data Format for Import

[Next](#)

- Best performance = Avro format
- Scenario: Import multi-TB databases with millions of rows

#### Faster

Avro - Compressed

Avro - Uncompressed

Parquet

CSV

JSON

CSV - Compressed

JSON - Compressed

#### Slower

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## ***BigQuery Best Practices***

[Previous](#)[Next](#)

### **Partitioned Tables**

#### **What is a partitioned table?**

- Special single table
  - Divided into segments known as “partitions”

#### **Why is this important?**

- Query only certain rows (partitions) instead of entire table
  - Limits amount of read data
  - Improves performance
  - Reduces costs
- Partition types
  - Ingests time — when the data/row is created
  - Includes **TIMESTAMP** or **DATE** column
- **Scenario:** A large amount of data gets generated every day, and we need to query for only certain time periods within the same table.

#### **Why not use multiple tables (one for each day) plus wildcards?**

- Limited to 1000 tables per dataset
- Substantial performance drop vs. a single table

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *BigQuery Best Practices*

[Previous](#)[Next](#)

### Clustered Tables

- Taking partitioned tables “to the next level”
- Similar to partitioning, divides table reads by a specified column field
  - Instead of dividing by date/time, divides by field
- **Scenario:** Logistics company needs to query by tracking ID
  - Cluster by tracking ID column = only reading table rows with specified tracking ID's
- Restriction: only (currently) available for partitioned tables

### Slots

- Computational capacity required to run a SQL query
  - Bigger/more complex queries need more slots
- Default, on-demand pricing allocates 2000 slots
  - Only an issue for extremely complex queries, or high number of simultaneous users
  - If more than 2000 slots required, switch to **flat-rate pricing**

[Return to Table of Contents](#)

## Choose a Lesson

[BigQuery Overview](#)[Interacting with BigQuery](#)[Load and Export Data](#)[Optimize for Performance and Costs](#)[Streaming Insert Example](#)[BigQuery Logging and Monitoring](#)[BigQuery Best Practices](#)

## *BigQuery Best Practices*

[Previous](#)

### Backup and Recovery

- Highly available = multi-regional dataset vs. regional
- Backup/recovery = BigQuery automatically takes continuous snapshots of tables
  - 7 day history, but 2 days if purposely deleted
- Restore to previous point in time using `@(time)`, in milliseconds
- Example: Get snapshot from one hour ago

#legacySQL

```
SELECT * FROM [PROJECT_ID:DATASET.TABLE@-3600000]
```

- Alternatively, export table data to GCS, though not as cost effective

[Return to Table of Contents](#)

# The Data Dossier

## Choose a Lesson

[What is Machine Learning?](#)
[Working with Neural Networks](#)
[Preventing Overfitted Training Models](#)

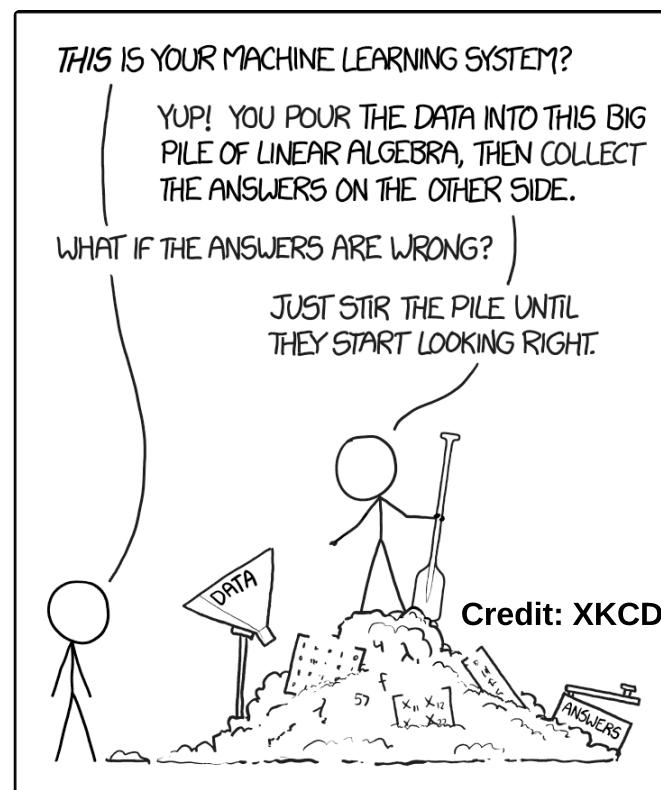
**For Data Engineer:**  
Know the training and inference stages of ML

## *What is Machine Learning?*

Popular view of machine learning...

[Next](#)

DATA →



→ MAGIC!

## So what is machine learning?

Process of combining inputs to produce useful predictions on never-before-seen data

Makes a machine learn from data to make predictions on future data, instead of programming every scenario



New, unlabeled  
image



"I have never seen  
this image before,  
but I'm pretty sure  
that this is a cat!"

[Return to Table of Contents](#)

# The Data Dossier

## Choose a Lesson

[What is Machine Learning?](#)
[Working with Neural Networks](#)
[Preventing Overfitted Training Models](#)
[Previous](#)
[Next](#)

## *What is Machine Learning?*

### How it works

- Train a model with examples
- Example = input + label
- Training = adjust model to learn relationship between features and label - minimize error:
  - Optimize weights and biases (parameters) to different input features
- Feature = input variable(s)
- Inference = apply trained model to unlabeled examples
- Separate test and training data ensures model is generalized for additional data:
  - Otherwise, leads to overfitting (only models to training data, not new data)

Input + Label



"Cat"

Train on many examples  
Training dataset

Everything is numbers!



Train on ML model      "I think this is a cat"      Predict with trained model



Match labels by adjusting weights to input features

Array RGB									
Page 3 - blue intensity values									
0.342	0.647	0.515	0.816	...	0.689	0.703	0.118	0.884	...
0.111	0.300	0.205	0.526	...	0.538	0.538	0.653	0.925	...
0.523	0.428	0.712	0.929	...	0.314	0.268	0.159	0.101	...
0.214	0.604	0.918	0.344	...	0.553	0.633	0.528	0.493	...
0.100	0.121	0.173	0.126	...	0.441	0.465	0.512	0.512	...
0.204	0.175	0.204	0.175	...	0.298	0.401	0.421	0.398	...
0.760	0.531	0.912	0.713	...	0.912	0.713	0.219	0.328	...
0.997	0.910	0.219	0.328	...	0.995	0.726	0.128	0.133	...
0.455	0.783	0.224	0.395	...	...	...	...	...	...
0.021	0.500	0.311	0.123	...	...	...	...	...	...
1.000	1.000	0.867	0.051	...	...	...	...	...	...
1.000	0.945	0.998	0.893	...	...	...	...	...	...
0.990	0.941	1.000	0.876	...	...	...	...	...	...
0.902	0.867	0.834	0.798	...	...	...	...	...	...

n-dimensional arrays called 'tensor', hence TensorFlow

[Return to Table of Contents](#)

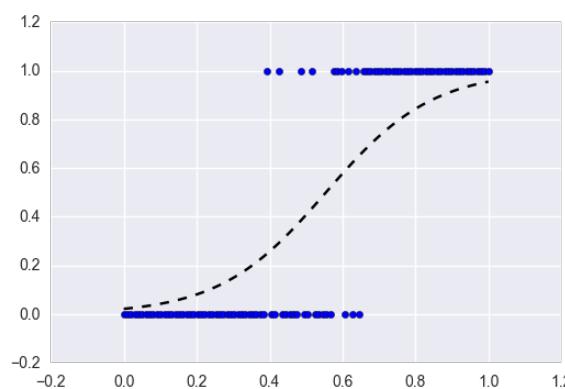
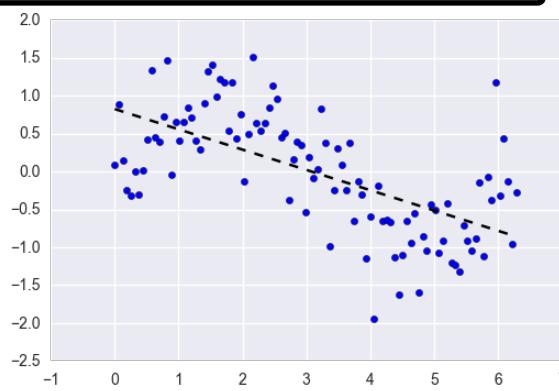
## Choose a Lesson

[What is Machine Learning?](#)[Previous](#)

## Learning types

- **Supervised learning**
  - Apply labels to data ("cat", "spam")
  - Regression - Continuous, numeric variables:
    - Predict stock price, student test scores
  - Classification - categorical variables:
    - yes/no, decision tree
    - "is this email spam?" "is this picture a cat?"
  - Same types for dataset columns:
    - continuous (regression) and categorical (classification)
    - income, birth year = continuous
    - gender, country = categorical
- **Unsupervised learning**
  - Clustering - finding patterns
  - Not labeled or categorized
  - "Given the location of a purchase, what is the likely amount purchased?"
  - Heavily tied to statistics
- **Reinforcement Learning**
  - Use positive/negative reinforcement to complete a task
    - Complete a maze, learn chess

## Classification



[Return to Table of Contents](#)

## Choose a Lesson

[What is Machine Learning?](#)[Working with Neural Networks](#)[Preventing Overfitted Training Models](#)

## **Working with Neural Networks**

Hands on learning tool

[playground.tensorflow.org](http://playground.tensorflow.org)

[Next](#)

### Key terminology

- Neural network - model composed of layers, consisting of connected units (neurons):
  - Learns from training datasets
- Neuron - node, combines input values and creates one output value
- Input - what you feed into a neuron (e.g. cat pic)
- Feature - input variable used to make predictions
  - Detecting email spam (subject, key words, sender address)
  - Identify animals (ears, eyes, colors, shapes)
- Hidden layer - set of neurons operating from same input set
- Feature engineering - deciding which features to use in a model
- Epoch - single pass through training dataset
  - Speed up training by training on a subset of data vs. all data

### Making Adjustments with Parameters

- Weights - multiplication of input values
- Bias - value of output given a weight of 0
- ML adjusts these parameters automatically
- Parameters = variables adjusted by training with data

$$w_1x_1 + w_2x_2 > b$$

[Return to Table of Contents](#)

## Choose a Lesson

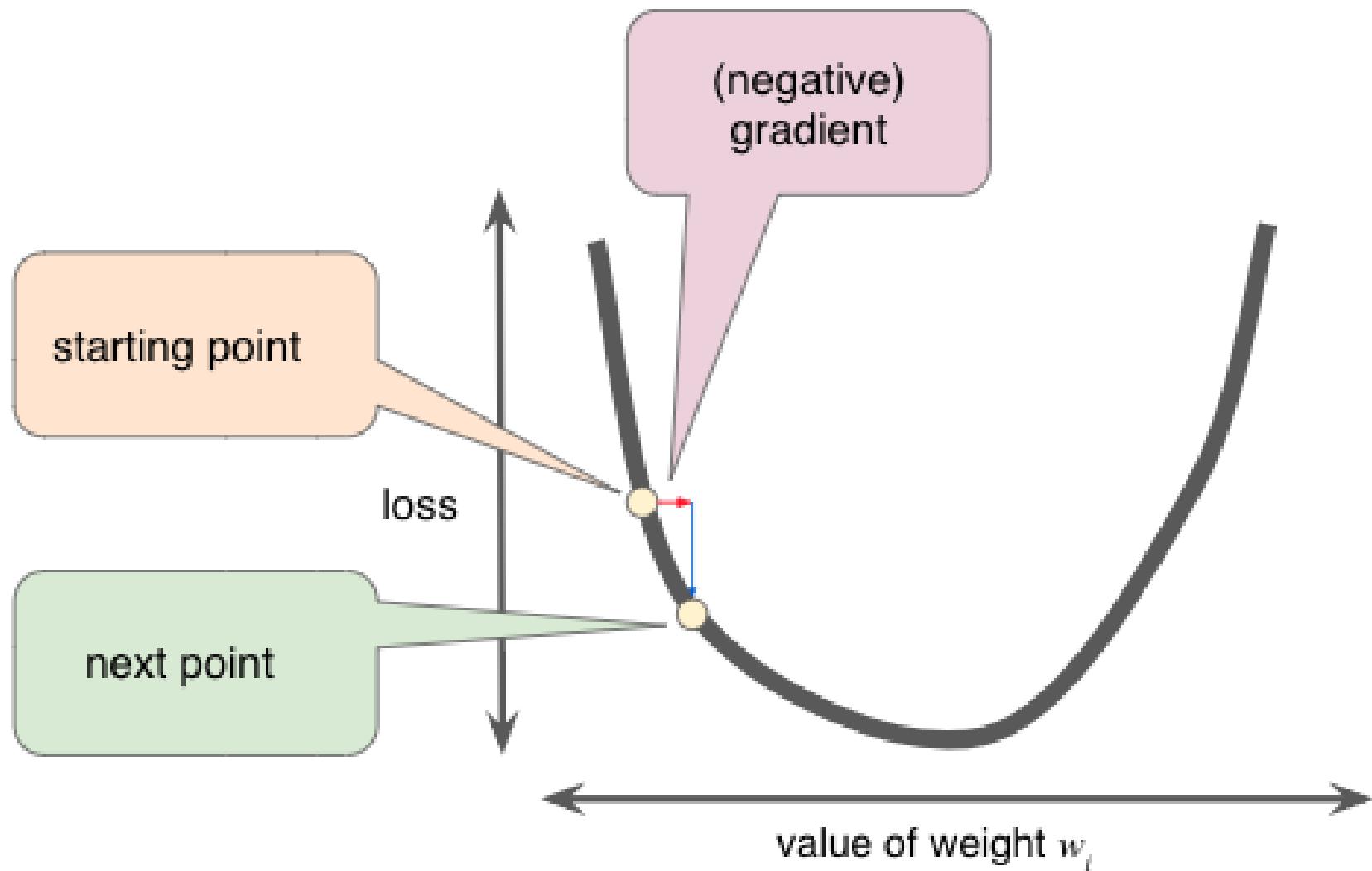
[What is Machine Learning?](#)[Working with Neural Networks](#)[Preventing Overfitted Training Models](#)

## Working with Neural Networks

[Previous](#)[Next](#)

### Rate of adjustments with Learning Rate

- Magnitude of adjustments of weights and biases
- Hyperparameter = variables about the training process itself:
  - Also includes hidden layers
  - Not related to training data
- Gradient descent - technique to minimize loss (error rate)
- Challenge is to find the correct learning rate:
  - Too small - takes forever
  - Too large - overshoots



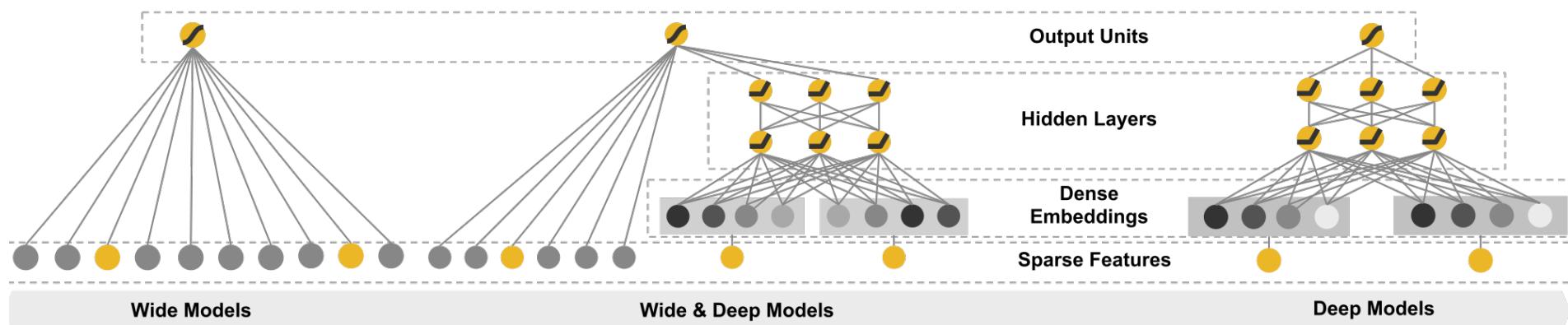
[Return to Table of Contents](#)

## Choose a Lesson

[What is Machine Learning?](#)[Working with Neural Networks](#)[Preventing Overfitted Training Models](#)[Previous](#)

## Deep and wide neural networks

- **Wide - memorization:**
  - Many features
- **Deep - generalization:**
  - Many hidden layers
- **Deep and wide = both:**
  - Good for recommendation engines



[Return to Table of Contents](#)

## Choose a Lesson

[What is Machine Learning?](#)[Working with Neural Networks](#)[Preventing Overfitted Training Models](#)[Next](#)

## *Preventing Overfitted Training Models*

### What is Overfitting?

- Training model *overfitted* to training data: Unable to generalize with new data
- Training model *fails to generalize*: Accounting for slightly different but close enough data

### Causes of Overfitting:

- Not enough training data
  - Need more variety of samples
- Too many features
  - Too complex
- Model fitted to unnecessary features unique to training data, a.k.a. "Noise"

### Solving for Overfitted Model:

- Use more data:
  - Add more training data.
  - More varied data allows for better generalization.
- Make the model less complex:
  - Use less (but more relevant) features.
  - Combine multiple co-dependant/redundant features into a single representative feature:
    - This also helps reduce model training time.
- Remove noise:
  - Increase regularization parameters

[Return to Table of Contents](#)

## Choose a Lesson

[What is Machine Learning?](#)[Working with Neural Networks](#)[Preventing Overfitted Training Models](#)

## Preventing Overfitted Training Models

[Previous](#)

### Regularization?

- Adds a penalty to a model as it becomes more complex
- Penalizing parameters = better generalization
- Cuts out *noise* and unimportant data, to avoid overfitting

### Regularization types:

- L1 and L2 regularization - Different approaches to tuning out noise.  
Each has different use case and purpose.
- L1 - Lasso Regression: Assigns greater importance to more influential features
  - Shrinks less important features influence to zero
  - Good for models with many features, some more important than others
  - Example: Choosing features to predict likelihood of home selling:
    - House price more influential feature than carpet color
- L2 - Ridge Regression: Performs better when all the input features influence the output, and with all weights being of roughly equal size

[Return to Table of Contents](#)

## Choose a Lesson

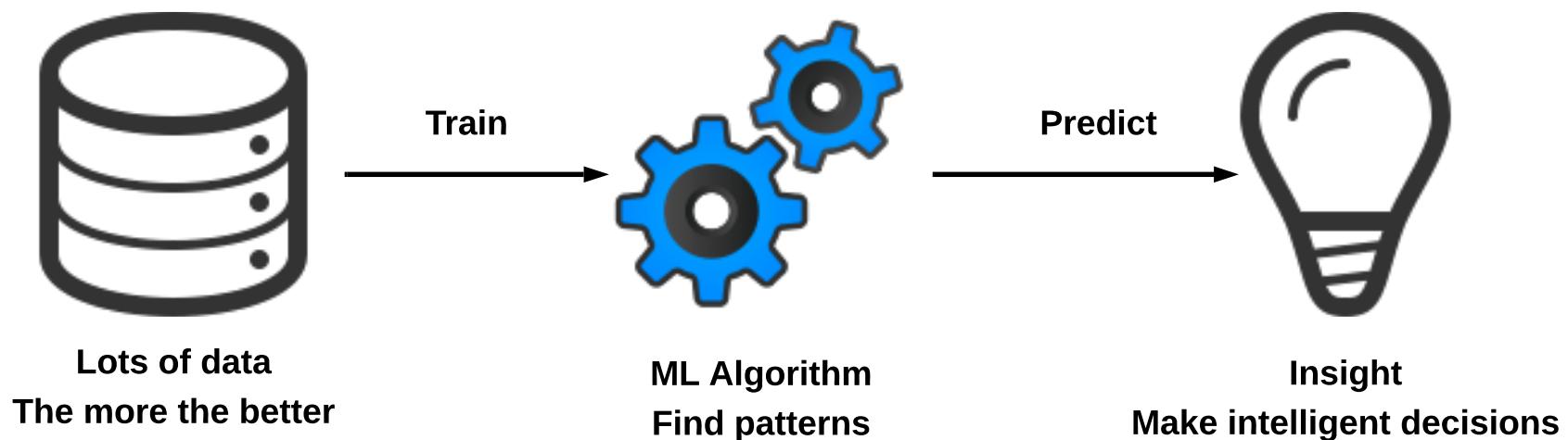
[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## **GCP Machine Learning Services**

### Machine Learning - In a nutshell

[Next](#)

- Algorithm that is able to learn from data



Achieving this requires:  
**Lots of data (and data storage)**  
**Lots of Compute**  
**How can GCP help?**

[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## **GCP Machine Learning Services**

[Previous](#)[Next](#)

### Different Roles = Different Priorities

- **Data Scientist/Machine Learning Engineer**
- **Application Developer**

#### **Data Scientist/ML Engineer**

- Works directly with ML libraries (e.g. Tensorflow)
- Creates, trains, and adjusts ML models
- "Does the math" for ML algorithms
- Gets into the ML details:
  - Parameters, biases, features, etc
- Values customization over simplicity

#### **Application Developer**

- Wants to 'plug in' ML capabilities to their app
- Avoids the mathematical details
- Values 'plug and play' solution

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Pre-trained ML API's*

[Next](#)

**AI Platform**  
**(Formerly**  
**Cloud ML**  
**Engine)**

- Train, deploy, and manage custom ML models on managed infrastructure resources.
- You create the model, then Google provides managed infrastructure for testing it.



**Pre-trained  
ML models**

- Pre-trained models
- Common use cases (not customizable)
- Simply 'plug' into your application
- "Make Google do it"

[Return to Table of Contents](#)

## AI Platform Overview

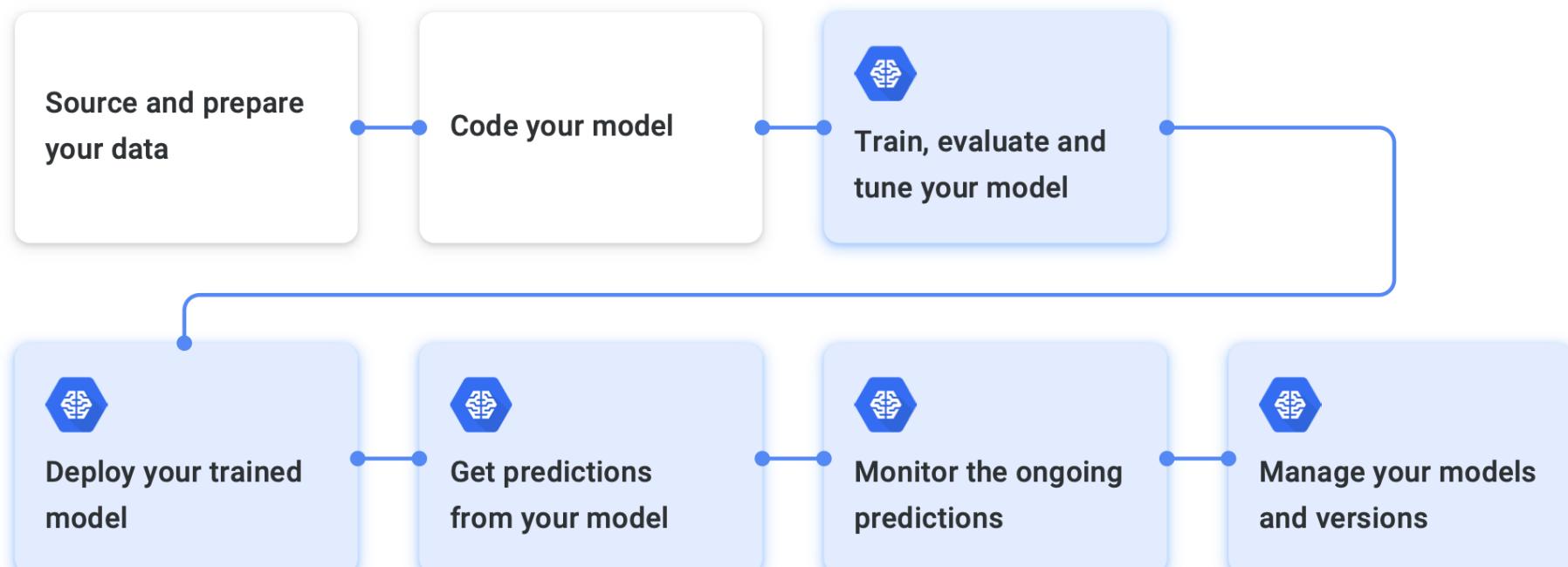
### Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)[Next](#)

### AI Platform (formerly Cloud ML Engine)

- Fully managed Tensorflow (and other ML libraries) platform
- Distributed training and prediction:
  - Breaks jobs down into pieces, distributes to multiple workers
- Scales to tens of CPUs/GPUs/TPUs
- Hyperparameter tuning with Hypertune
- Automate the "annoying bits" of machine learning
- "I want to train my own model, but automate it."

### High Level Overview



[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## *AI Platform Overview*

[Previous](#)[Next](#)

### Tensorflow? ML Libraries?



### TensorFlow

- Software library for high performance numerical computation
- Released as open source by Google in 2015
- Often the default ML library of choice
- Pre-processing, feature creation, model training
- "I want to work with all of the detailed pieces."

[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## AI Platform Overview

[Previous](#)[Next](#)

### How AI Platform Works

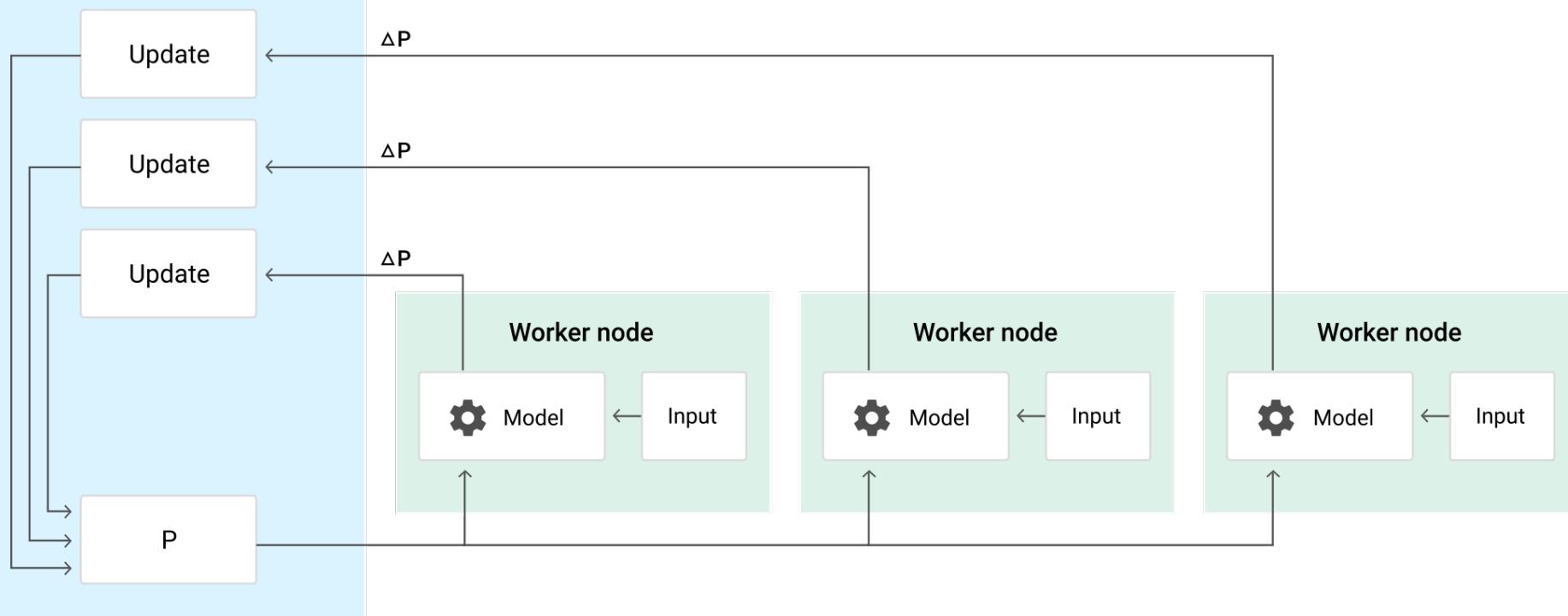
#### Prepare trainer and data for the cloud:

- Write training application in Tensorflow (or other ML library)
- Python is language of choice

#### Train your model with AI Platform:

- **Master** - Manages other nodes
- **Workers** - Works on portion of training job
- **Parameter servers** - Coordinates shared model states between workers

Parameter node



[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## *AI Platform Overview*

[Previous](#)[Next](#)

### Get Predictions - two types:

- **Online:**
  - High rate of requests with minimal latency
  - Give job data in JSON request string, predictions returned in its response message
- **Batch:**
  - Get inference (predictions) on large collections of data with minimal job duration
  - Input and output in Cloud Storage

### Key Terminology

- **Model** - Logical container of individual solutions to a problem:
  - Can deploy multiple versions
  - e.g. Sale price of houses given data on previous sales
- **Version** - Instance of model:
  - e.g. version 1/2/3 of how to predict above sale prices
- **Job** - interactions with AI Platform:
  - Train models:
    - Command = 'submit job train model' on AI Platform
  - Deploy trained models:
    - Command = 'submit job deploy trained model' on AI Platform
  - 'Failed' jobs can be monitored for troubleshooting

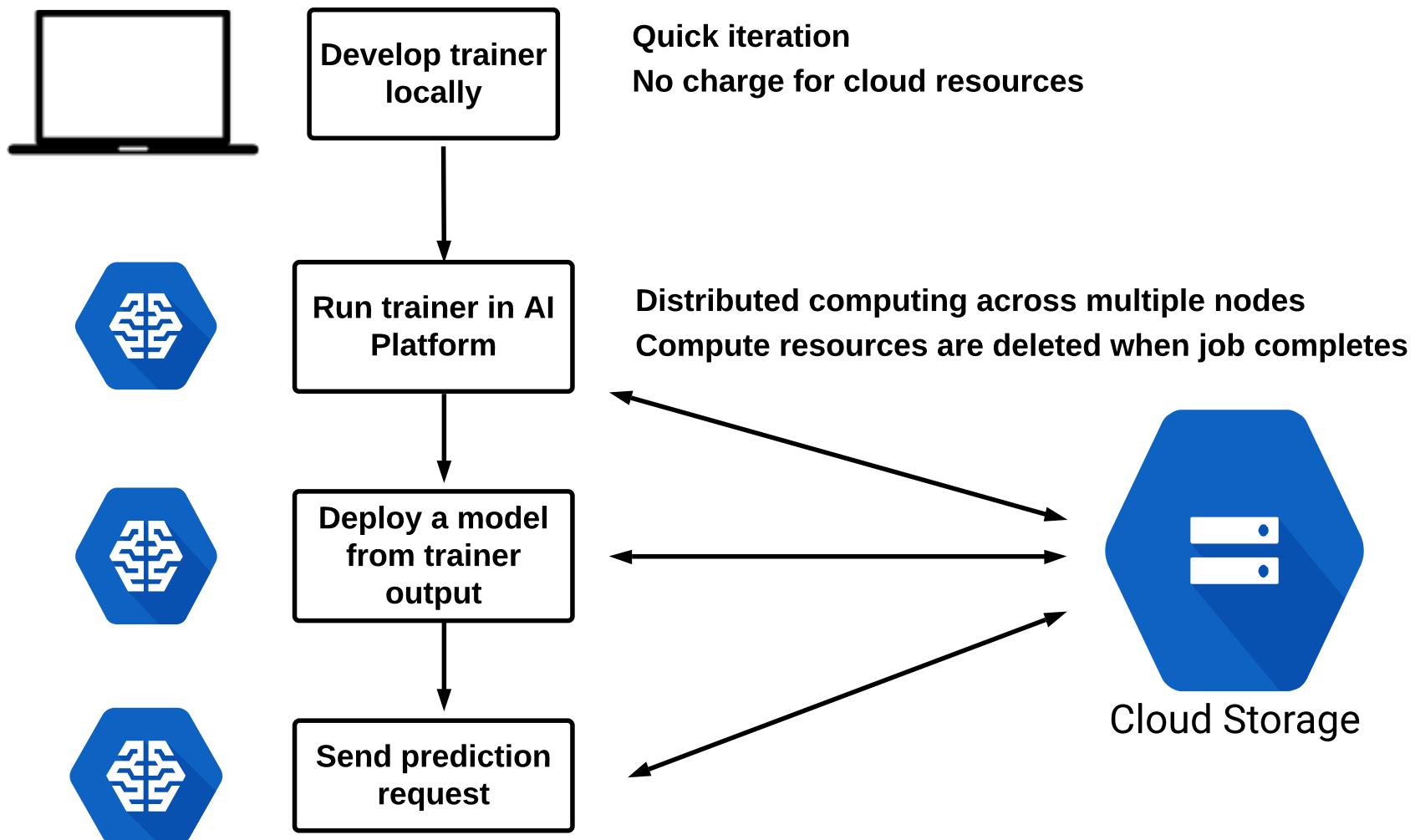
[Return to Table of Contents](#)

## AI Platform Overview

### Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)[Previous](#)[Next](#)

### Typical process



[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## AI Platform Overview

[Previous](#)[Next](#)

### Must-Know Info

- Currently supports Tensorflow, scikit-learn, and XGBoost frameworks

*Note: This list is subject to change over time*

- 

### IAM roles:

- **Project and Models:**
  - **Admin** - Full control
  - **Developer** - Create training/prediction jobs, models/versions, and send prediction requests
  - **Viewer** - Read-only access to above
- **Models only:**
  - **Model Owner:**
    - Full access to model and versions
  - **Model User:**
    - Read models and use for prediction
    - Easy to share specific models

### Using BigQuery for data source:

- Can read directly from BigQuery via training application
- Recommended to pre-process into Cloud Storage
- Using gcloud commands, only works with Cloud Storage



BigQuery



Cloud Storage



AI Platform

[Return to Table of Contents](#)

## AI Platform Overview

### Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)[Previous](#)[Next](#)

### Machine Scale Tiers and Pricing

- BASIC: single worker instance
- STANDARD\_1: 1 master, 4 workers, 3 parameter servers
- PREMIUM\_1: 1 master, 19 workers, 11 parameter servers
- BASIC\_GPU: 1 worker with GPU
- CUSTOM

#### GPU/TPU:

- Much faster processing performance

#### Pricing:

- Priced per hour
- Higher cost for TPU/GPU's

Training - Predefined scale tiers - price per hour		Training - AI Platform machine types - price per hour		Training - Compute Engine machine types - price per hour		Training - Accelerators - price per hour	
BASIC	\$0.1900	standard	\$0.1900	n1-standard-4	\$0.1900	NVIDIA_TESLA_K80	\$0.4500
STANDARD_1	\$1.9880	large_model	\$0.4736	n1-standard-8	\$0.3800	NVIDIA_TESLA_P4 (Beta)	\$0.6000
PREMIUM_1	\$16.5536	complex_model_s	\$0.2836	n1-standard-16	\$0.7600	NVIDIA_TESLA_P100	\$1.4600
BASIC_GPU	\$0.8300	complex_model_m	\$0.5672	n1-standard-32	\$1.5200	NVIDIA_TESLA_T4 (Beta)	\$0.9500
BASIC_TPU	\$4.6900	complex_model_l	\$1.1344	n1-standard-64	\$3.0400	NVIDIA_TESLA_V100	\$2.4800
CUSTOM	See the tables of machine types.	standard_gpu	\$0.8300	n1-standard-96	\$4.5600	Eight TPU_V2 cores*	\$4.5000
		complex_model_m_gpu	\$2.5600	n1-highmem-2	\$0.1184	Batch prediction - price per node hour	
		complex_model_l_gpu	\$3.3200	n1-highmem-4	\$0.2368	\$0.0791	
		standard_p100	\$1.8400	n1-highmem-8	\$0.4736	Online prediction - Machine types - price per node hour.	
		complex_model_m_p100	\$6.6000	n1-highmem-16	\$0.9472	mls1-c1-m2 (default)	\$0.0401
		standard_v100	\$2.8600	n1-highmem-32	\$1.8944	mls1-c4-m2 (Beta)	\$0.1349
		large_model_v100	\$2.9536	n1-highmem-64	\$3.7888		
		complex_model_m_v100	\$10.6800	n1-highmem-96	\$5.6832		
		complex_model_l_v100	\$21.3600	n1-highcpu-16	\$0.5672		
		cloud_tpu*	\$4.5000	n1-highcpu-32	\$1.1344		
				n1-highcpu-64	\$2.2688		
				n1-highcpu-96	\$3.4020		

[Return to Table of Contents](#)

## Choose a Lesson

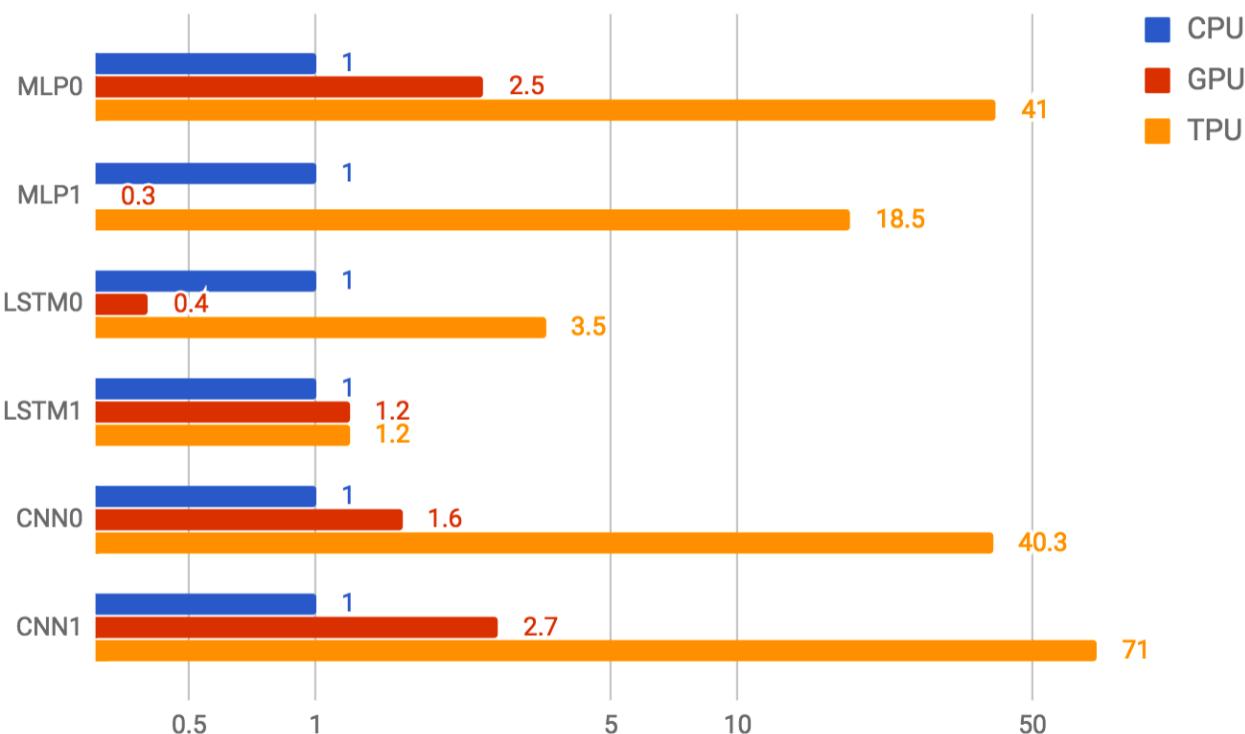
[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## AI Platform Overview

[Previous](#)[Next](#)

### Tensor Processing Unit (TPU)

- Hardware processing specifically designed for machine learning
  - Like a GPU, but even more optimized for ML
  - Faster and more efficient



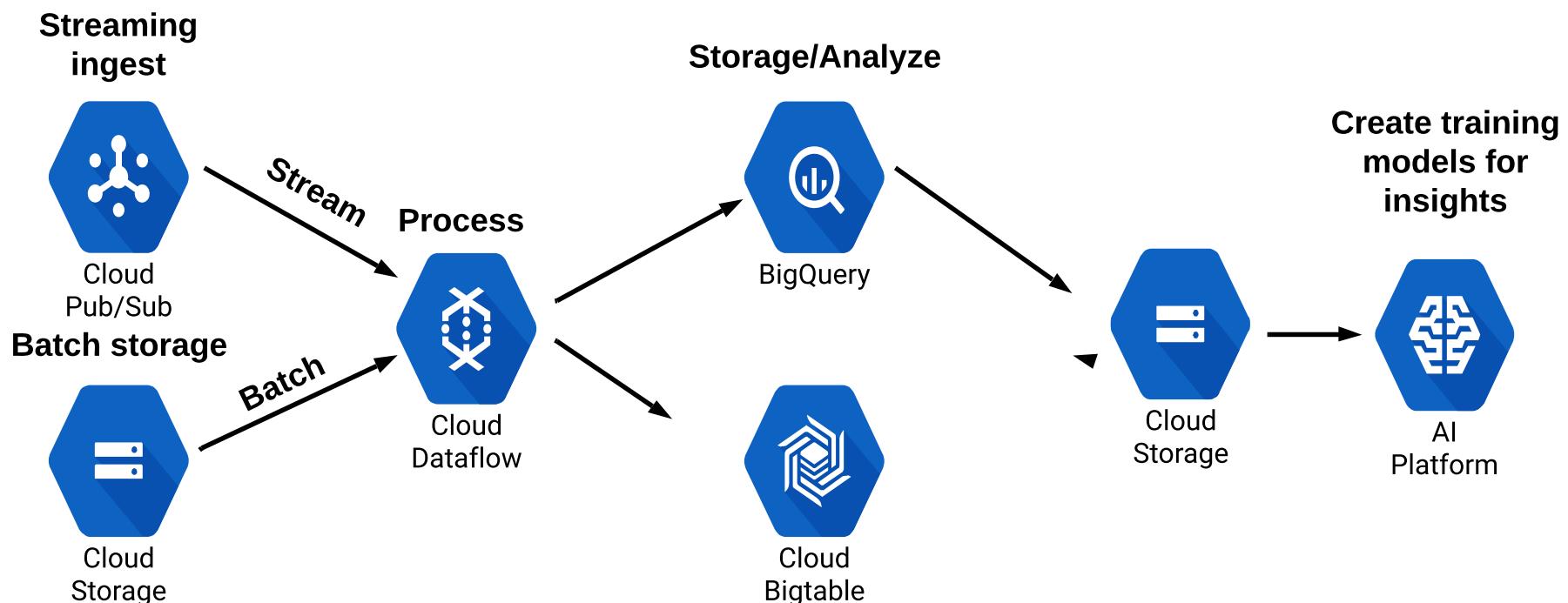
[Return to Table of Contents](#)

## AI Platform Overview

### Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)[Previous](#)

### Big Picture



[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## *AI Platform Hands On*

### What We Are Doing:

- Working with pre-packaged training model:
  - Focusing on the AI Platform aspect, not TensorFlow
- Heavy command line/gcloud focus, using Cloud Shell

### Main steps: The big picture

- Submit training job locally using ai-platform commands
- Submit training job on AI Platform, both single and distributed
- Deploy trained model, and submit predictions

## Instructions for Hands On

Download scripts to Cloud Shell to follow along:

```
gsutil -m cp gs://gcp-course-exercise-scripts/data-engineer/ai-platform/* .
```

[Return to Table of Contents](#)

## ML Engine Hands On

### Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)[Previous](#)[Next](#)

### Current ML API's (page 1 of 2)



Image recognition/analysis

Cloud Vision



Text analysis  
Information extraction  
Understanding sentiment

Cloud Natural  
Language



Convert audio to text  
Multi-lingual support  
Understanding sentence structure

Cloud Speech to  
Text



Cloud Translation



Cloud Job  
Discovery



Cloud Text to  
Speech (Beta)

Detect and translate languages

More relevant job searches:  
Power recruitment, job boards

Convert text to audio  
Multiple languages/voices  
Natural sounding synthesis

[Return to Table of Contents](#)

## *ML Engine Hands On*

### Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)[Previous](#)[Next](#)

### Current ML API's (page 2 of 2)



Cloud Video  
Intelligence

Video analysis  
Labels, shot changes, explicit  
content



Dialogflow

Dialogflow for  
Enterprise

Conversational experiences  
Virtual assistants

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Pre-trained ML API's*

[Previous](#)[Next](#)

### Current ML APIs (new ones being added)



Image recognition/analysis

Cloud Vision



Detect and translate languages

Cloud Translation



Text analysis  
Extract information  
Understand sentiment

Cloud Natural Language



More relevant job searches:  
Power recruitment, job boards

Cloud Job Discovery



Convert audio to text  
Multi-lingual support  
Understand sentence structure

Cloud Speech to Text



Convert text to audio  
Multiple languages/voices  
Natural sounding synthesis

Cloud Text to Speech



Video analysis  
Labels, shot changes, explicit content

Cloud Video Intelligence



Dialogflow

Dialogflow for Enterprise

Conversational experiences  
Virtual assistants

[Return to Table of Contents](#)

## Choose a Lesson

[GCP Machine Learning Services](#)[AI Platform Overview](#)[AI Platform Hands On](#)

## **GCP Machine Learning Services**

[Previous](#)

### ML Options on Google Cloud Platform

- Products for ML Engineer to Developer roles, and everything in between
- Rapid expansion of solutions: Two primary ones to focus on for exam



**AI Platform**  
**(Formerly**  
**Cloud ML**  
**Engine)**

- Train, deploy, and manage custom ML models on managed infrastructure resources
- You create the model, Google provides managed infrastructure for testing it



**Pre-trained  
ML models**

- Pre-trained models
- Common use cases (not customizable)
- Simply 'plug' into your application
- "Make Google do it"

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Pre-trained ML API's*

[Previous](#)[Next](#)

### Current ML APIs (new ones being added)



Data Loss  
Prevention  
API

Detect, Manage, and Redact Sensitive data

- Credit card numbers, SSN, birthdates, credentials

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Pre-trained ML API's*

[Previous](#)[Next](#)

### Cloud Vision: A Closer Look



<b>Label Detection</b>	Extract info in image across categories: Plane, sports, cat, night, recreation
<b>Text Detection (OCR)</b>	Detect and extract text from images
<b>Safe Search</b>	Recognize explicit content: Adult, spoof, medical, violent
<b>Landmark Detection</b>	Identify landmarks
<b>Logo Detection</b>	Recognize logos
<b>Image Properties</b>	Dominant colors, pixel count
<b>Crop Hints</b>	Crop coordinates of dominant object/face
<b>Web Detection</b>	Find matching web entries

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Pre-trained ML API's*

[Previous](#)[Next](#)

### Newer ML options

#### Auto ML

- Pre-trained APIs, but for custom models!
  - Example: Identify specific geographical features
- Supply your own data to train on
- Currently available for:
  - Vision
  - Video Intelligence
  - Natural Language
  - Translation
  - Structured Data

#### BigQuery ML

- Create and train ML models inside BigQuery
- Use SQL syntax to create models

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Pre-trained ML API's*

[Previous](#)

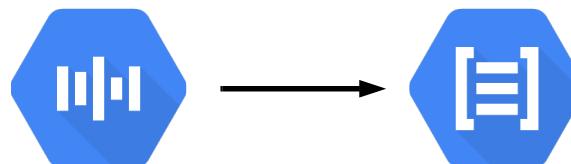
### Exam Perspectives

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

- Can use Cloud Storage URI for GCS stored objects
- Encodes in Base64 format

#### **How to combine API's for scenarios?**

- Search customer service calls and analyze for sentiment



Convert call audio to text  
Make searchable

Analyze text  
for sentiment

[Return to Table of Contents](#)

## Choose a Lesson

[Pre-trained ML API's](#)[Vision API demo](#)

## *Vision API Demo*

### Basic steps for most APIs:

- Enable the API
- Create API key
- Authenticate with API key
- Encode in Base64 (optional)
- Make an API request
- Requests and outputs via JSON

Commands will be in lesson description.

[Return to Table of Contents](#)

## Choose a Lesson

[Datalab Overview](#)

## Datalab Overview

[Next](#)

### What is it?

- Interactive tool for exploring and visualizing data:
  - Notebook format
  - Great for data engineering, machine learning
- Built on Jupyter (formerly iPython):
  - Open source - Jupyter ecosystem
  - Create documents with live code and visualizations
- Visual analysis of data in BigQuery, ML Engine, Compute Engine, Cloud Storage, and Stackdriver
- Supports Python, SQL, and JavaScript
- Runs on GCE instance, dedicated VPC and Cloud Source Repository
- Cost: free - only pay for GCE resources Datalab runs on and other Google Cloud services you interact with



[Return to Table of Contents](#)

## Datalab Overview

### Choose a Lesson

[Datalab Overview](#)[Previous](#)[Next](#)

### How It Works

#### Create and connect to a Datalab instance

`datalab create (instance-name)` →

- Connect via SSH and open web preview
- `datalab connect (instance-name)`
- Open web preview - port 8081



datalab-network



datalab-instance



datalab-notebooks

Source repository

## Working with Datalab

1 Write code in Python

2 Run cell (Shift+Enter)

3 Examine output

4 Write commentary in markdown

5 Share and collaborate



1 j = data[data['dayofweek'] == 7].plot(kind='scatter', x='maxtemp', y='numtrips')

2 Run ▶

3

4 Adding 2014 data

Let's add in 2014 data to the Pandas dataframe. Note how useful it was for us to modularize our queries around the YEAR. Now, the data seem a bit more robust.

5 trips = bq.Query(taxiquery, YEAR=2014).to\_dataframe()

[Return to Table of Contents](#)

## Choose a Lesson

[Datalab Overview](#)

## Datalab Overview

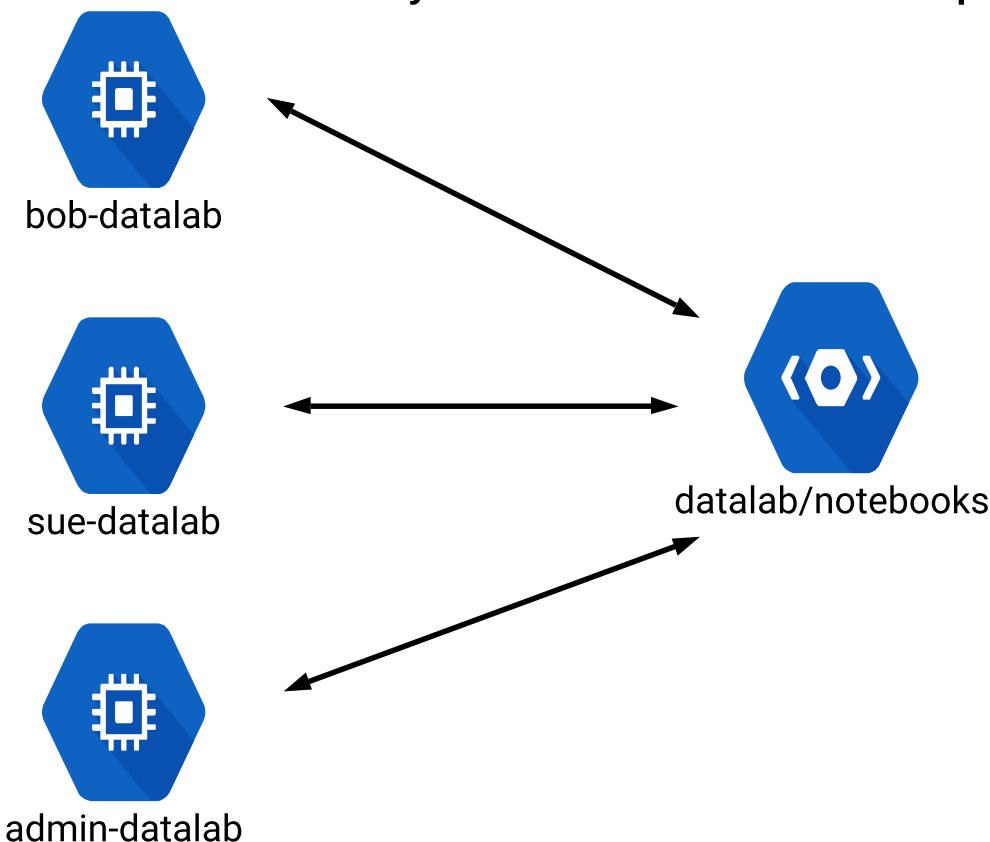
[Previous](#)

### Sharing notebook data:

- GCE access based on GCE IAM roles:
  - Must have Compute Instance Admin and Service Account Actor roles
- Notebook access per user only
- Sharing data performed via shared Cloud Source Repository
- Sharing is at the project level

### Creating team notebooks - two options:

- Team lead creates notebooks for users using --for user option:
  - `datalab create [instance] --for-user bob@professionalwireless.net`
- Each user creates their own datalab instance/notebook
- Everyone accesses same shared repository of datalab/notebooks



[Return to Table of Contents](#)

## Choose a Lesson

[What is Dataprep?](#)

## What is Dataprep?

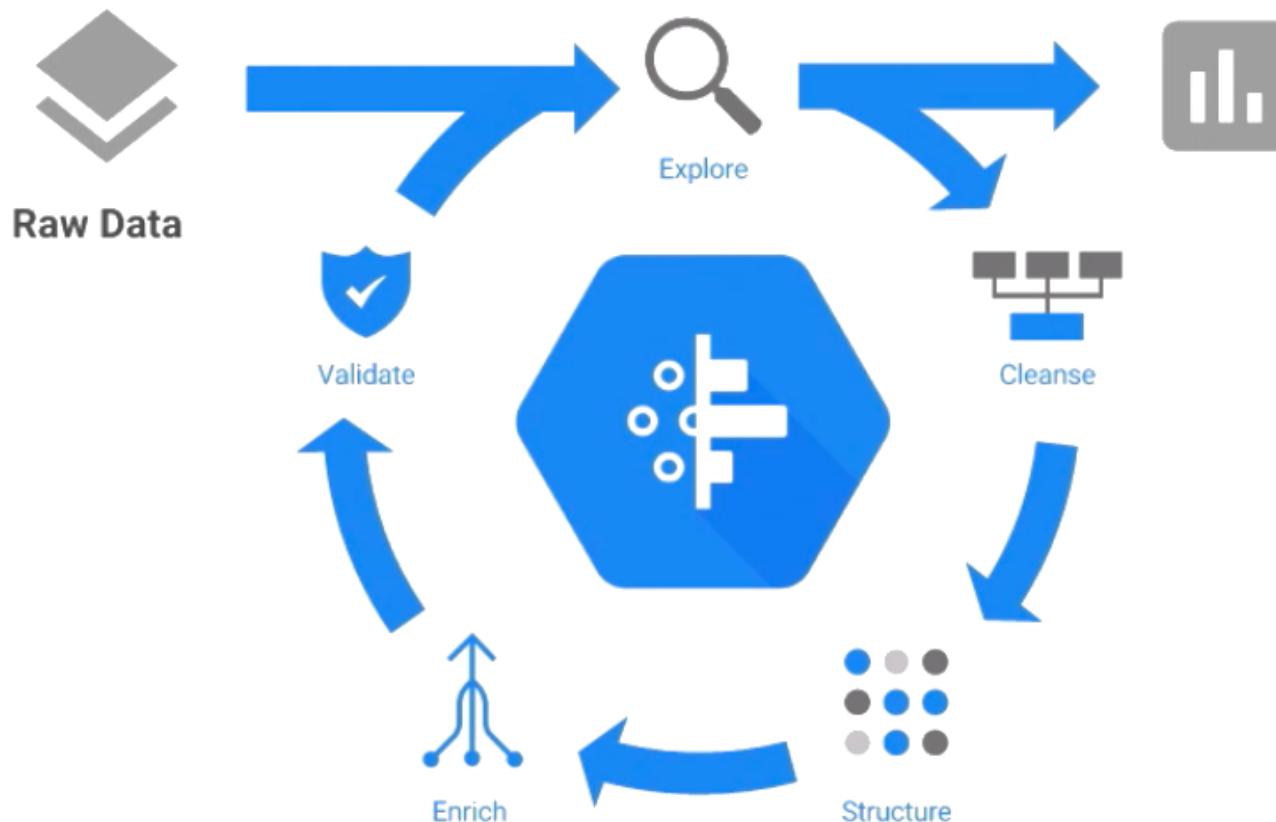
[Next](#)

### What is it?

- Intelligent data preparation
- Partnered with Trifacta for data cleaning/processing service
- Fully managed, serverless, and web-based
- User-friendly interface:
  - Clean data by clicking on it
- Supported file types:
  - Input - CSV, JSON (including nested), Plain text, Excel, LOG, TSV, and Avro
  - Output - CSV, JSON, Avro, BigQuery table:
    - CSV/JSON can be compressed or uncompressed

### Why is this important?

- Data Engineering requires high quality, cleaned, and prepared data
- 80% - time spent in data preparation
- 76% - view data preparation as the least enjoyable part of work
- Dataprep democratizes the data preparation process



[Return to Table of Contents](#)

## Choose a Lesson

[What is Dataprep?](#)[Previous](#)

## *What is Dataprep?*



### 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
- 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 use auto-suggestions

### IAM:

- Dataprep User - Run Dataprep in a project
- Dataprep Service Agent - Gives Trifecta necessary access to project resources:
  - Access GCS buckets, Dataflow Developer, BigQuery user/data editor
  - Necessary for cross-project access + GCE service account

### Pricing:

- **1.16 \* cost of Dataflow job**

[Return to Table of Contents](#)

## Choose a Lesson

[Data Studio Introduction](#)

## Data Studio Introduction

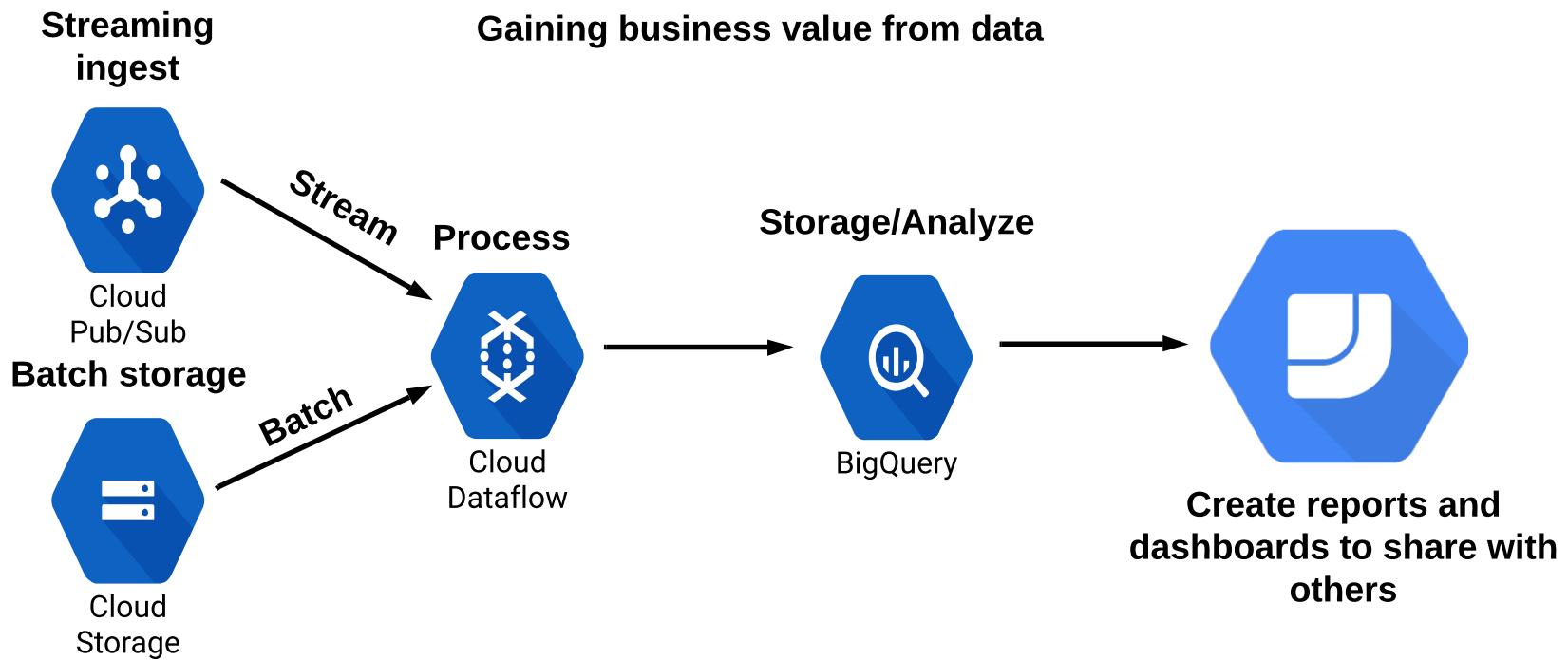
[Next](#)

### What is Data Studio?

- Easy to use data visualization and dashboards:
  - Drag and drop report builder
- Part of G Suite, not Google Cloud:
  - 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:
  - BigQuery, Cloud SQL, GCS, Spanner
  - YouTube Analytics, Sheets, AdWords, local upload
  - Many third party integrations
- Price - Free:
  - BigQuery access run normal query costs

### Data Lifecycle - Visualization

Gaining business value from data



[Return to Table of Contents](#)

## Choose a Lesson

[Data Studio Introduction](#)

## *Data Studio Introduction*

[Previous](#)

### 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 - options for using cached data performance/costs

Two cache types, query cache and prefetch cache

Query cache:

- Remembers queries issued 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:

- '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
- Can be turned off

When to turn caching off:

- Need to view 'fresh data' from rapidly changing data set

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Composer Overview](#)[Hands On - Cloud Composer](#)

## *Cloud Composer Overview*

[Next](#)

### What is Cloud Composer?

- Fully managed Apache Airflow implementation:
  - Infrastructure/OS handled for you

### What is Apache Airflow?

- Programmatically create, schedule, and monitor data workflows

### Why is this important?

- Automation and monitoring
- Big data pipelines are often a multi-step, complex process:
  - Create resources in multiple services
  - Process and move data from one service to another
  - Remove resources when they complete a task
- Collaborate workflow process with other team members

### How Airflow/Composer helps

- Automates the above steps, including scheduling
- Built on open source, using Python as common language
- Easy to work with, and share workflow with others
- Works with non-GCP providers (on-premises, other clouds)



[Return to Table of Contents](#)

## Choose a Lesson

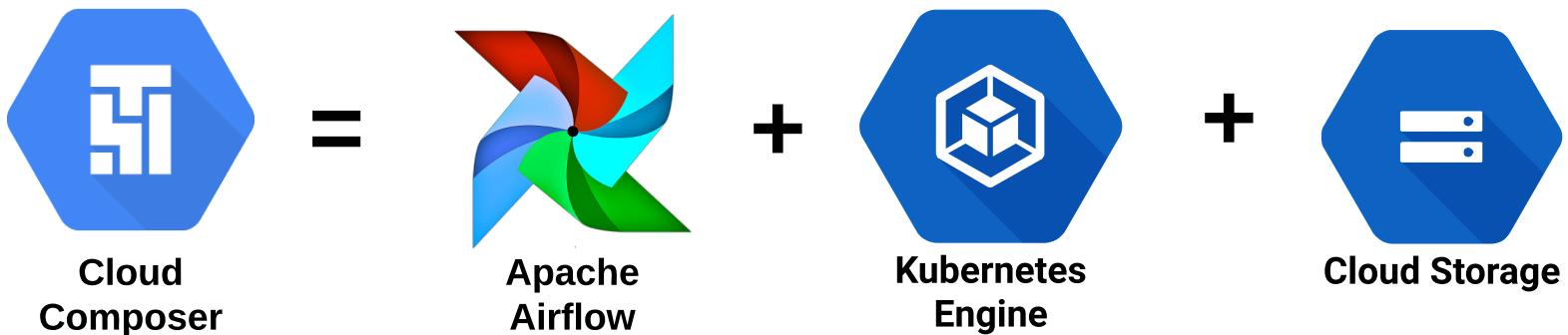
[Cloud Composer Overview](#)[Hands On - Cloud Composer](#)[Previous](#)[Next](#)

## Cloud Composer Overview

### How It Works

Behind the scenes:

- GKE cluster with Airflow implemented
- Cloud Storage bucket for workflow files (and other application files)



### Workflows?

- Orchestrate data pipelines:
  - Like a walkthrough of tasks to run
- Format = Direct Acyclic Graph (DAG):
  - Written in Python
  - Collection of organized tasks that you want to schedule and run
- **Cloud Composer** creates **workflows** using **DAG** files

### The Process

- Create Composer Environment
- Set Composer variables (i.e. project ID, GCS bucket, region)
- Add Workflows (DAG files), which Composer will execute

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Composer Overview](#)[Hands On - Cloud Composer](#)

## *Cloud Composer Overview*

[Previous](#)

### Examples and Exam Perspective

- Create a Dataproc cluster, submit a job, and then delete the cluster.
- Execute a Cloud Dataflow pipeline from data in GCS, and write output to BigQuery.
- Ingest third party data into Cloud Dataflow, process, then upload to GCS.
- **Exam perspective:** Know what DAGs are, and why you'd want to use workflows.

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Composer Overview](#)[Hands On - Cloud Composer](#)

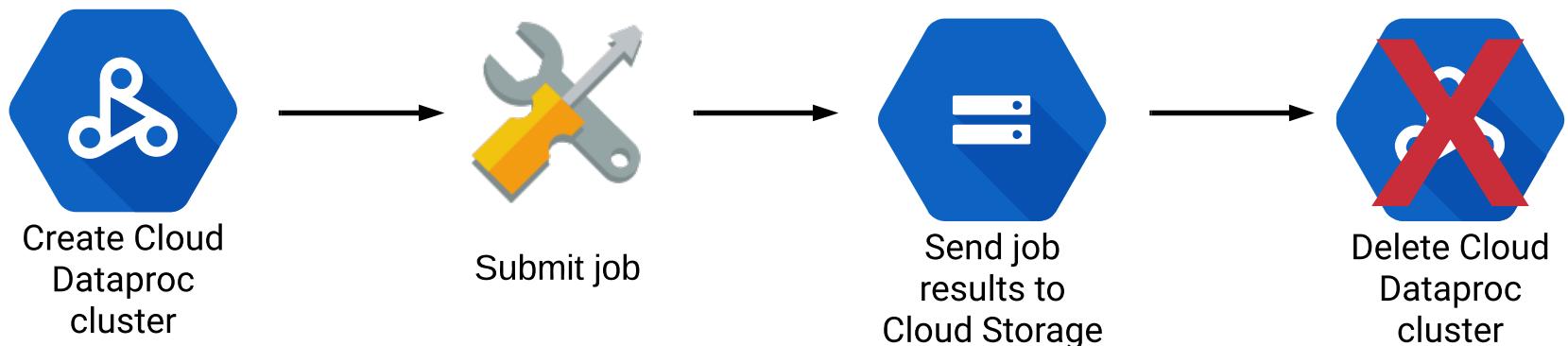
## ***Hands On - Cloud Composer***

[Next](#)

### **The Process:**

- Create the Composer environment.
- Then create the GCS bucket for Dataproc output.
- Assign Cloud Composer variables.
- Upload the workflow file to DAG folder.
- View the results.

### **Automatic processes -- Workflow**



### **Create Composer Environment**

- Enable Composer/Dataproc API
- Create environment in closest region:
  - What's happening?
  - Creating GKE cluster + GCS bucket

### **Create GCS bucket to output Dataproc results**

- `gsutil mb -l us-central1 gs://output-$DEVSHELL_PROJECT_ID`

[Return to Table of Contents](#)

## Choose a Lesson

[Cloud Composer Overview](#)[Hands On - Cloud Composer](#)

## *Hands On - Cloud Composer*

[Previous](#)

### Configure Cloud Composer Variables

- Format
  - `gcloud composer environments run (ENVIRONMENT_NAME) --location (LOCATION) variables -- --set (KEY VALUE)`
- `gcloud composer environments run my-environment --location us-central1 variables -- --set gcp_project (PROJECT-ID)`
- `gcloud composer environments run my-environment --location us-central1 variables -- --set gcs_bucket gs://output-(PROJECT-ID)`
- `gcloud composer environments run my-environment --location us-central1 variables -- --set gce_zone us-central1-c`

### Add workflow file (Python) to Composer DAG folder:

- [github link](#)

Next step? There is none! Cloud Composer will take it from here...

[Return to Table of Contents](#)

## ***Additional Study Resources***

### **SQL deep dive**

- Course - SQL Primer
- <https://linuxacademy.com/cp/modules/view/id/52>

### **Machine Learning**

- Google Machine Learning Crash Course (free)
- <https://developers.google.com/machine-learning/crash-course/>

### **Hadoop**

- Hadoop Quick Start
- <https://linuxacademy.com/cp/modules/view/id/294>

### **Apache Beam (Dataflow)**

- Google's guide to designing your pipeline with Apache Beam (using Java)
- <https://cloud.google.com/dataflow/docs/guides/beam-creating-a-pipeline>