Linux Academy | Cloud Assessments

The Data Dossier

Choose a Section

linux Academy

Leonf.info@gmail.com

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)

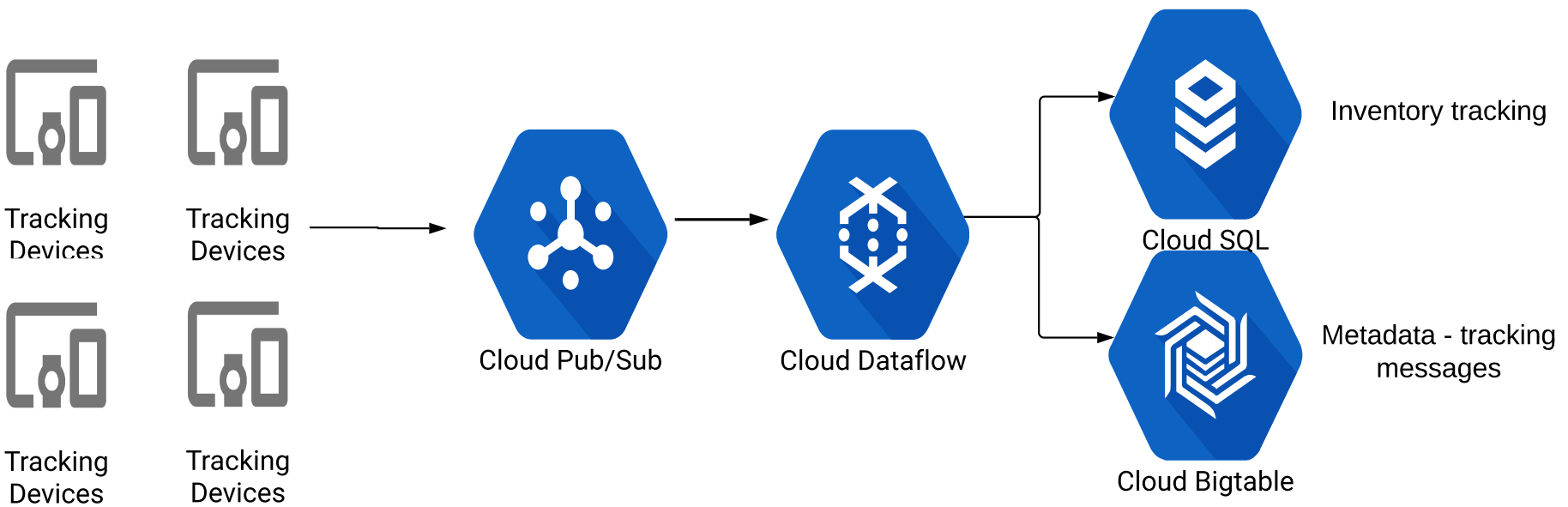
Return to Table of Contents

Flowlogistic Case Study

Previous
Next

Choose a Lesson

Inventory Tracking Data Flow



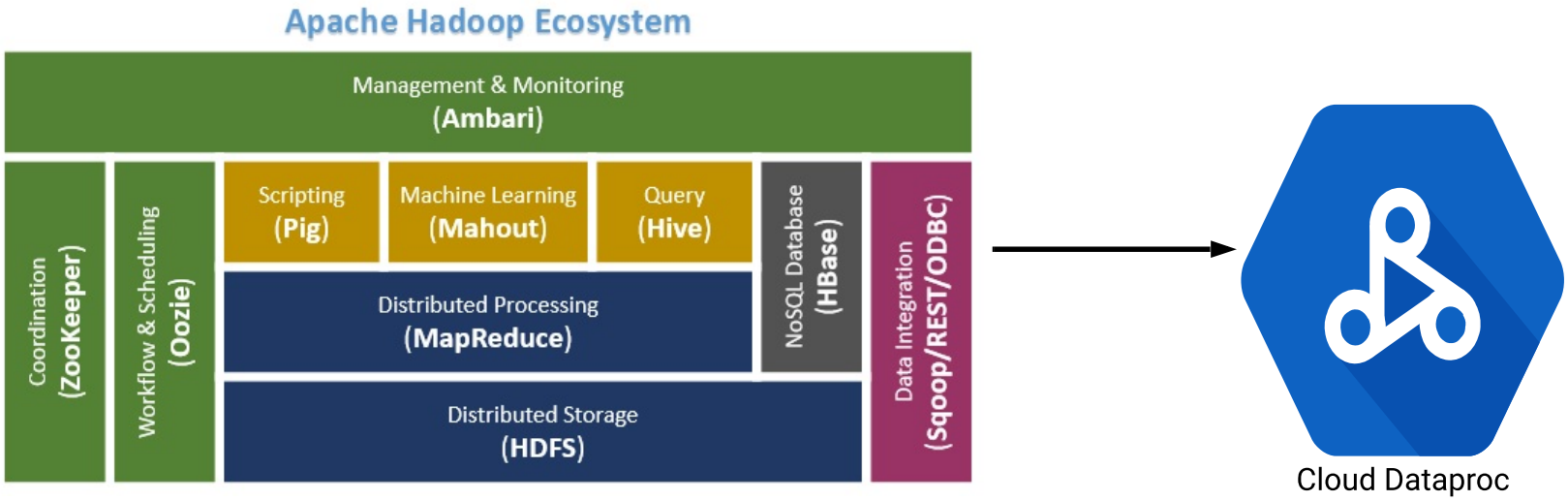
Return to Table of Contents

Flowlogistic Case Study

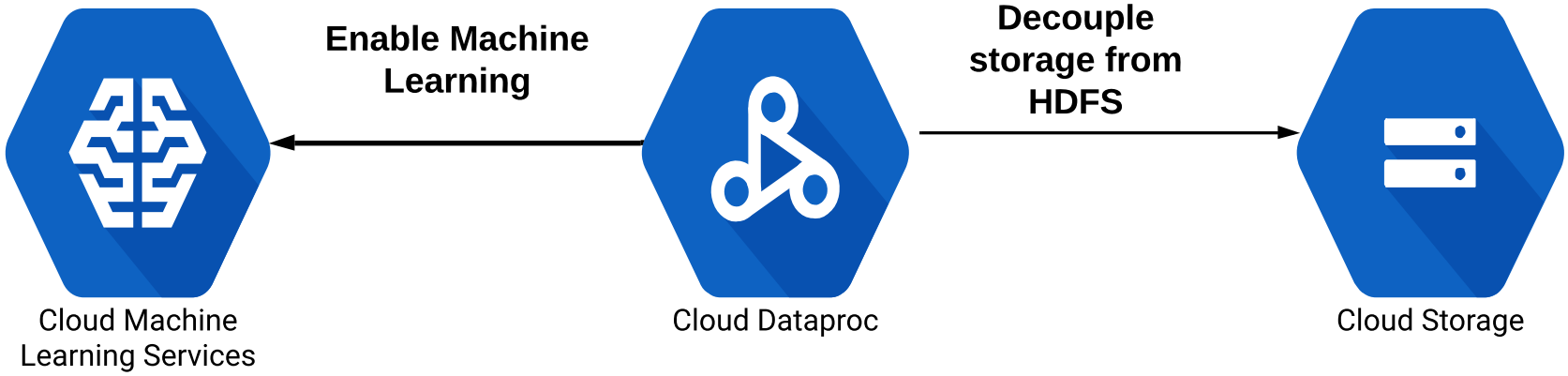
Previous

Choose a Lesson

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

- 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

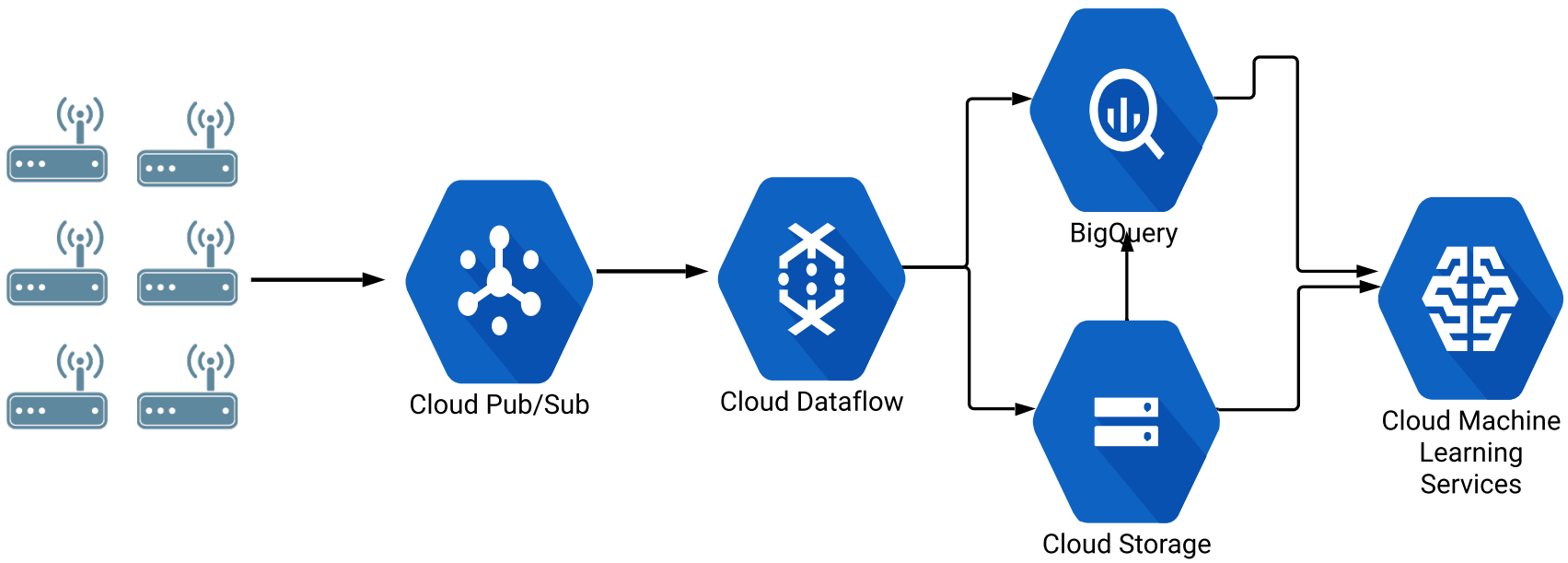
Case Study Overview

Flowlogistic

MJTelco

Previous

Data Flow Model



Return to Table of Contents

Choose a Lesson

Data Lifecycle

Batch and Streaming Data

Cloud Storage as Staging Ground

Database Types

Monitoring Unmanaged Databases

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

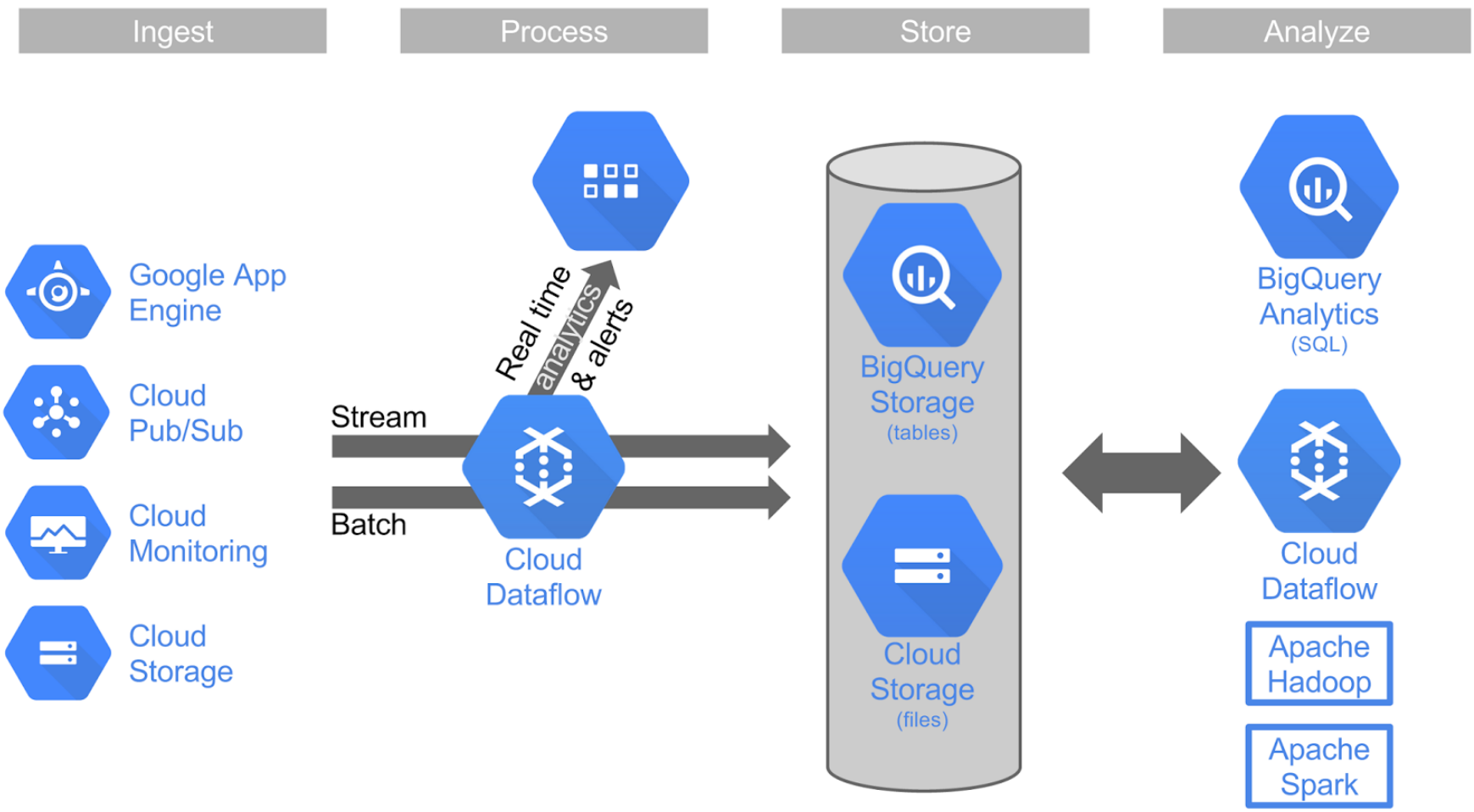
Choose a Lesson

- 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



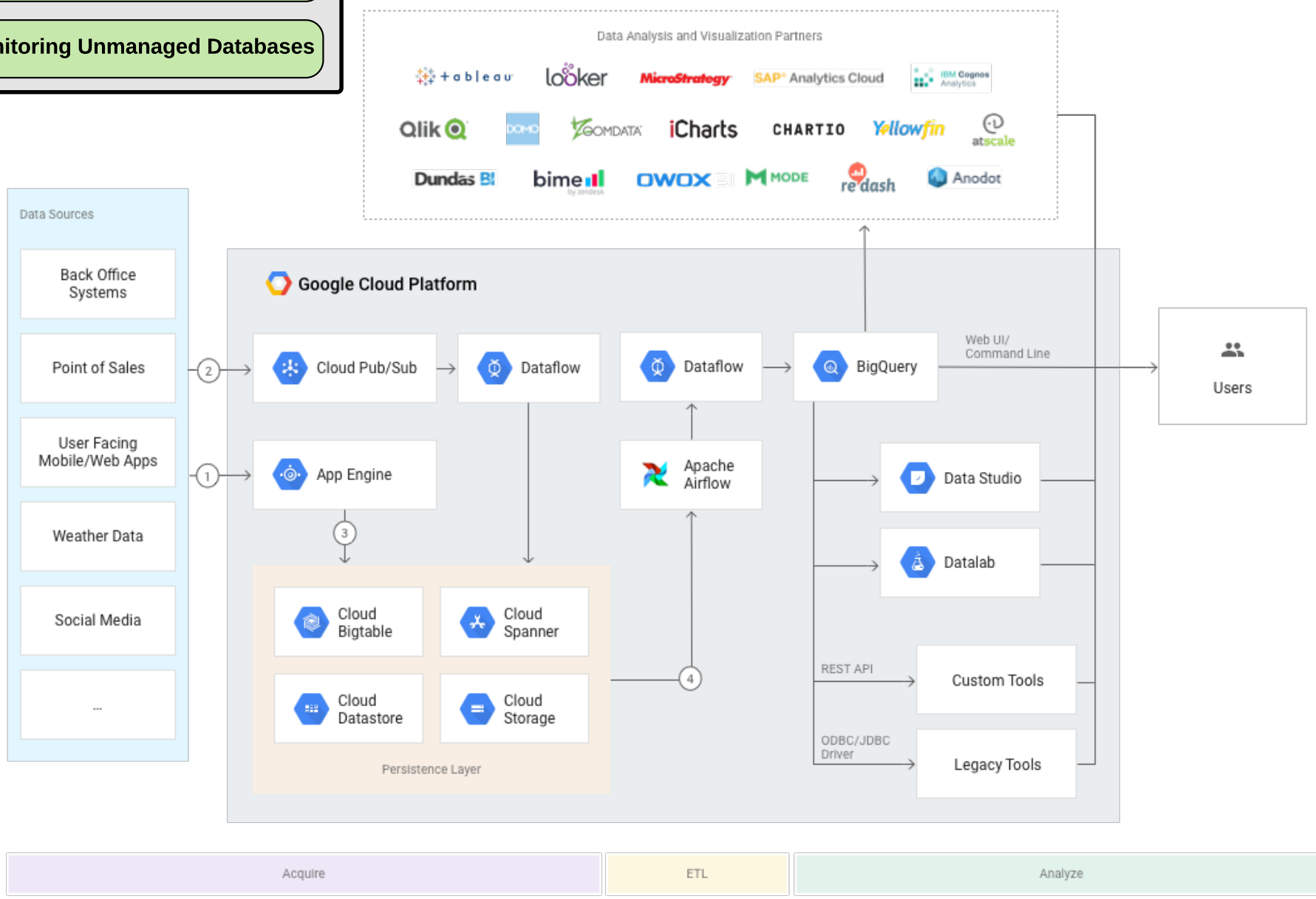
Return to Table of Contents

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

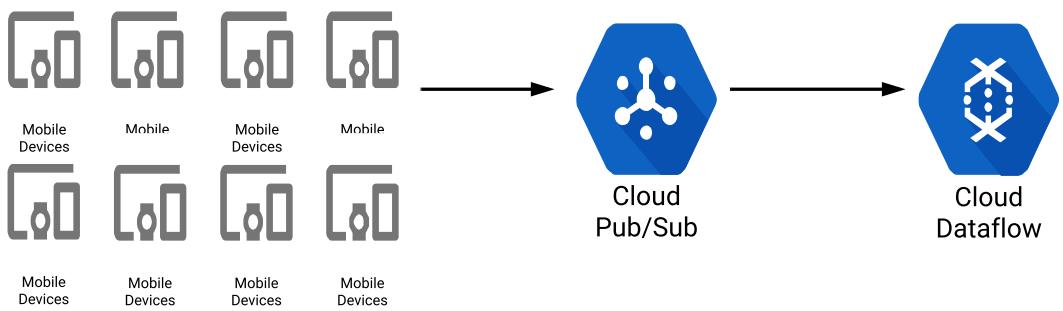
- Data Lifecycle
- Batch and Streaming Data
- Cloud Storage as Staging Ground
- Database Types
- Monitoring Unmanaged Databases

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



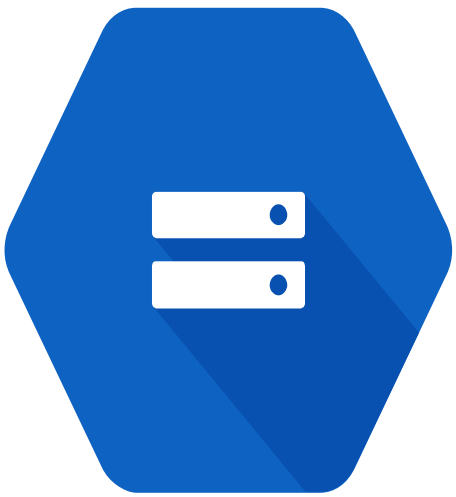
Batch (or bulk) data:

- 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

Choose a Lesson

- Data Lifecycle
- Batch and Streaming Data
- Cloud Storage as Staging Ground
- Database Types
- Monitoring Unmanaged Databases

Getting data in and out of Cloud Storage

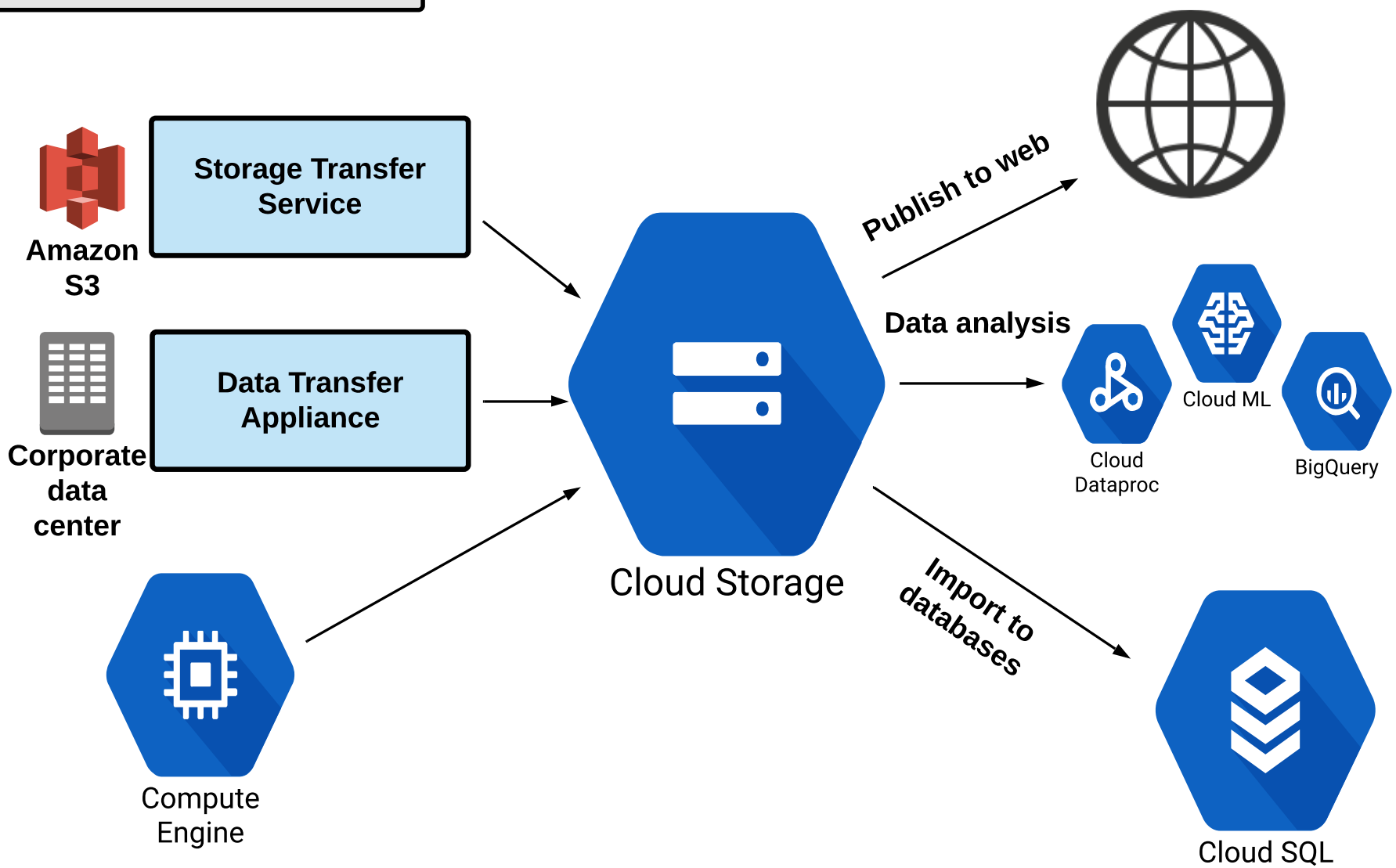
Previous

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

Database Types

Previous

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

Choosing a Managed Database

Next

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?

yes / no — data size

direct lift

shift transfer

1 2 3 4

Relational	Non-relational	Object - Unstructured	Data Warehouse

	Cloud SQL	Cloud Spanner	Cloud Datastore	Cloud Bigtable	Cloud Storage	BigQuery
Use Case	Structured data Web framework	RDBMS+scale High transactions	Semi-structured Key-value data	High throughput analytics	Unstructured data Holds everything	Mission critical apps Scale+consistency
e.g.	Medical records Blogs	Global supply chain Retail	Product catalog Game state	Graphs IoT Finance	Multimedia Analytics Disaster recovery	Large data analytics Processing using SQL

import
export

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

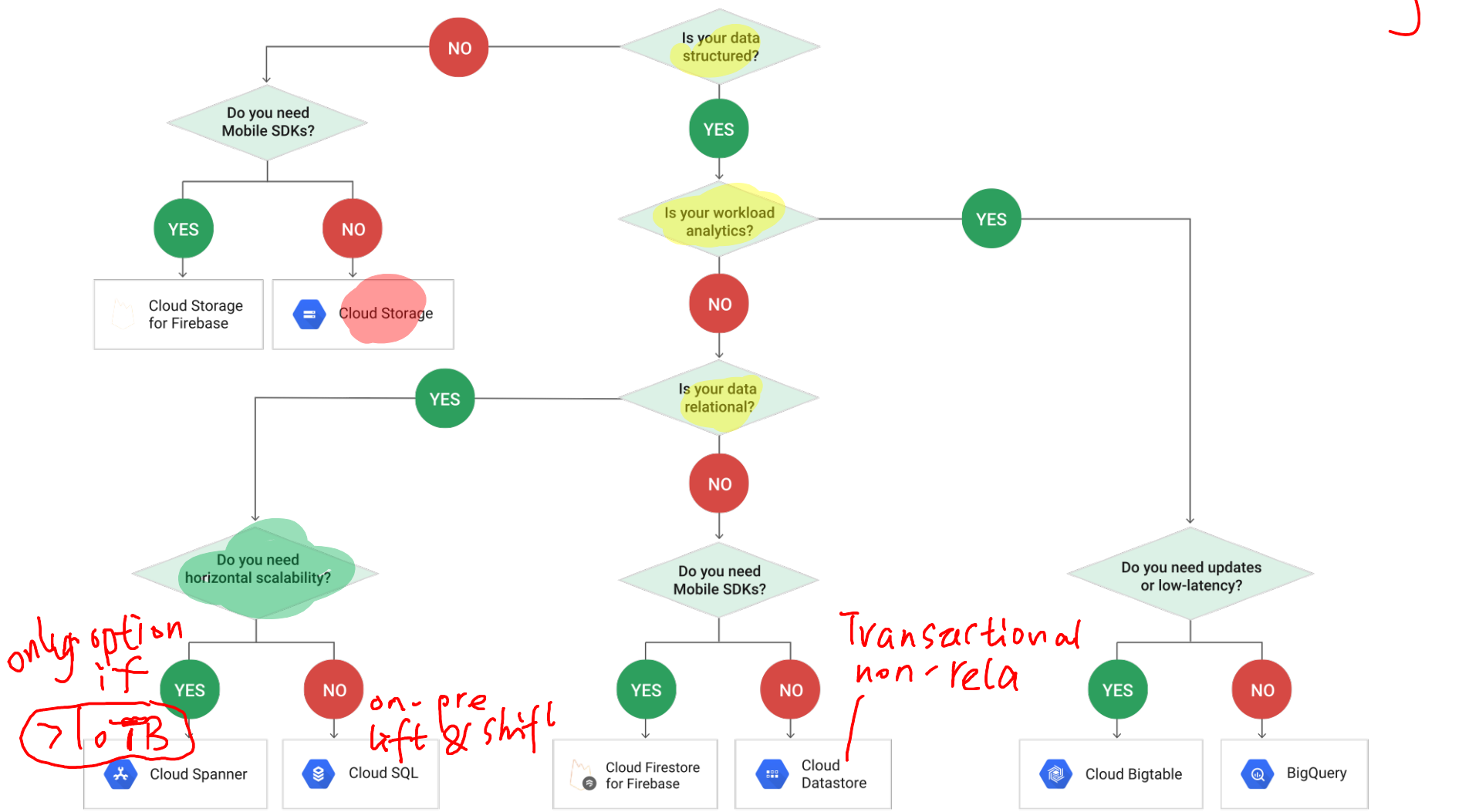
Previous

Decision tree criteria:

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

purpose

low large latency



only option if >10TB

on-pre lift & shift

Transactional non-rela

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?

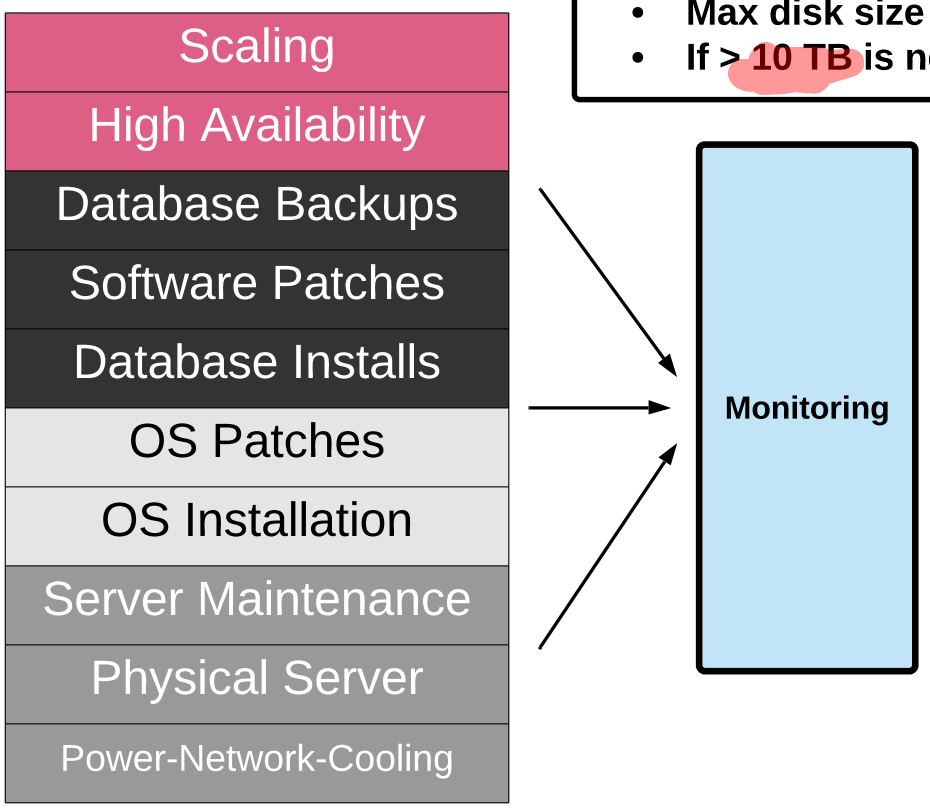
- **Direct lift and shift of traditional MySQL/PostgreSQL workloads with the maintenance stack managed for you**

What is managed?

- **OS installation/management**
- **Database installation/management**
- **Backups**
- **Scaling - disk space**
- **Availability:**
  - **Failover**
  - **Read replicas**
- **Monitoring**
- **Authorize network connections/proxy/use SSL**

Limitations:

- **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**



import/export
CSV file
to instan

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:

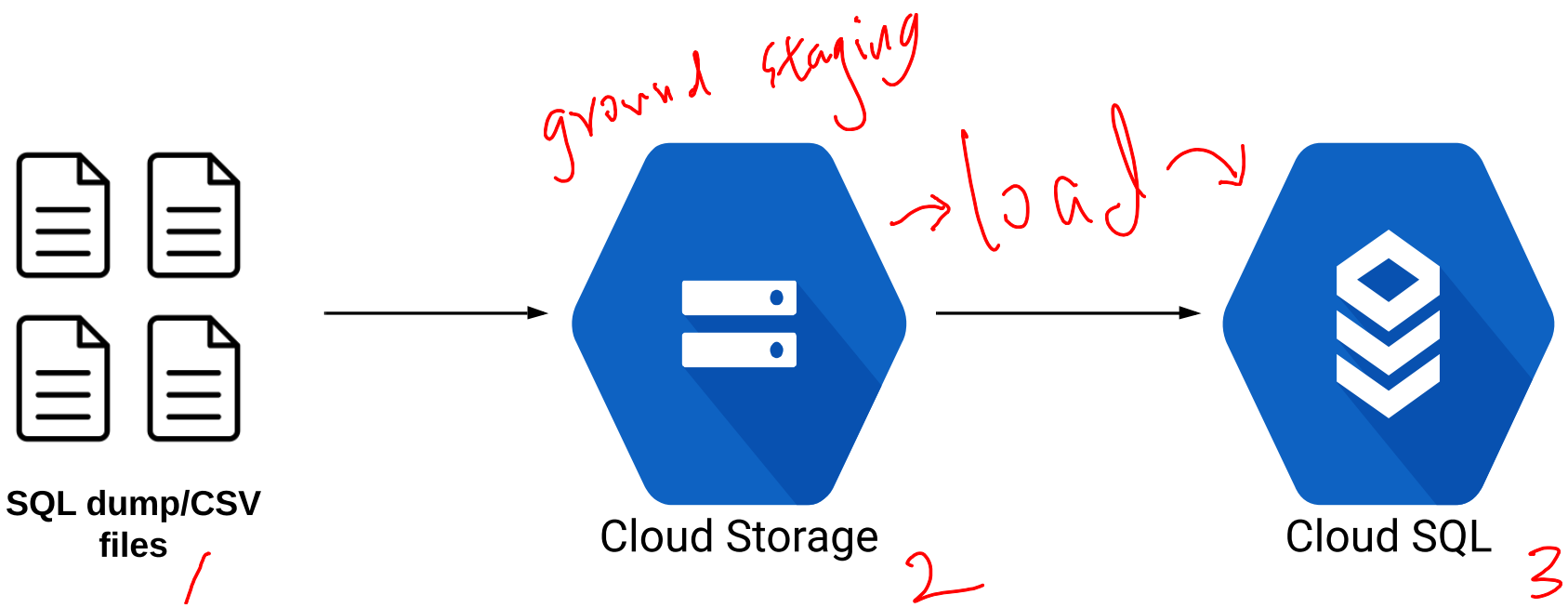
- Cloud Storage as a staging ground
- SQL dump/CSV file format

Export/Import process:

1. Export SQL dump/CSV file:
   - SQL dump file cannot contain triggers, views, stored procedures
2. Get dump/CSV file into Cloud Storage
3. 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

Questions:

SQL table grow over time

Larger table → slow query

Return to Table of Contents

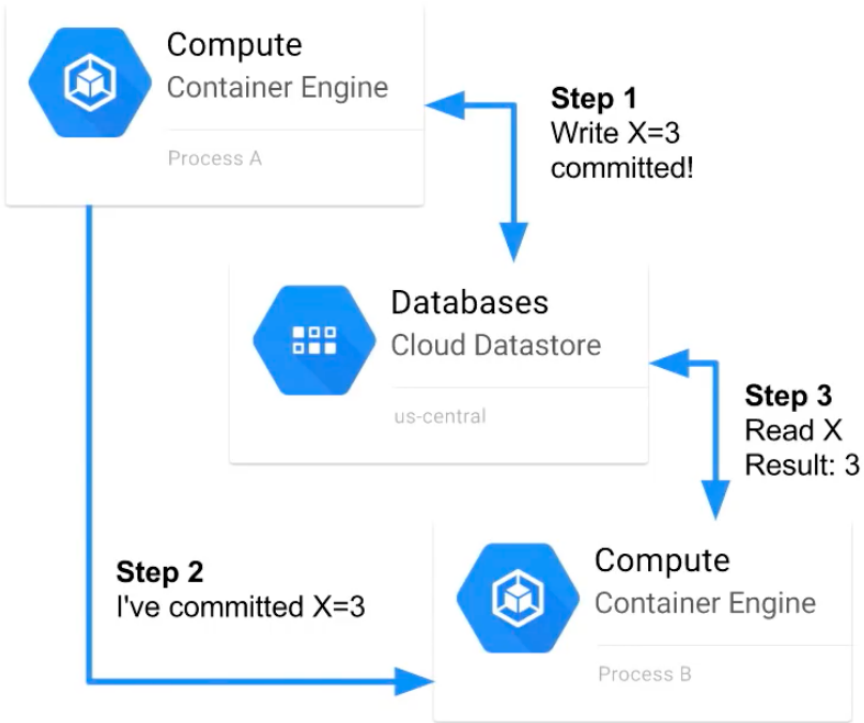
Choose a Lesson

- Cloud Datastore Overview
- Data Organization
- Queries and Indexing
- Data Consistency

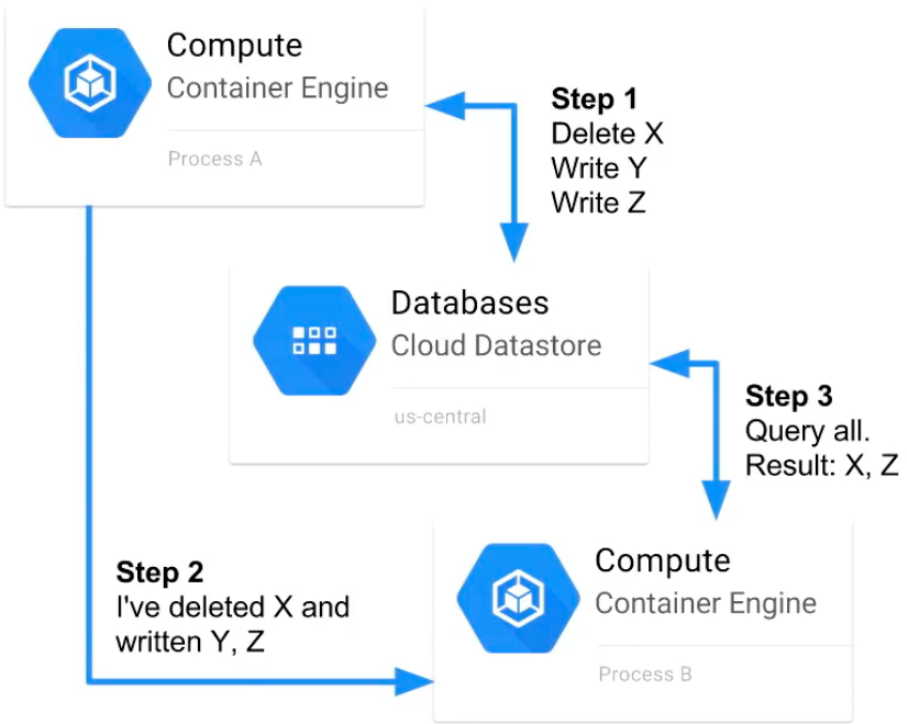
Data Consistency

Previous

Strong



Eventual



Return to Table of Contents

Choose a Lesson

- Cloud Datastore Overview
- Data Organization
- Queries and Indexing
- Data Consistency

(Cloud Datastore): Data Consistency

*Datastore is a NoOps Database

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

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

Queries and Indexing

Next

Query:

- Retrieve an entity from Datastore that meets a set of conditions
- Query includes:
  - Entity kind (table)
  - Filters ← "where" in SQL
  - Sort order
- Query methods:
  1. Programmatic
  2. Web console
  3. 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
```

example

Return to Table of Contents

Choose a Lesson

Cloud Datastore Overview

Data Organization

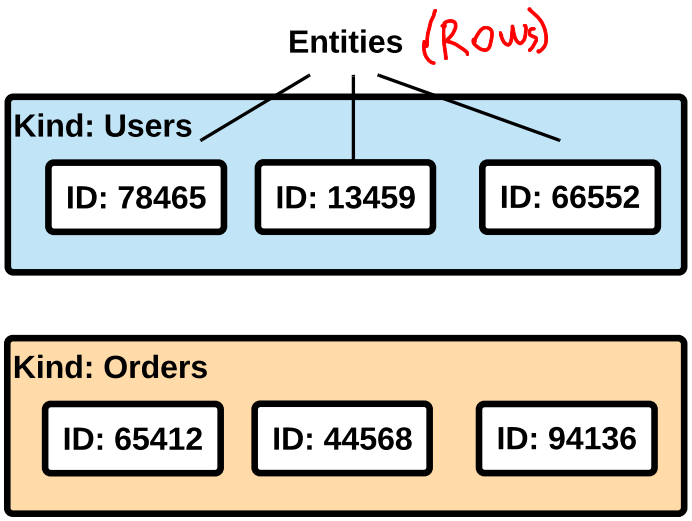
Queries and Indexing

Data Consistency

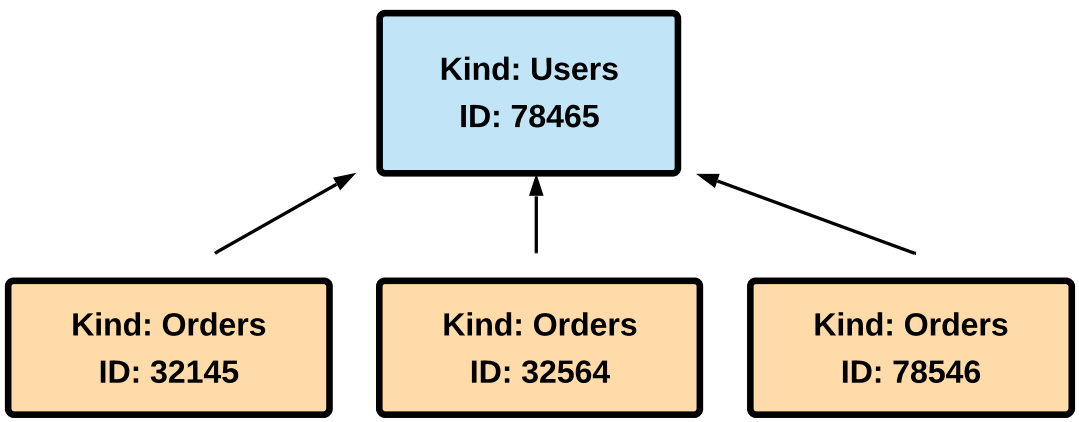
Data Organization

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

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

Key : Value stored

Concept	Relational Database	Datastore
Category of object	Table	Kind (category)
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

Cloud Datastore Overview

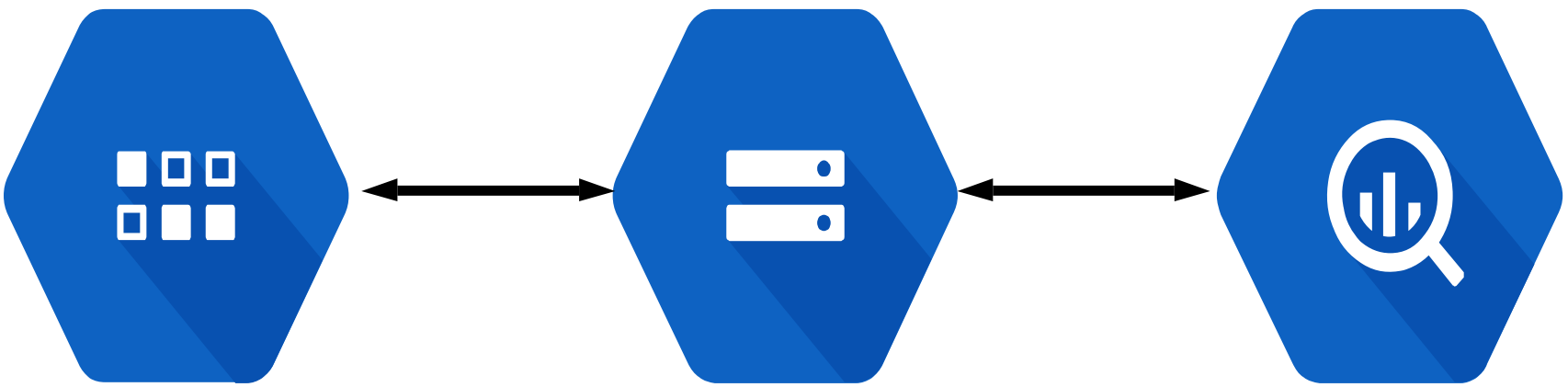
Previous

Other important facts:

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

Backup/Export/Import/Analyze

Managed export/import service

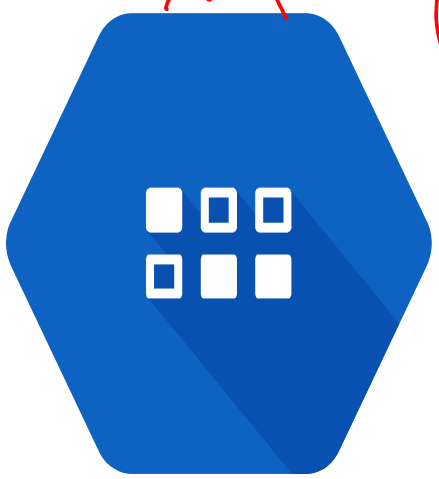


staging

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:
  - 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)

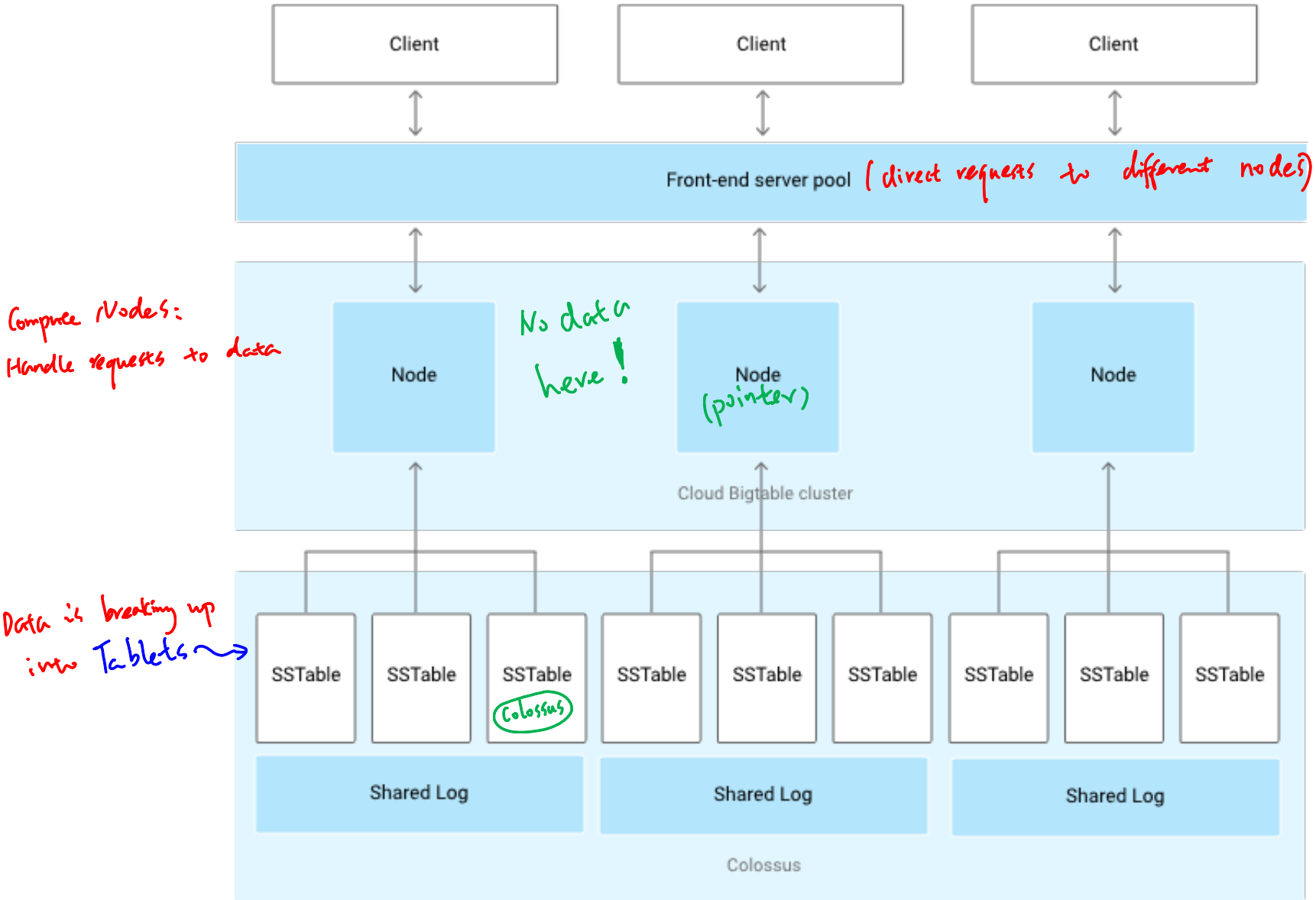
Return to Table of Contents

Cloud Bigtable Overview

Previous

Choose a Lesson

Cloud Bigtable Infrastructure



Return to Table of Contents

Choose a Lesson

- Cloud Bigtable Overview
- Instance Configuration
- Data Organization
- Schema Design

exam

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

> 1TB datasets

Use Cases

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

Access Control

(IAM access)

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

Cloud Bigtable

expensive, not for small business

Return to Table of Contents

Choose a Lesson

- Cloud Bigtable Overview
- Instance Configuration
- Data Organization
- Schema Design

Cloud Bigtable

Instance Configuration

Next

Instance basics

- 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 in a single cluster
    - No replication (for testing)
  - Production - 3+ nodes per cluster, also can create other clusters too
    - Replication available, throughput guarantee

Replication and Changes

- Synchronize data between clusters
  - One additional cluster, total
  - (Beta) available cross-region (may change later)
- Resizing
  - Add and remove nodes and clusters with no downtime
- Changing disk type (e.g. HDD to SSD) requires new instance
  1. export data to cloud storage
  2. create new instance
  3. re-import data again

Interacting with Bigtable

- Command line - cbt tool ① or HBase shell ②
  - cbt tool is simpler and preferred option

① How many nodes in your cluster?

② What type of disk you use?

Return to Table of Contents

Choose a Lesson

- Cloud Bigtable Overview
- Instance Configuration
- Data Organization
- Schema Design

cbt : we're interacting with bigtable

Cloud Bigtable

Instance Configuration

Previous

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 (family name is 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'

Memorize the page ↑

Return to Table of Contents

Choose a Lesson

- Cloud Bigtable Overview
- Instance Configuration
- Data Organization
- Schema Design

Data Organization

How data is organized in big table

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

	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
- Schema design is necessary for efficient queries!
- Field promotion - move fields from column data to row key

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

Row Key
memusage+user+timestamp
20-mattu-201805082048

good schema for row key

Schema Design (exam topic)

choose the best schema for the row key for max out of efficiency.

Schema Design

1. Per table – Row key is the only indexed item
2. Keep all entity info in a single row
3. Related entities should be in adjacent rows
   - More efficient reads
4. Tables are sparse – empty columns take no space

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
  1. Reverse domain names (com.linuxacademy.support)
  2. String identifiers (mattu)
  3. Timestamps (reverse, NOT at front/or only identifier) Y-m-d
- Poor row keys = hotspotting
  - Domain names (support.linuxacademy.com)
  - Sequential ID's
  - Timestamps alone/at front

Table Design - Time Series Data

{ tall and narrow / short and wide

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

Return to Table of Contents

Choose a Lesson

Cloud Spanner Overview

Data Organization and Schema

Cloud Spanner Overview

a "kind" of SQL

Next

What is Cloud Spanner?

- Fully managed, highly scalable/available, relational database
- Similar architecture to Bigtable
- "NewSQL" is used for strong transactional consistency or ACID compliance

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 resource on single instance (CPU/RAM)
- 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/PostreSQL compatible, so not a lift-and-shift version of SQL
  - 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

trade-off

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

best for both ∞ database

Return to Table of Contents

Choose a Lesson

Cloud Spanner Overview

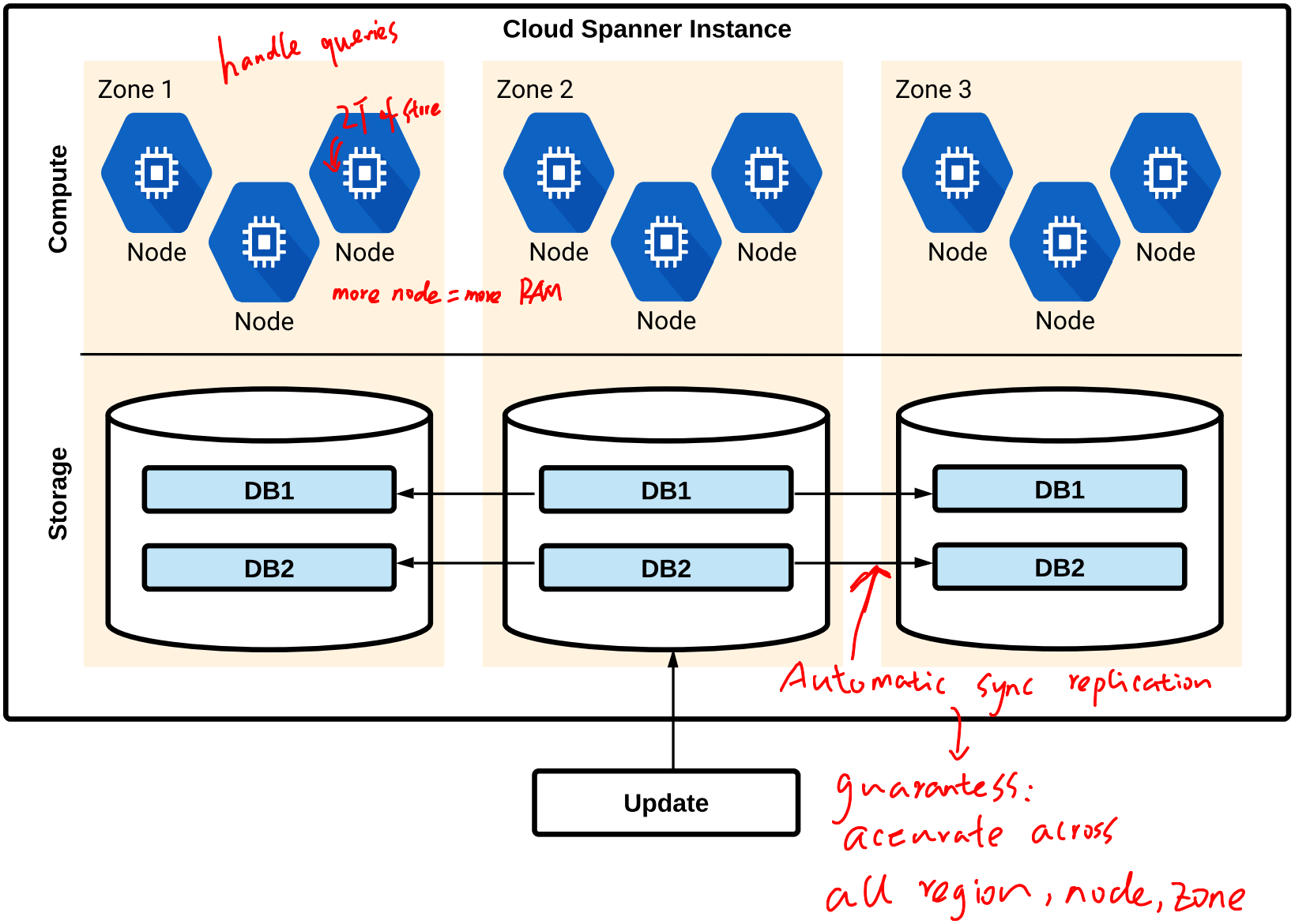
Data Organization and Schema

Cloud Spanner Overview

Previous

Next

Cloud Spanner Architecture
(similar to Bigtable)



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. (role name)
  1. Admin - Full access to all Spanner resources
  2. Database Admin - Create/edit/delete databases, grant access to databases
  3. Database Reader - read/execute database/schema
  4. Viewer - view instances and databases
     - Cannot modify or read from database

Roles

* IAM Questions will be on the exam for any given services

The Data Dossier

Return to Table of Contents

Choose a Lesson

Data Organization and Schema

Organization

Next

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

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

Spanner Interleave Tables for 3 Tables

parent → child → child of child

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"

Return to Table of Contents

Choose a Lesson

Cloud Spanner Overview

Data Organization and Schema

Data Organization and Schema

Previous

Primary keys and Schema

- How to tell which child tables to store with which parent tables
- Usually a natural fit
  - 'Customer ID'
  - 'Invoice ID' (child table)
- Avoid hotspotting – like bigtable
  - No sequential numbers for primary key
  - No timestamps (also sequential)
    - Use descending order if timestamps required

2 Recommends

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

Next

What is Streaming Data?

- "Unbounded" data vs batch data (Bounded)
- Infinite, never completes, always flowing, never ending

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

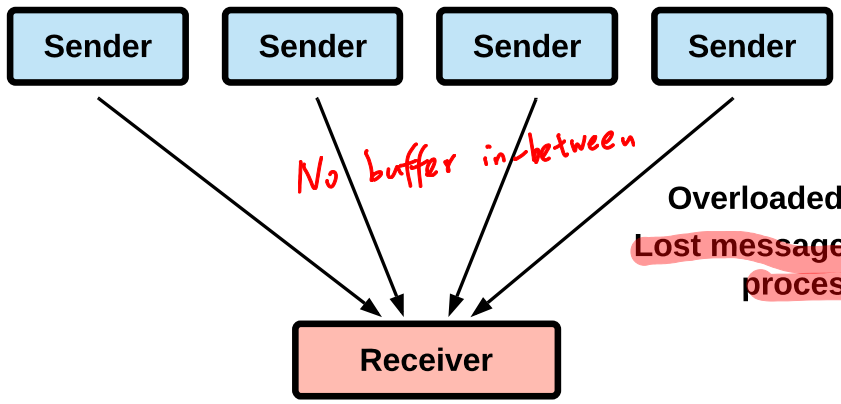
Streaming Data Challenges

Previous

Tight vs. Loose Coupling in Systems

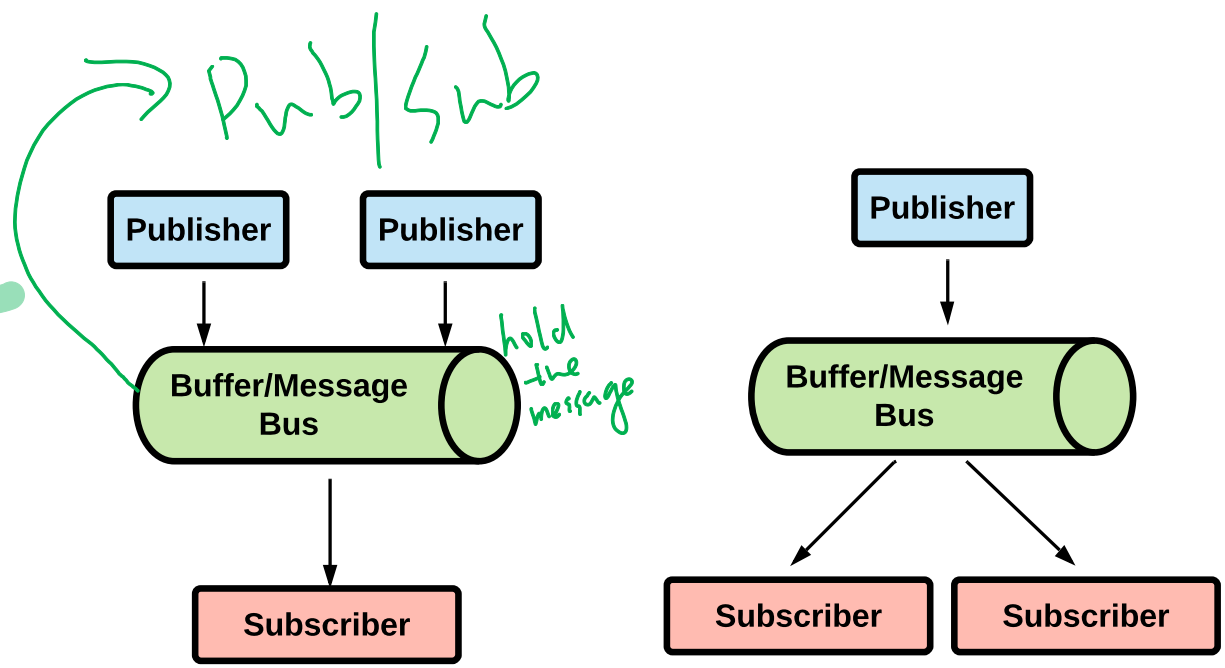
- Tightly (direct) coupled systems more likely to fail (overloaded)
- Loosely coupled systems with 'buffer' scale have better fault tolerance

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

No ops → No second figure; No cluster to provision

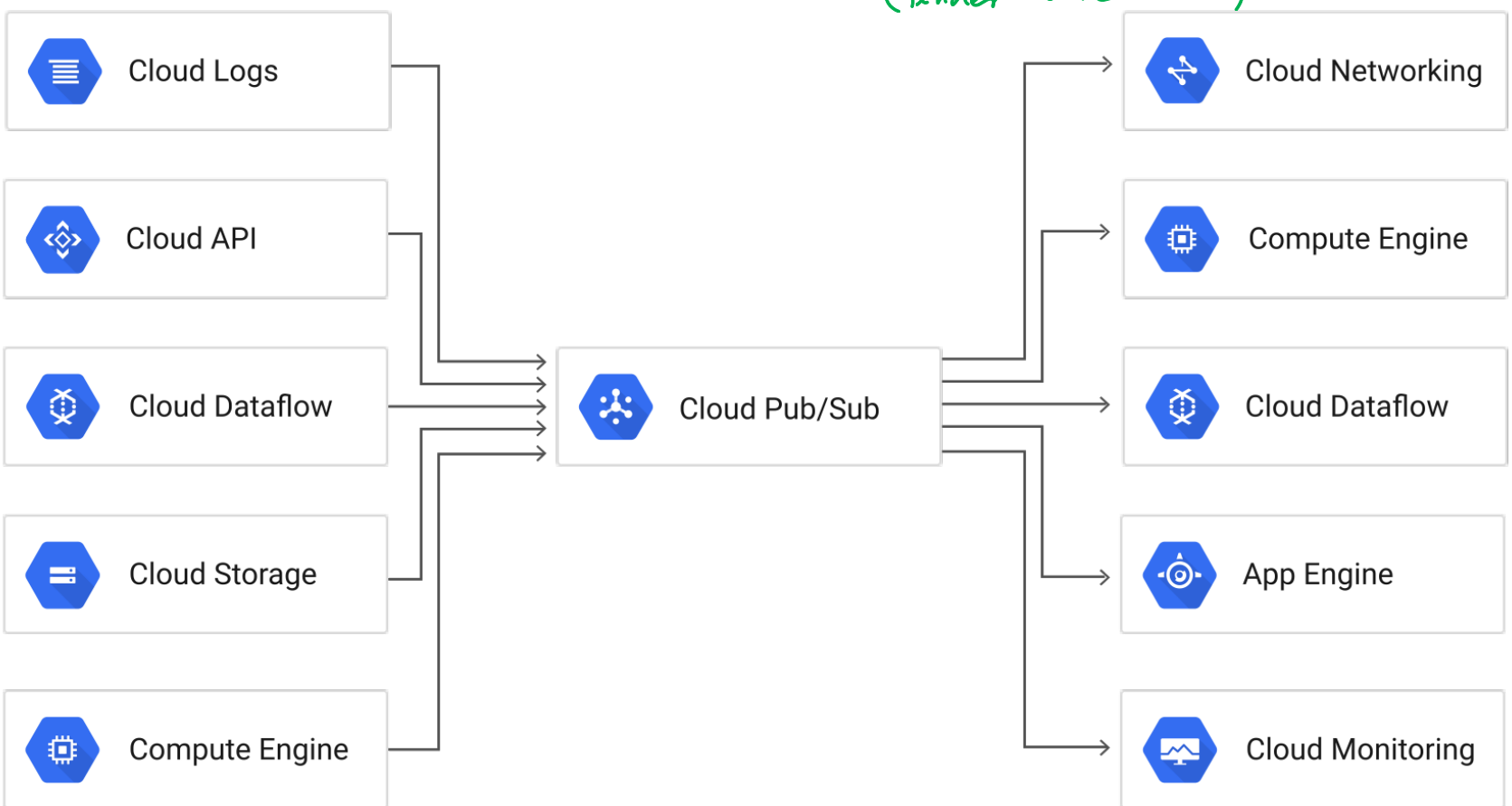
exam question

Terminology ↓

1 to 1
many to many
1 to 8, 6 to 2, doesn't matter

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

(Sender to reciever)



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

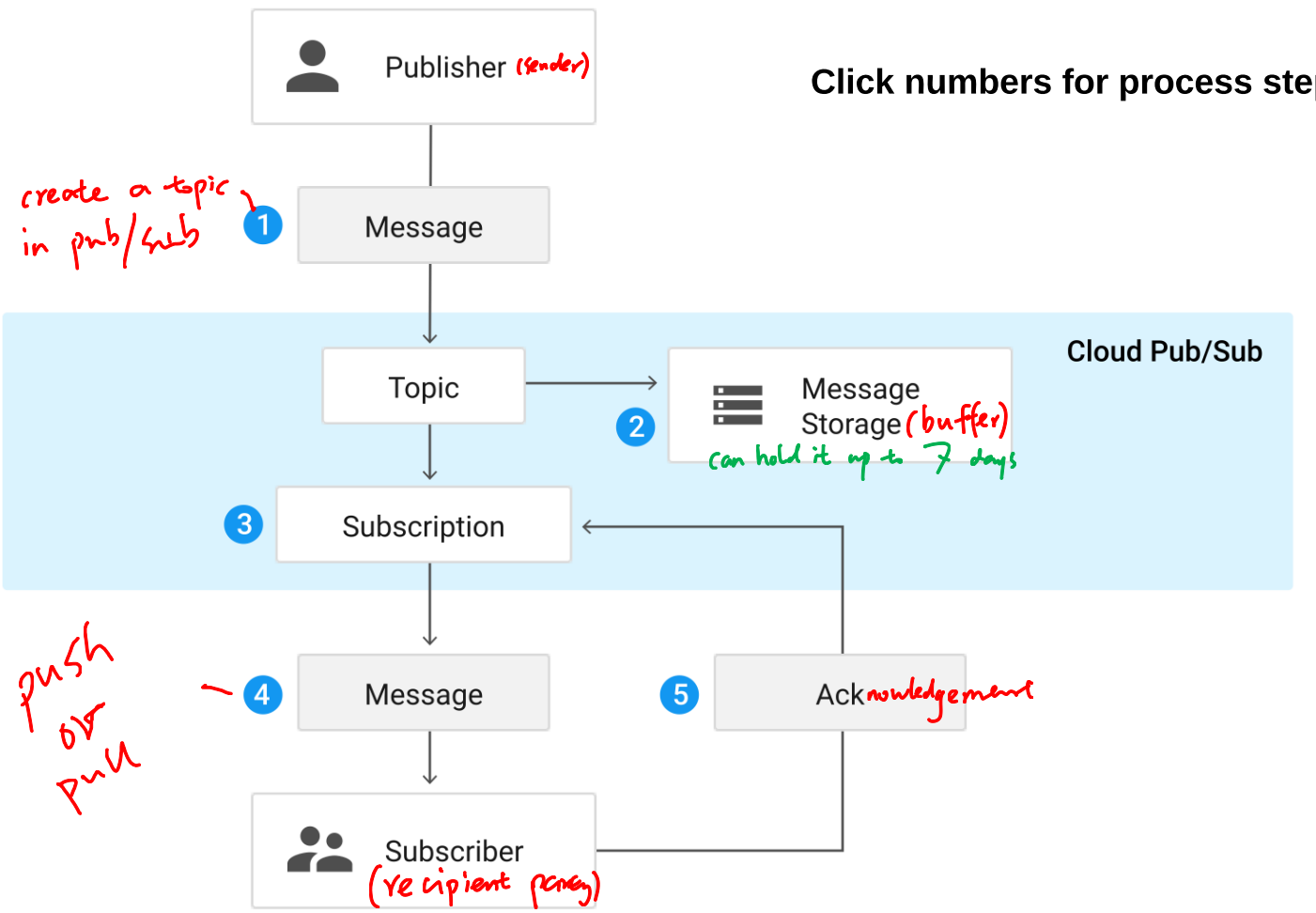
Cloud Pub/Sub Overview

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

Cloud Pub/Sub Overview

Previous

Next

Push and Pull (maybe on exam)

- 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

pull is the default option

IAM (exam)

- Allows for controlling access at project, topic, or subscription level
- Admin, Editor, Publisher, Subscriber
- Service accounts are best practice (especially for built-ins an application)

Pricing

- Data volume used per month (per GB)

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

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.

(Pub/Sub doesn't)

exam

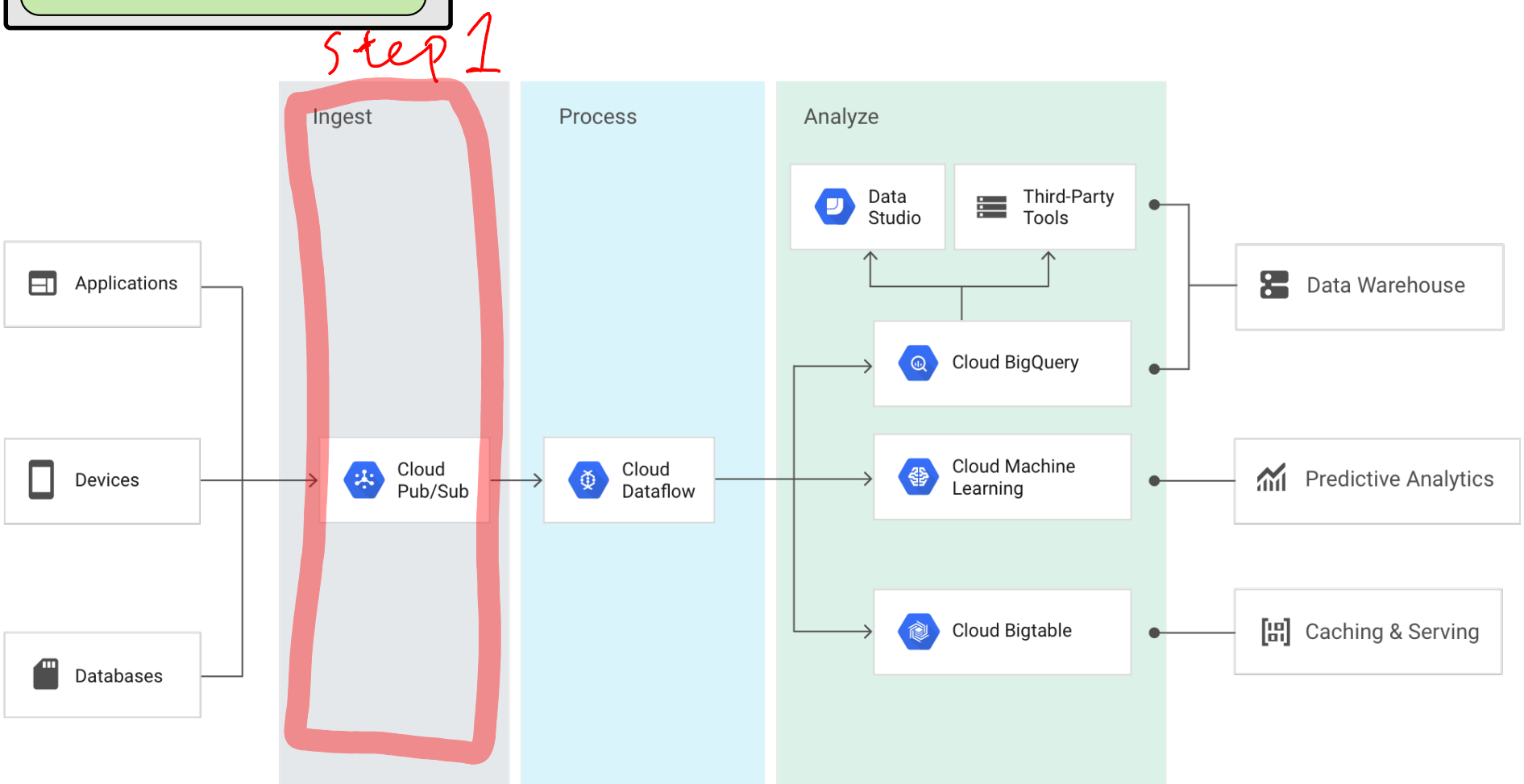
Return to Table of Contents

Cloud Pub/Sub Overview

Previous

Choose a Lesson

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

Pub/Sub Hands On

Next

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

Next

Does Pub/Sub Replace Kafka?

- 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?

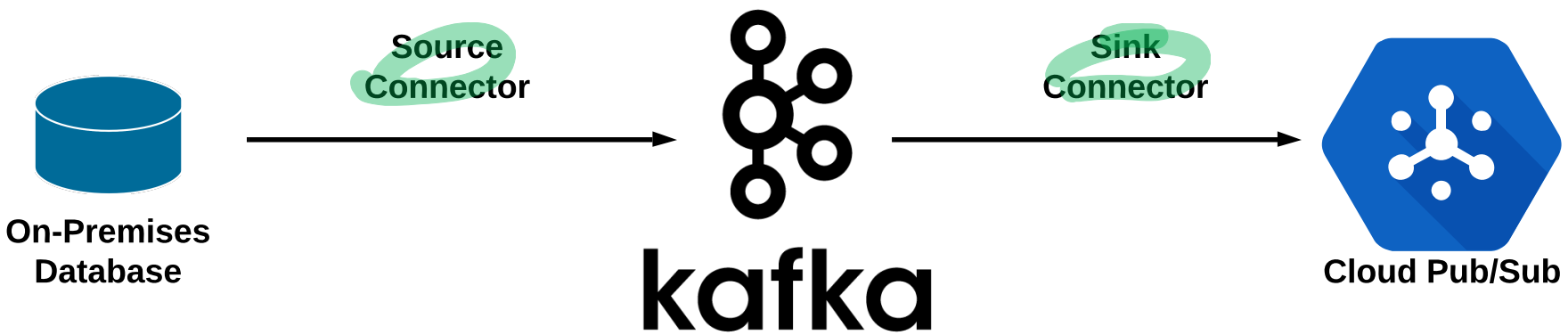
Overview on Connectors:

- 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

exam related terminology

- **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

Trouble shooting

In a Perfect World...

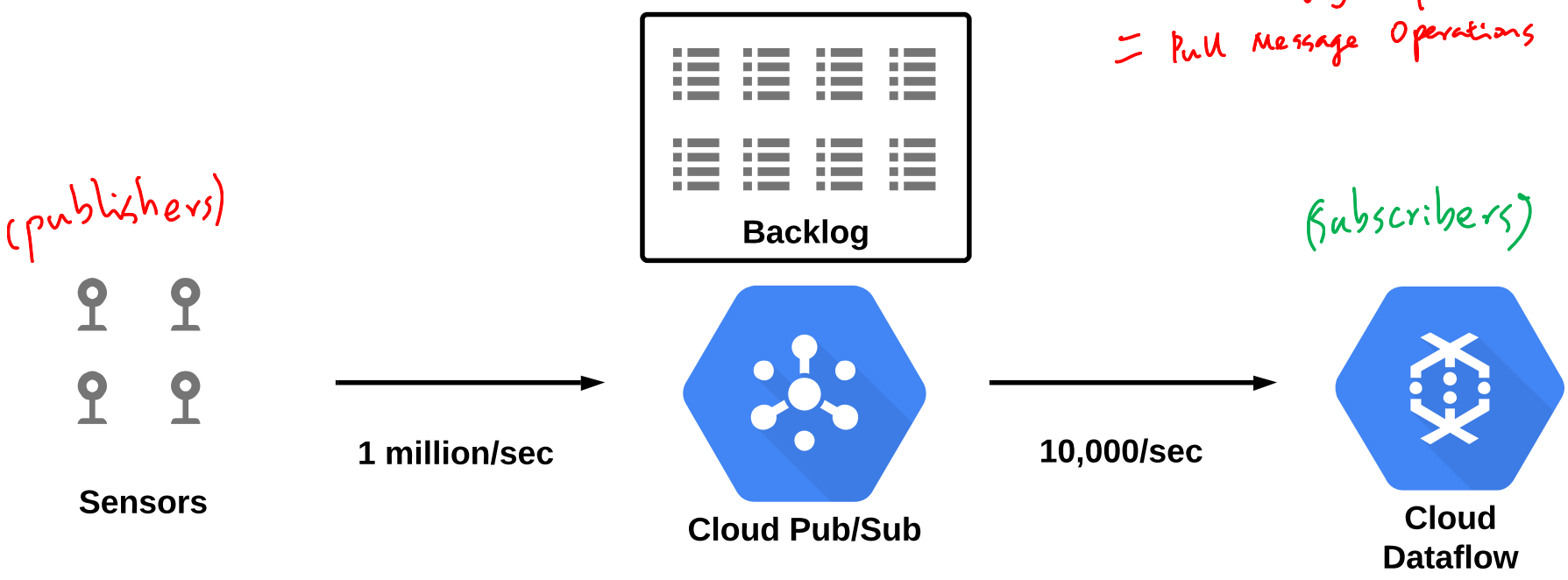
- Subscribers and Publishers work in perfect harmony:
  - Example:
    - 1 million messages/second published
    - 1 million messages/second successfully pulled/pushed
    - Result: No backlog in Pub/Sub queue
- But we don't live in a perfect world...
  - Subscriber cannot keep up with publish rate
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

Troubleshooting Subscriber Health (Backlog)

4 trouble shootings

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

acknowledging Requests = Pull message Operations



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

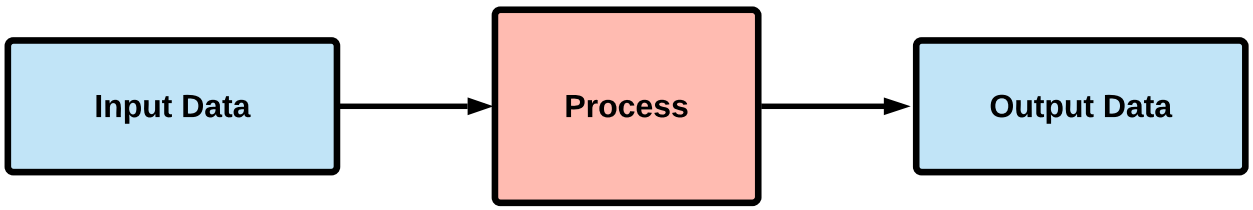
Data Processing Challenges

Next

What is Data Processing?

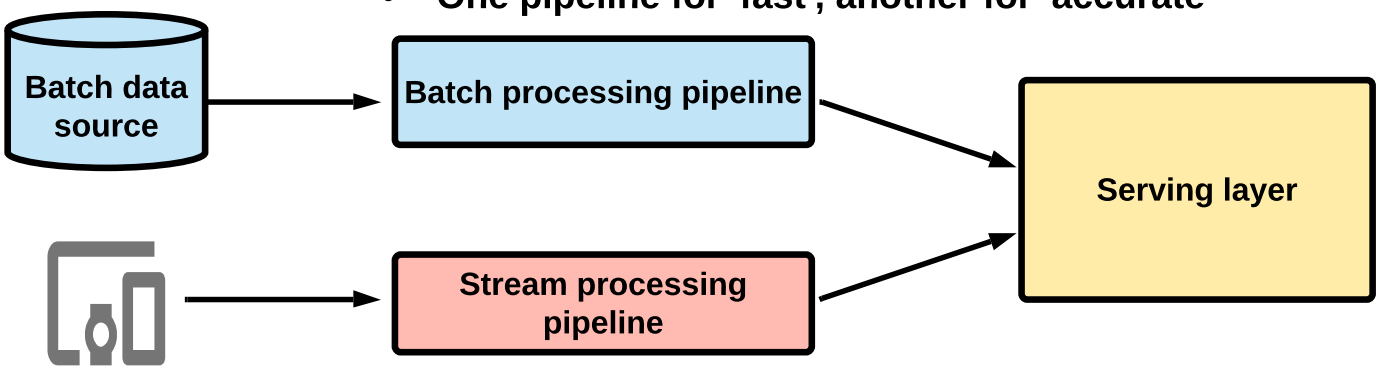
- Read Data (Input)
- Transform it to be relevant - Extract, Transform, and Load (ETL)
- Create output

to our particular interest



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

Data Processing Challenges

Previous

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

create batch and data processing pipeline

Cloud Dataflow

execute pipelines

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

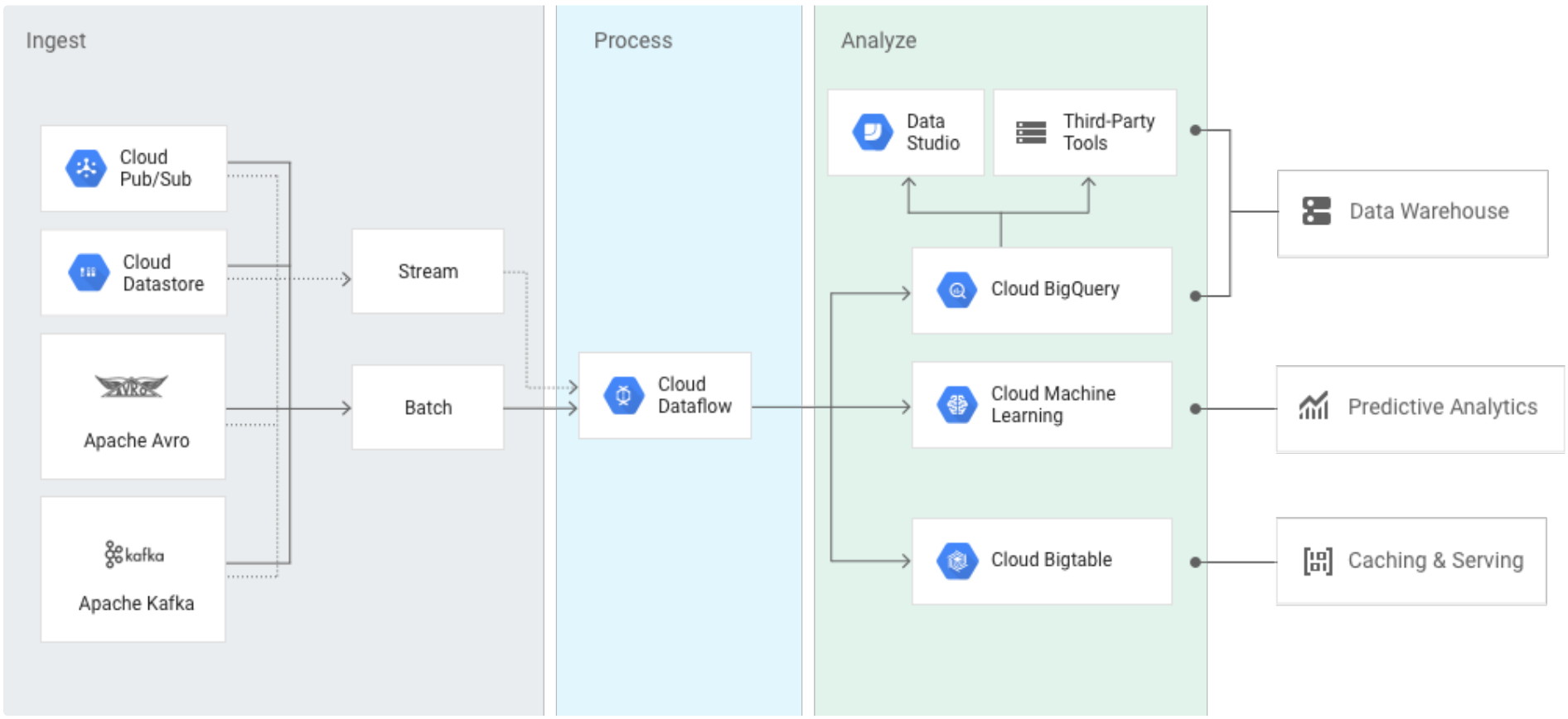
Cloud Dataflow Overview

Next

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)

Dataproc can not do!

Return to Table of Contents

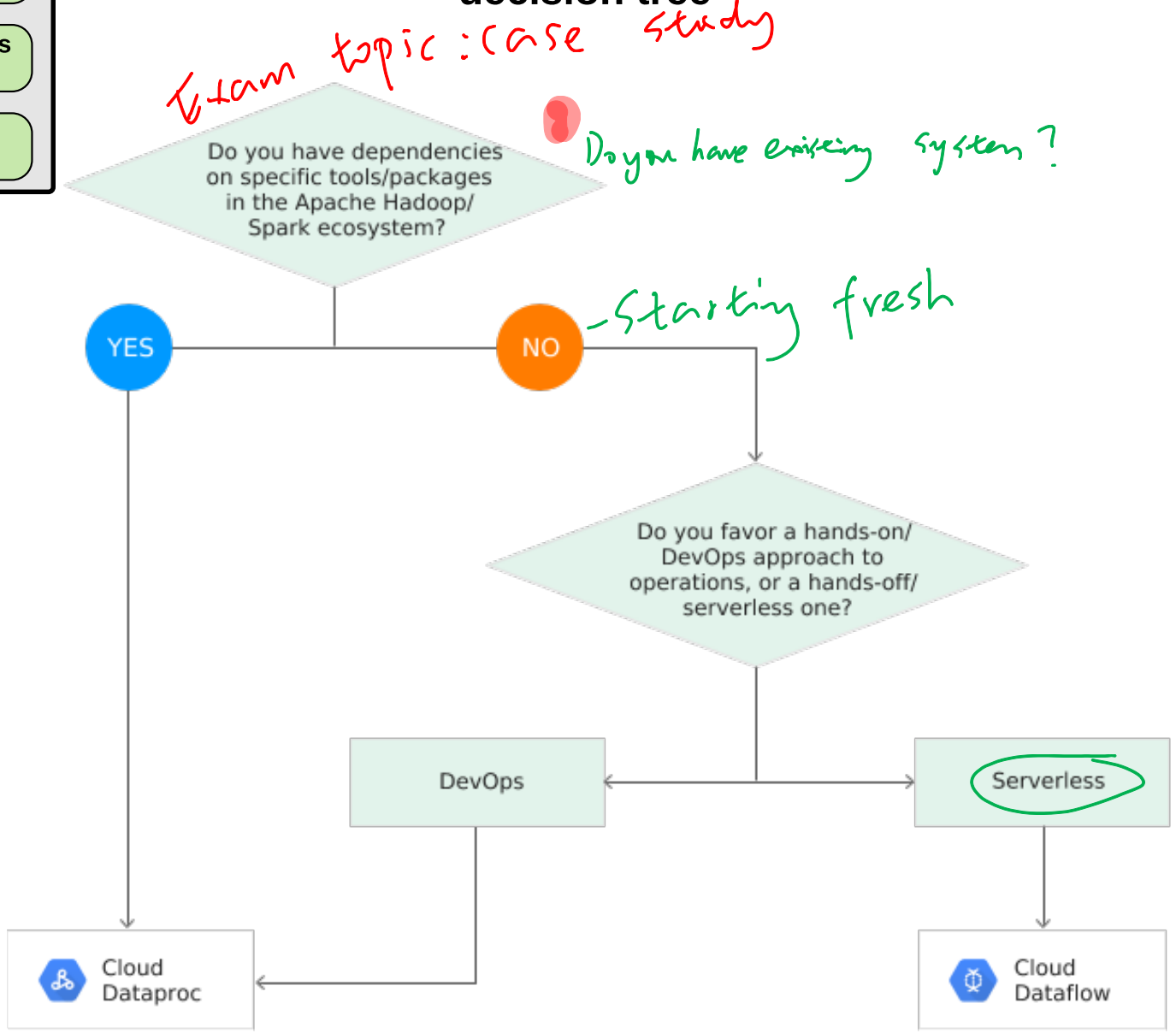
Choose a Lesson

- 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

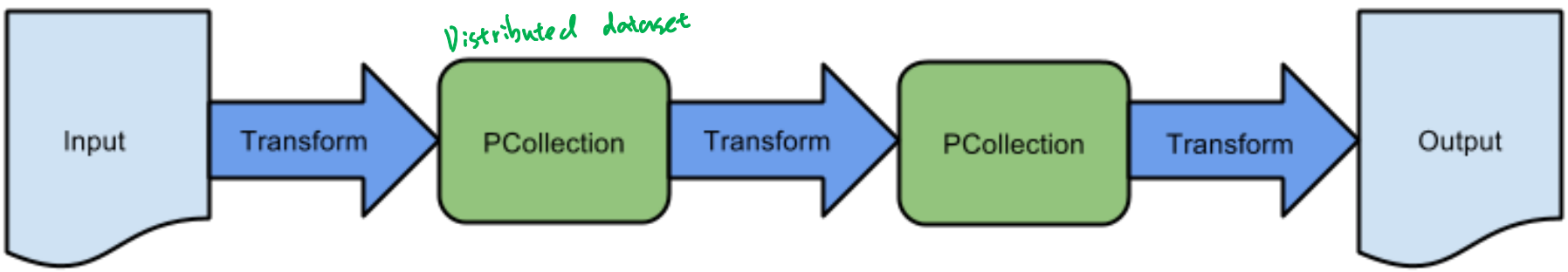
Next

Course/exam perspective:

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

Key terms: (exam)

- 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.newLaneInfo(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

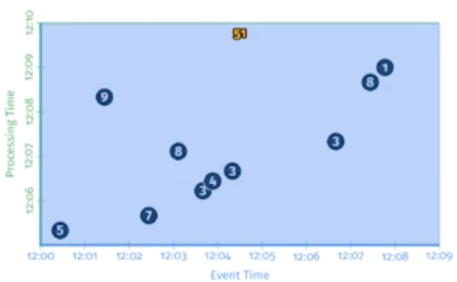
Cloud Dataflow Overview

Previous

(streaming data)

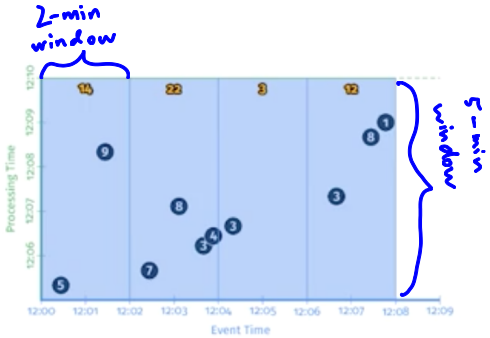
Dealing with late/out of order data:

- 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



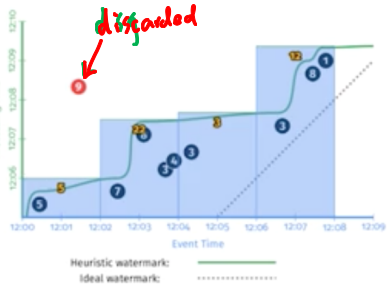
1

Classic Batch



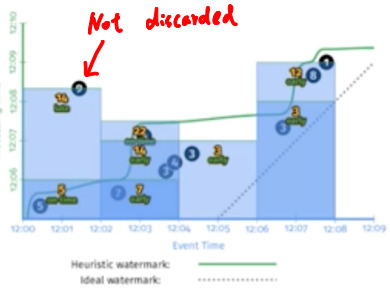
2

Windowed Batch



3

Streaming w/Discarding



4

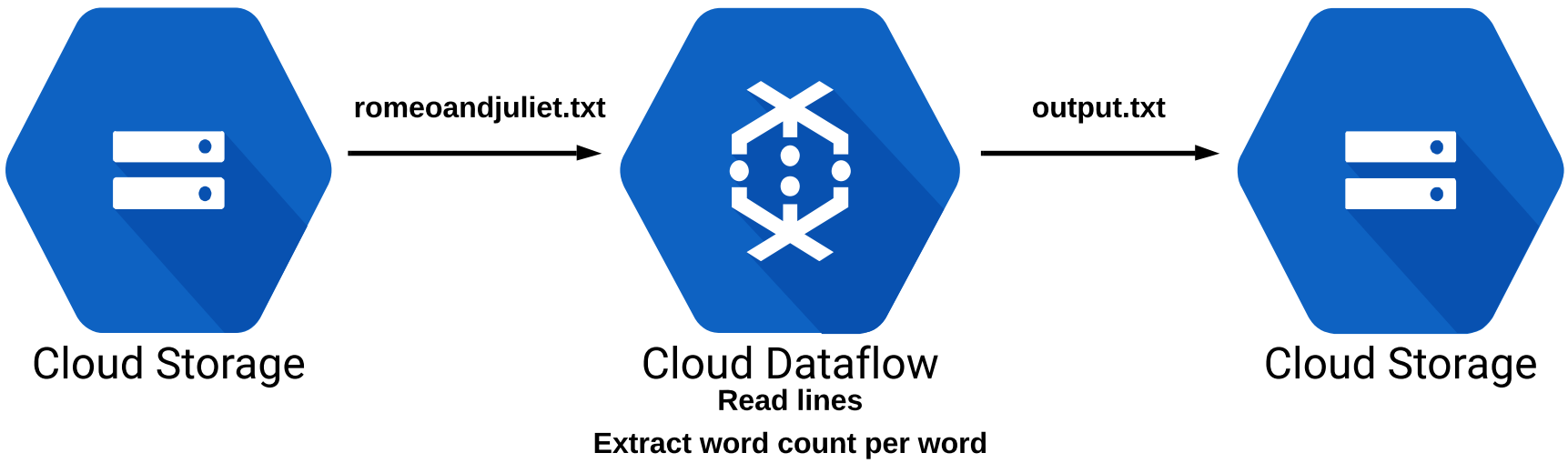
Streaming + Accumulation

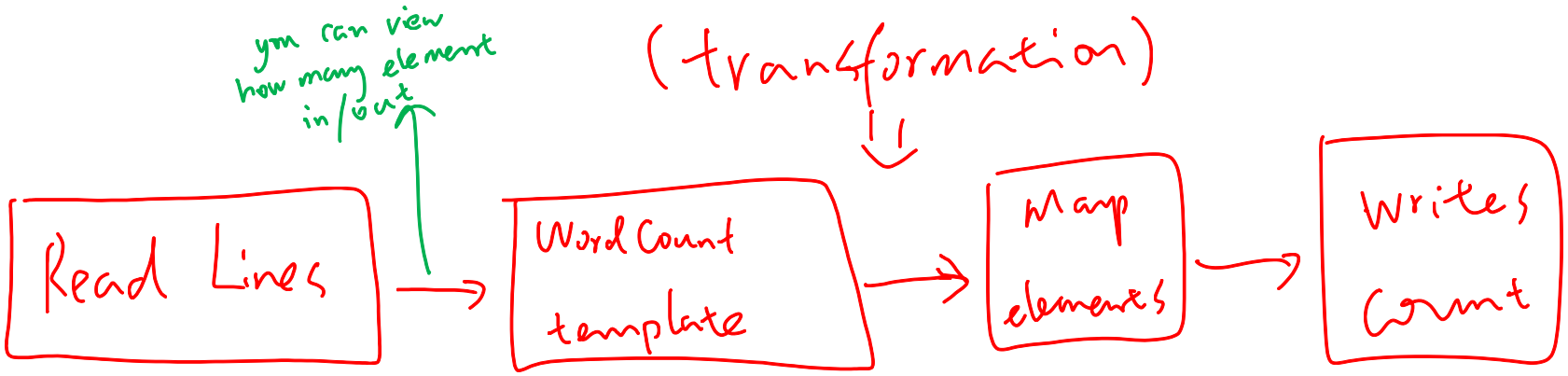
Return to Table of Contents

Choose a Lesson

Template Hands On

- Google-provided templates
- Simple word count extraction





Return to Table of Contents

Choose a Lesson

Streaming Ingest Pipeline Hands On

Next

- 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

Traffic data

Data ingest

Published streaming sensor data

Topic: sandiego

Subscription pulls messages

Cloud 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

ms = 1 min

Shut down pipeline:

- Drain - finishing processing buffered jobs before shutting down (Stop accepting new data, and shut down gracefully)
- 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

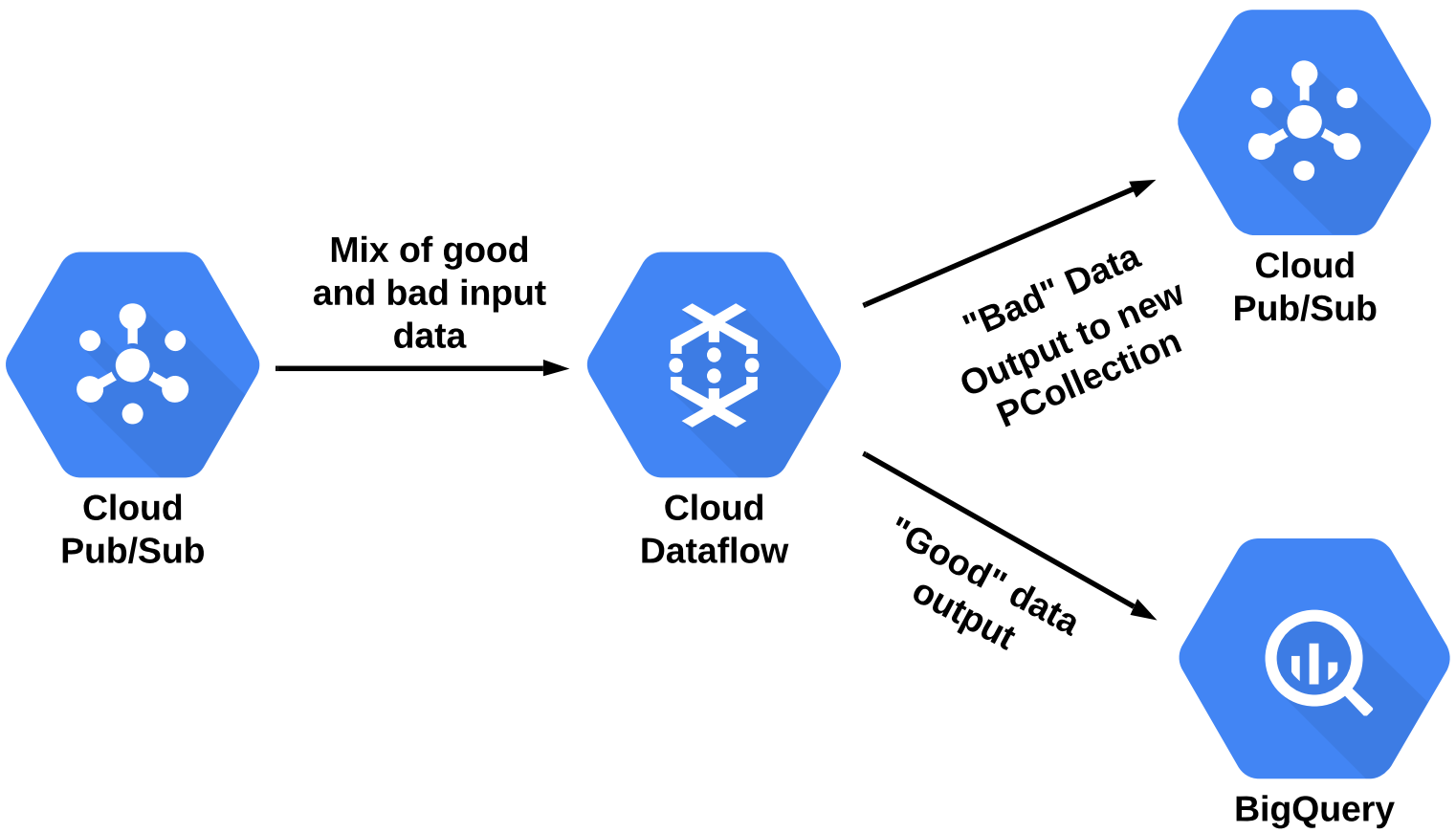
Additional Best Practices

Additional Best Practices

Handling Pipeline Errors

Next

- 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: (in Apache)
    - **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

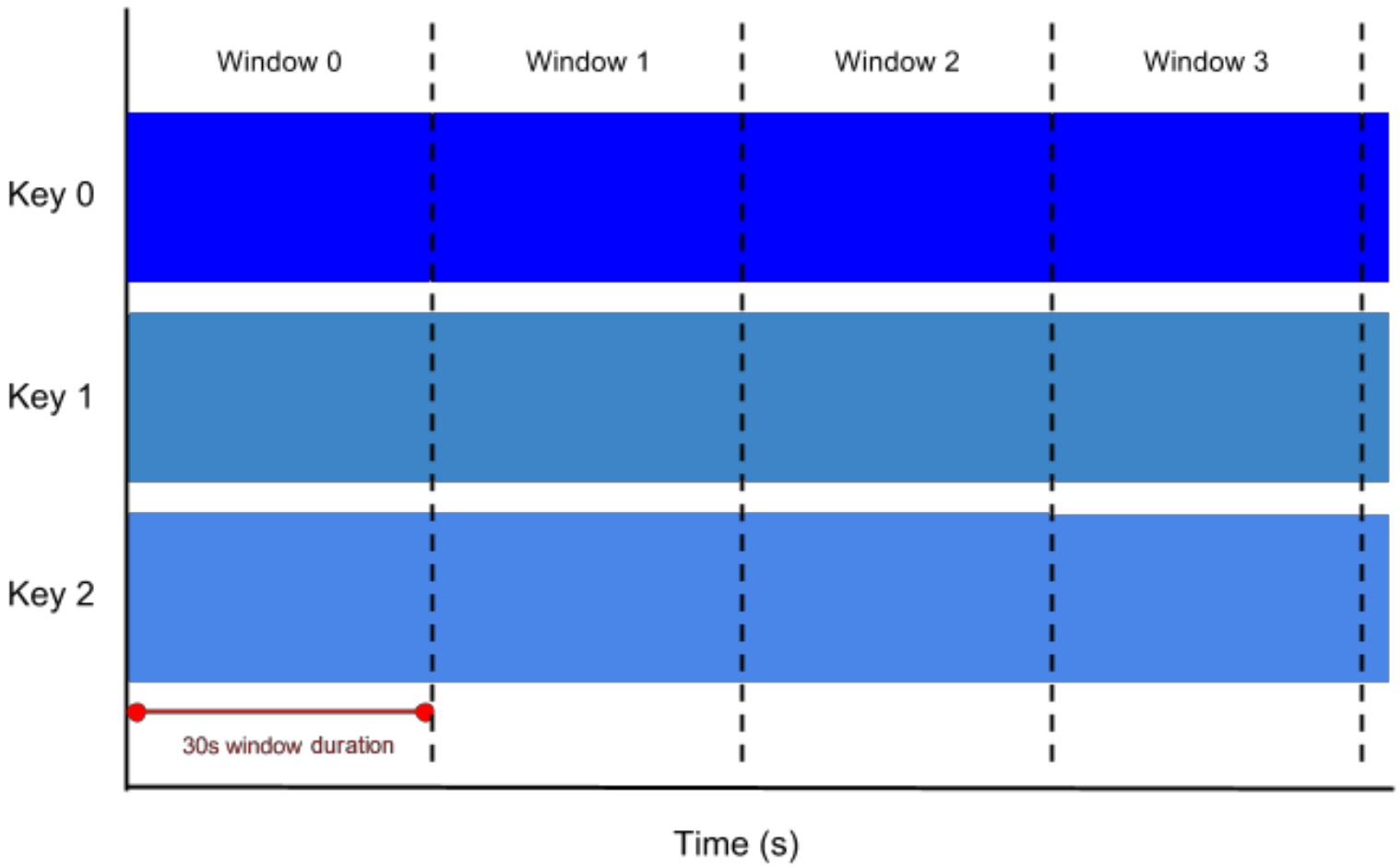
Previous

Next

② Know your window types

- **Global, Fixed, Sliding, Session** (4 types)
- **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

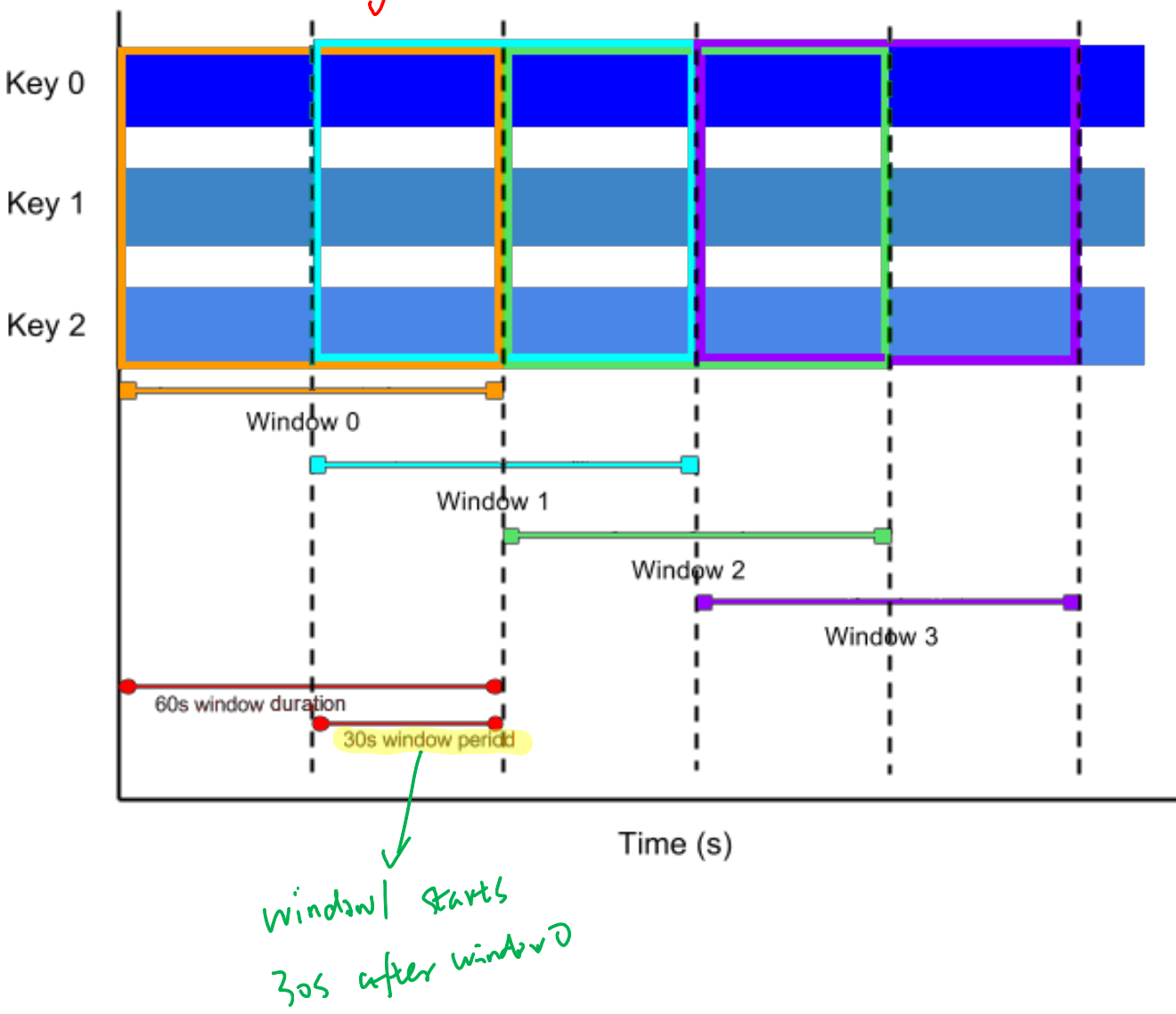
- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Additional Best Practices

Previous

Next

overlap with each other



window1 starts 30s after window0

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

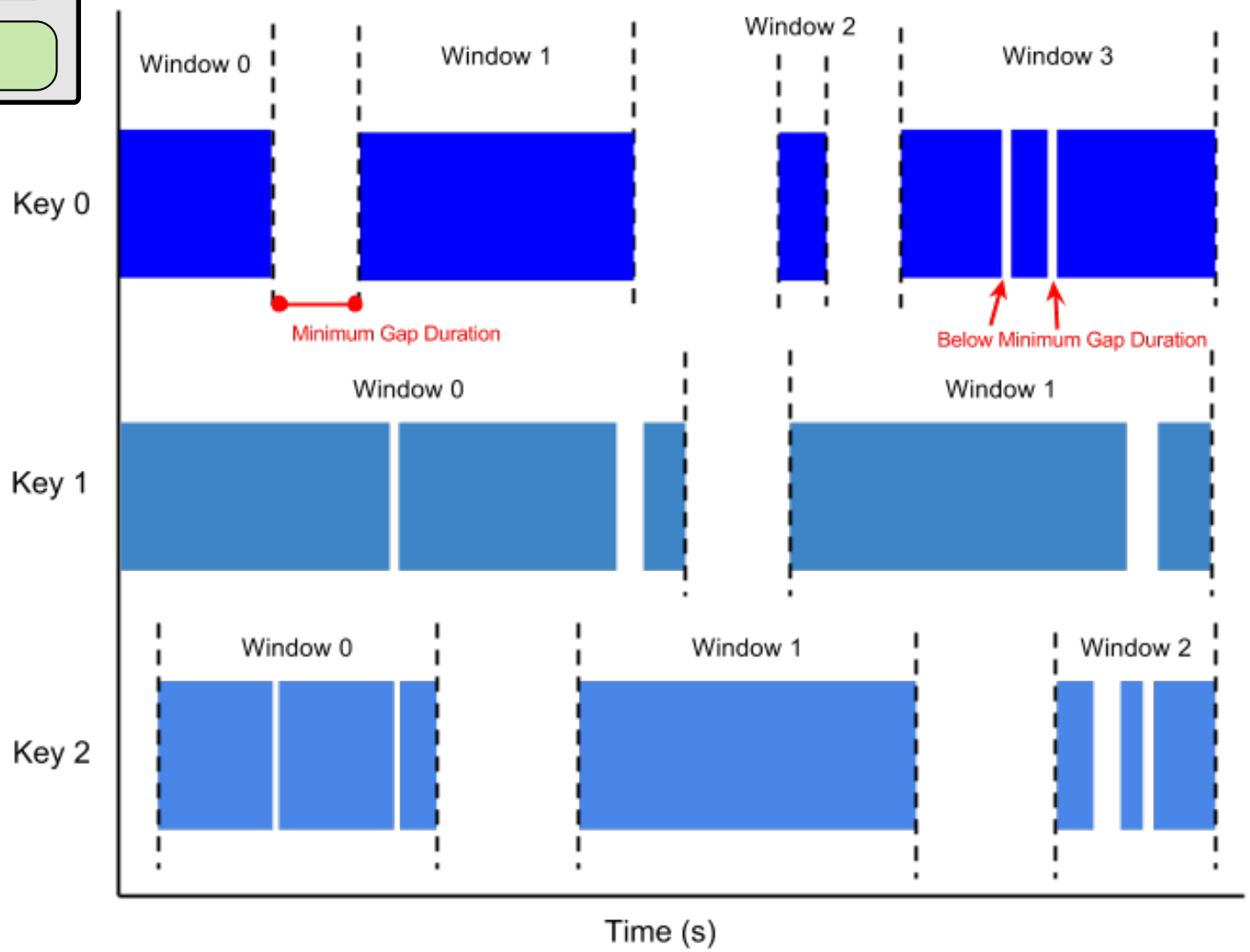
Additional Best Practices

Previous

Next

e-commerce website record "how long for each activity"

Session Window (based on incoming elements)



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

from a pub/sub topic

- **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