

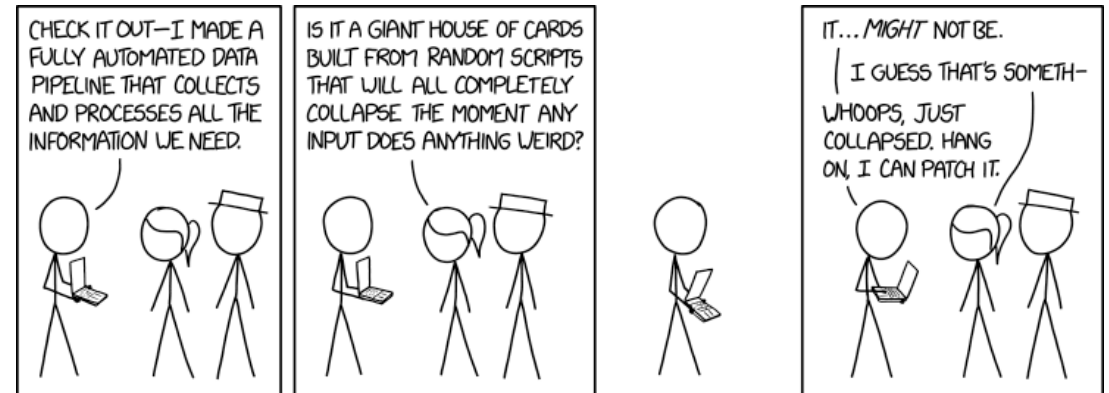
BIG DATA AND CLOUD PLATFORMS

Data pipelines on cloud (Storage)

Data pipeline

Data pipeline

"A sequence of operations to transform and consume raw data"



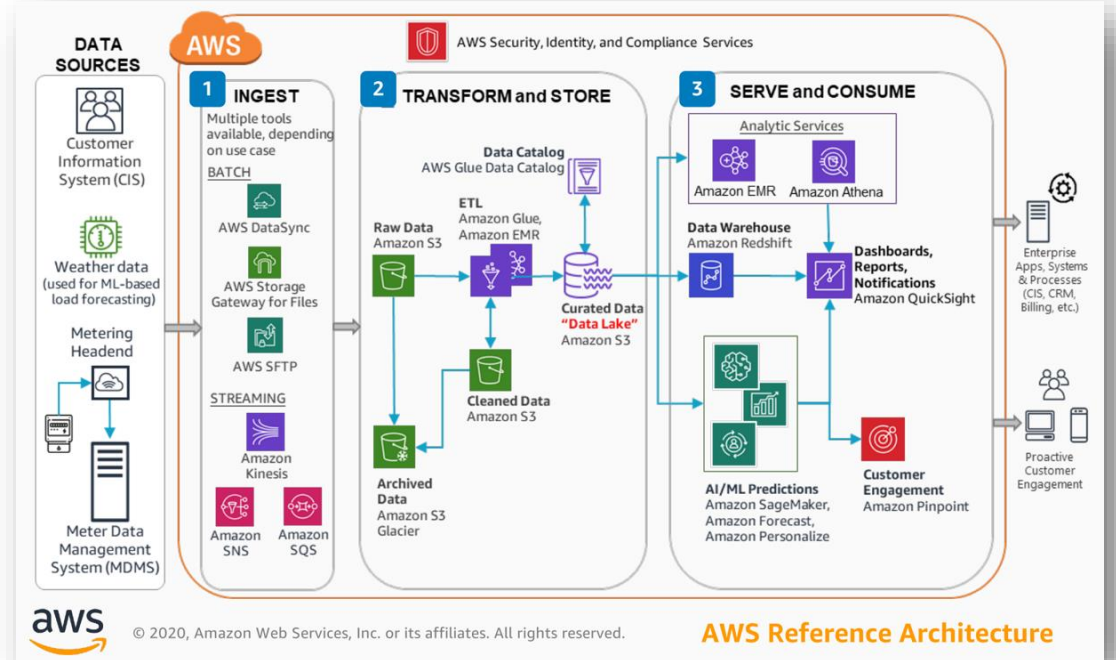
<https://xkcd.com/2054/>

Quemy, Alexandre. "Data Pipeline Selection and Optimization." *DOLAP*. 2019.

Data pipeline - AWS

Three main categories

- Ingest
 - Gateway, DataSync (batch)
 - Kinesis, SNS, SQS (stream)
- Transform and store
 - S3 and Glacier (storage)
 - Glue (ETL)
- Serve and consume
 - EMR (Hadoop-like cluster)
 - Athena (serverless query service to analyze data in Amazon S3)
 - (Many) Machine learning services

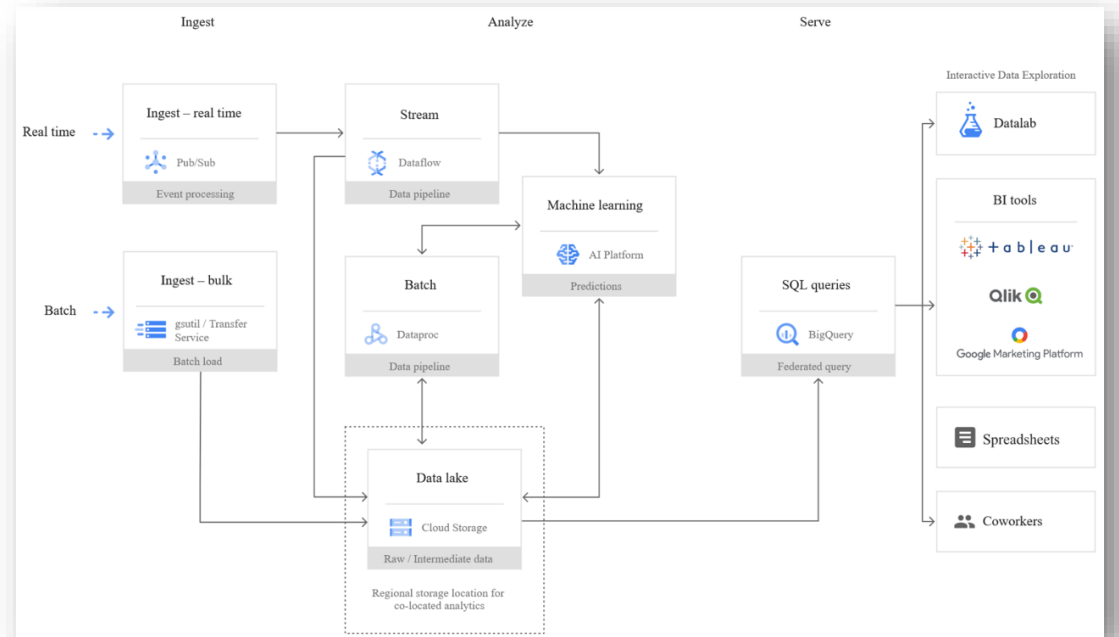


<https://console.aws.amazon.com/console>

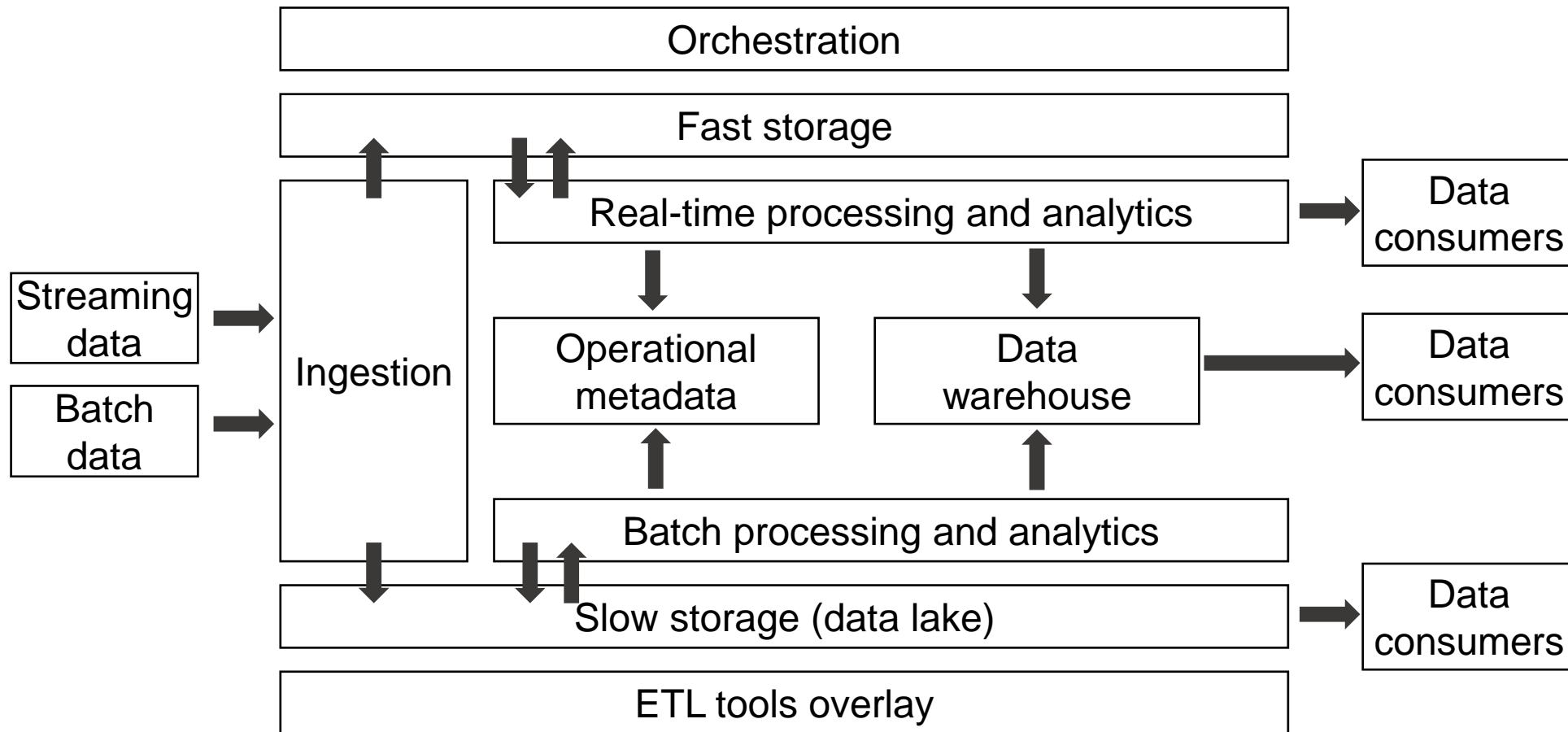
Data pipeline - Google cloud

Three main categories

- Ingest
 - Transfer service (batch)
 - Pub/Sub (stream)
- Analyze
 - Dataproc (batch)
 - Dataflow (stream)
 - Cloud storage (storage)
 - Machine learning services
- Serve
 - BigQuery (query service)



A tentative organization



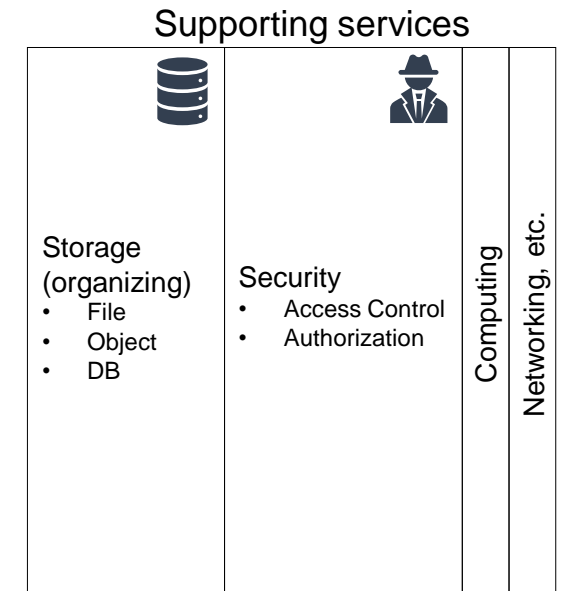
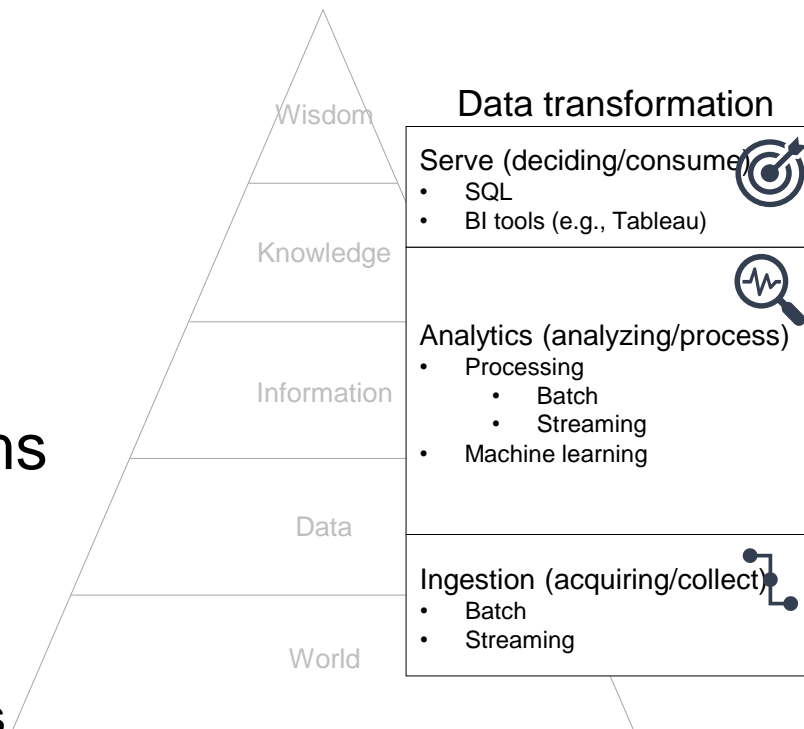
A tentative organization

We have services

- To transform data
- To support the transformation

The (DIKW) pyramid abstracts many techniques and algorithms

- Standardization
- Integration
- Orchestration
- Accessibility through APIs

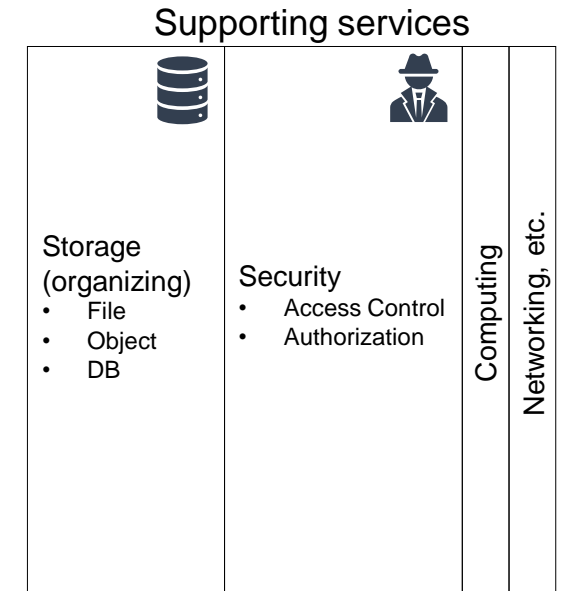
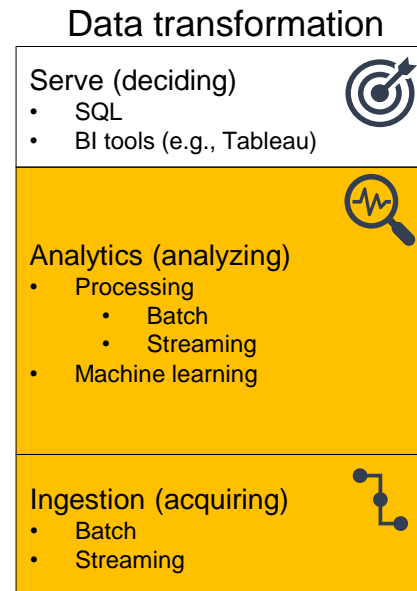


A tentative organization

This is not a sharp taxonomy

Ingestion vs Analytics

- Data streams are used for ingestion
- ... and (event) processing

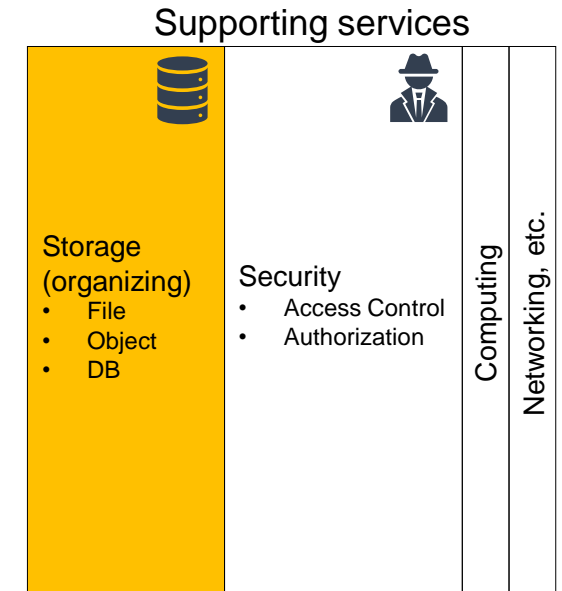
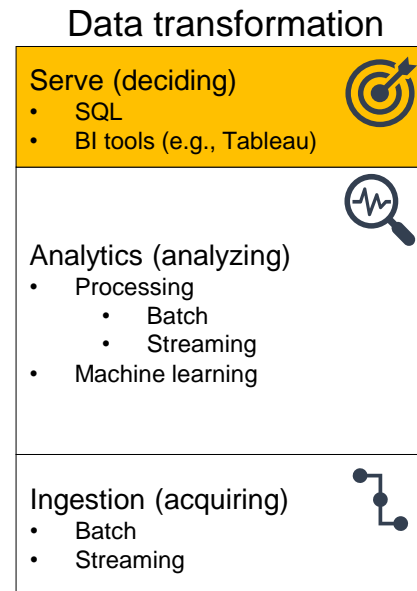


A tentative organization

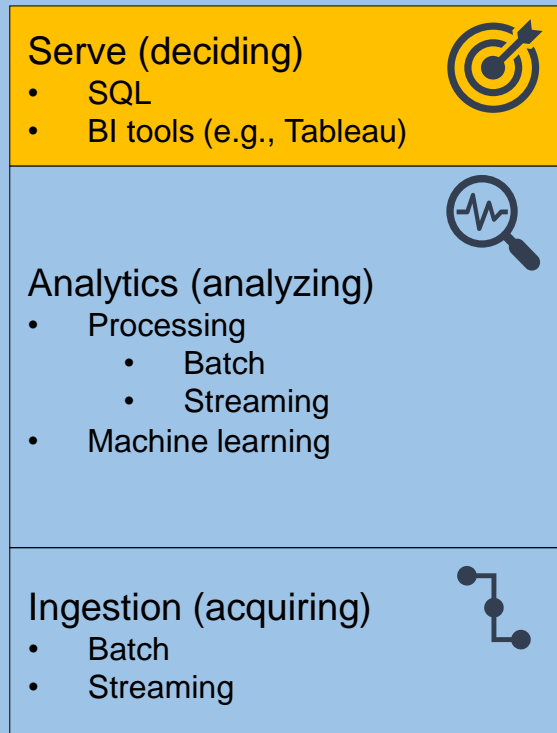
This is not a sharp taxonomy

Storage vs Serving

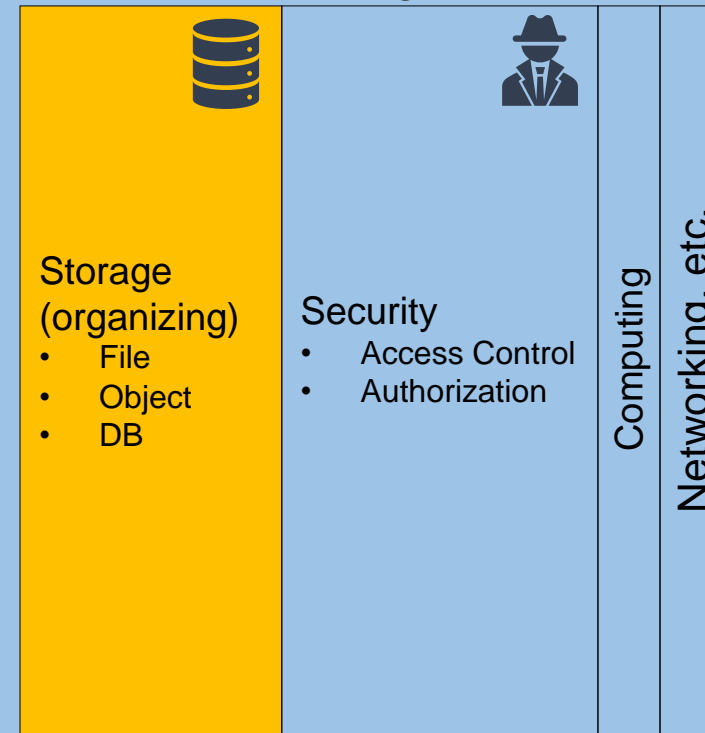
- Databases are storage
- ... with processing capability
- ... and with serving capability



Data transformation



Supporting services



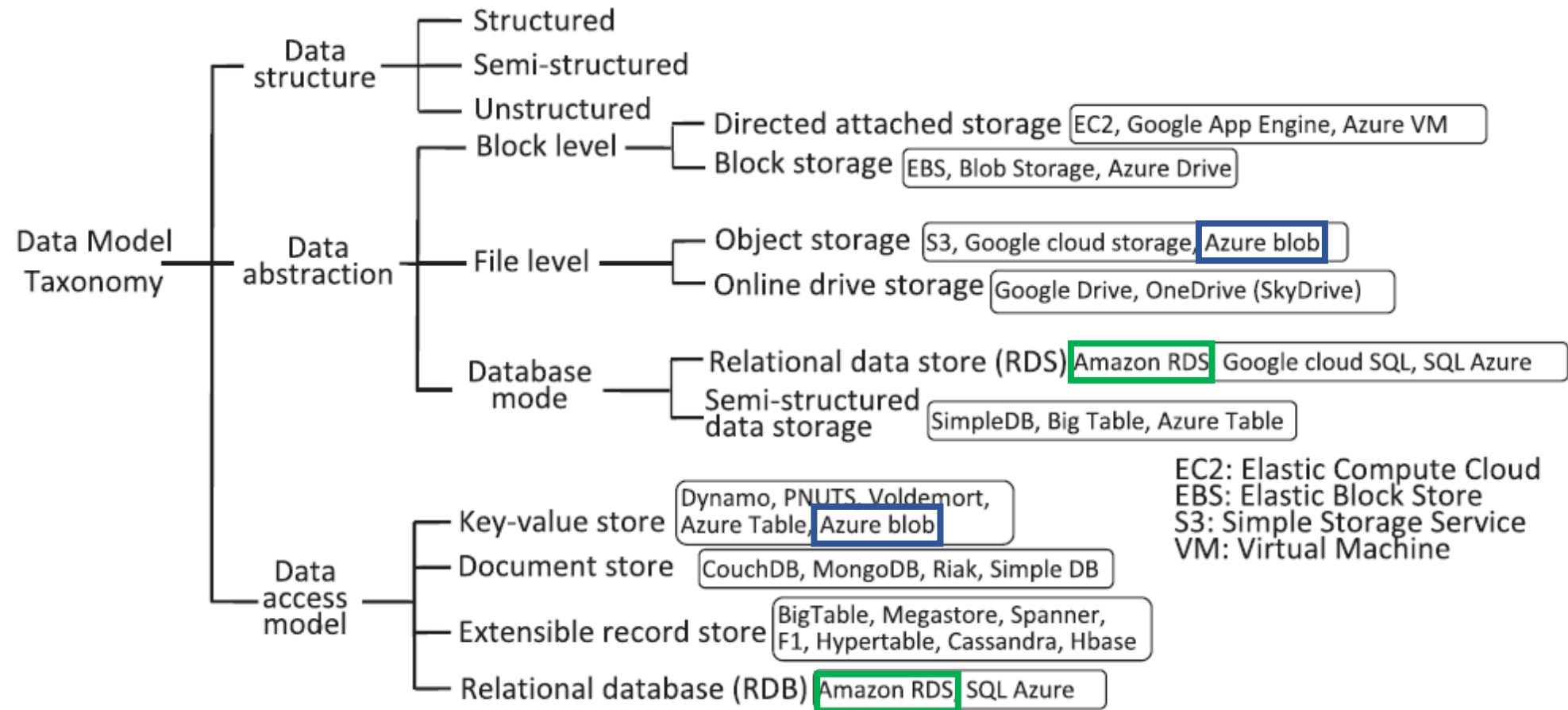
Storage

Goal: persisting data

Which storage do we choose?

- **Storage model** (or data model) ~= variety
 - How data are organized/accessed in a storage system
 - Structured vs unstructured
 - Data access model (key-value, column, etc.)
- Access **frequency**
- **Analyses** to be performed

Storage models



Mansouri, Yaser, Adel Nadjaran Toosi, and Rajkumar Buyya. "Data storage management in cloud environments: Taxonomy, survey, and future directions." ACM Computing Surveys (CSUR) 50.6 (2017): 1-51.

Storage models (AWS)












Data structure: structured

Data abstraction: database

Data access model: relational

Relational

- Store data with predefined schemas and relationships between them
- Support ACID transactions
- Maintain referential integrity

Database type	Use cases	AWS service
Relational	Traditional applications, ERP, CRM, e-commerce	 Amazon Aurora  Amazon RDS  Amazon Redshift
Key-value	High-traffic web apps, e-commerce systems, gaming applications	 Amazon DynamoDB
In-memory	Caching, session management, gaming leaderboards, geospatial applications	 Amazon ElastiCache for Memcached  Amazon ElastiCache for Redis
Document	Content management, catalogs, user profiles	 Amazon DocumentDB (with MongoDB compatibility)
Wide column	High scale industrial apps for equipment maintenance, fleet management, and route optimization	 Amazon Keyspaces (for Apache Cassandra)
Graph	Fraud detection, social networking, recommendation engines	 Amazon Neptune
Time series	IoT applications, DevOps, industrial telemetry	 Amazon Timestream
Ledger	Systems of record, supply chain, registrations, banking transactions	 Amazon QLDB

Storage models (AWS)












Data structure: semi/unstructured

Data abstraction: database

Data access model: *

- **Key/value:** store and retrieve large volumes of data
- **Document :** store semi-structured data as JSON-like documents
- **Columnar:** use tables but unlike a relational database, columns can vary from row to row in the same table
- **Graph:** navigate and query relationships between highly connected datasets
- **... and more**

<https://aws.amazon.com/products/databases/>

Database type	Use cases	AWS service
Relational	Traditional applications, ERP, CRM, e-commerce	 Amazon Aurora  Amazon RDS  Amazon Redshift
Key-value	High-traffic web apps, e-commerce systems, gaming applications	 Amazon DynamoDB
In-memory	Caching, session management, gaming leaderboards, geospatial applications	 Amazon ElastiCache for Memcached  Amazon ElastiCache for Redis
Document	Content management, catalogs, user profiles	 Amazon DocumentDB (with MongoDB compatibility)
Wide column	High scale industrial apps for equipment maintenance, fleet management, and route optimization	 * Amazon Keyspaces (for Apache Cassandra)
Graph	Fraud detection, social networking, recommendation engines	 Amazon Neptune
Time series	IoT applications, DevOps, industrial telemetry	 Amazon Timestream
Ledger	Systems of record, supply chain, registrations, banking transactions	 Amazon QLDB

Storage models (Google Cloud)

	Cloud Datastore	Bigtable	Cloud Storage	Cloud SQL	Cloud Spanner	BigQuery
Type	NoSQL document	NoSQL wide column	Blobstore	Relational SQL for OLTP	Relational SQL for OLTP	Relational SQL for OLAP
Transactions	Yes	Single-row	No	Yes	Yes	No
Complex queries	No	No	No	Yes	Yes	Yes
Capacity	Terabytes+	Petabytes+	Petabytes+	Terabytes	Petabytes	Petabytes+
Unit size	1 MB/entity	~10 MB/cell ~100 MB/row	5 TB/object	Determined by DB engine	10,240 MiB/row	10 MB/row

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

<https://cloud.google.com/products/databases>

Storage models (AWS)

Data structure: unstructured

Data abstraction: file (or database)

Data access model: key-value

File system (EFS), **object storage** (S3) (or **DB K-V**; e.g., DynamoDB)

- Handle unstructured data
- ... organized as files (or blob)
- ... accessed using a key-value

Differ in the supported features

- E.g., maximum item size (DynamoDB: 400KB, S3: 5TB)
- E.g., indexes, querying mechanisms, latency, etc.

AWS S3

Simple Storage Service (S3)

- Serverless storage, save data as **objects** within **buckets**
- An **object** is composed of a file and any metadata that describes that file (e.g., **object key**)
- **Buckets** are logical containers for objects
 - You can have one or more buckets in your account
 - Control access for each bucket individually
 - Choose the geographical region where Amazon S3 will store the bucket and its contents

Benefits

- Unified data architecture
 - Build a multi-tenant environment, where many users can bring their own data
 - Improve both cost and data governance over traditional solutions
- Decoupling of storage from compute and data processing
 - You can cost-effectively store all data types in their native formats
 - Then, launch transformations as you need

Storage: access frequency (AWS)

Object storage (AWS S3) classes

- **Standard:** general purpose
- **Infrequent (rapid) access**
- **One Zone-IA:** lower-cost option for infrequently accessed data that do not require high availability and resilience
- **Glacier:** low-cost storage class for data archiving, three retrieval options that range from a few minutes to hours
- **Deep Glacier:** long-term retention for data accessed once or twice in a year. E.g., retain data sets for 10 years or longer
- **Intelligent-Tiering:** move objects between access tiers when access patterns change

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)	99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

Storage: access frequency (AWS)

Lifecycle configuration

- A set of rules that define actions that Amazon S3 applies to a group of objects

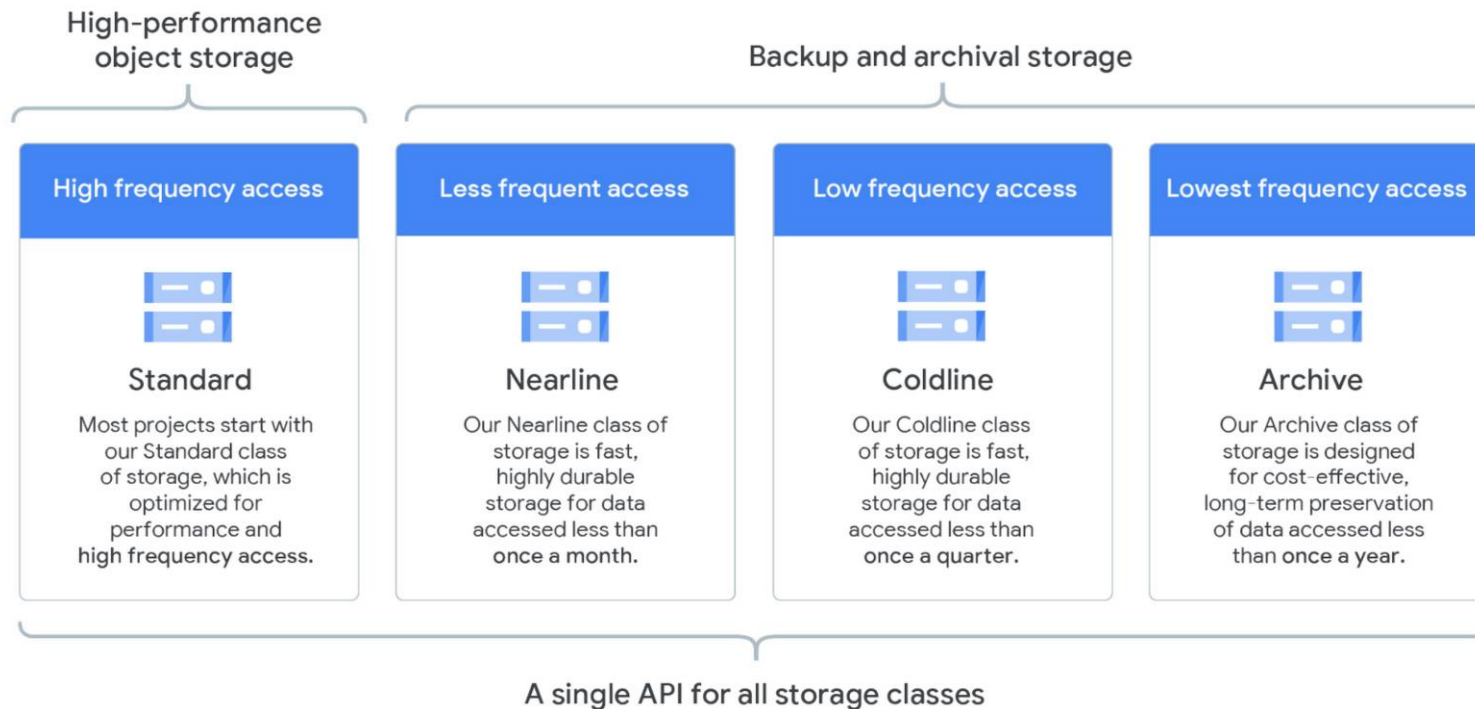
Two types of actions:

- **Transition:** when objects transition to another storage class. E.g., archive objects to the S3 Glacier storage class one year after creating them
- **Expiration:** when objects expire. Amazon S3 deletes expired objects on your behalf

	S3 Standard	S3 Intelligent-Tiering*	S3 Standard-IA	S3 One Zone-IA†	S3 Glacier	S3 Glacier Deep Archive
Designed for durability	99.999999999% (11 9's)	Transition →				99.999999999% (11 9's)
Designed for availability	99.99%	99.9%	99.9%	99.5%	99.99%	99.99%
Availability SLA	99.9%	99%	99%	99%	99.9%	99.9%
Availability Zones	≥3	≥3	≥3	1	≥3	≥3
Minimum capacity charge per object	N/A	N/A	128KB	128KB	40KB	40KB
Minimum storage duration charge	N/A	30 days	30 days	30 days	90 days	180 days
Retrieval fee	N/A	N/A	per GB retrieved	per GB retrieved	per GB retrieved	per GB retrieved
First byte latency	milliseconds	milliseconds	milliseconds	milliseconds	select minutes or hours	select hours
Storage type	Object	Object	Object	Object	Object	Object
Lifecycle transitions	Yes	Yes	Yes	Yes	Yes	Yes

<https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-mgmt.html>

Storage: access frequency (Google Cloud)



Access Frequency

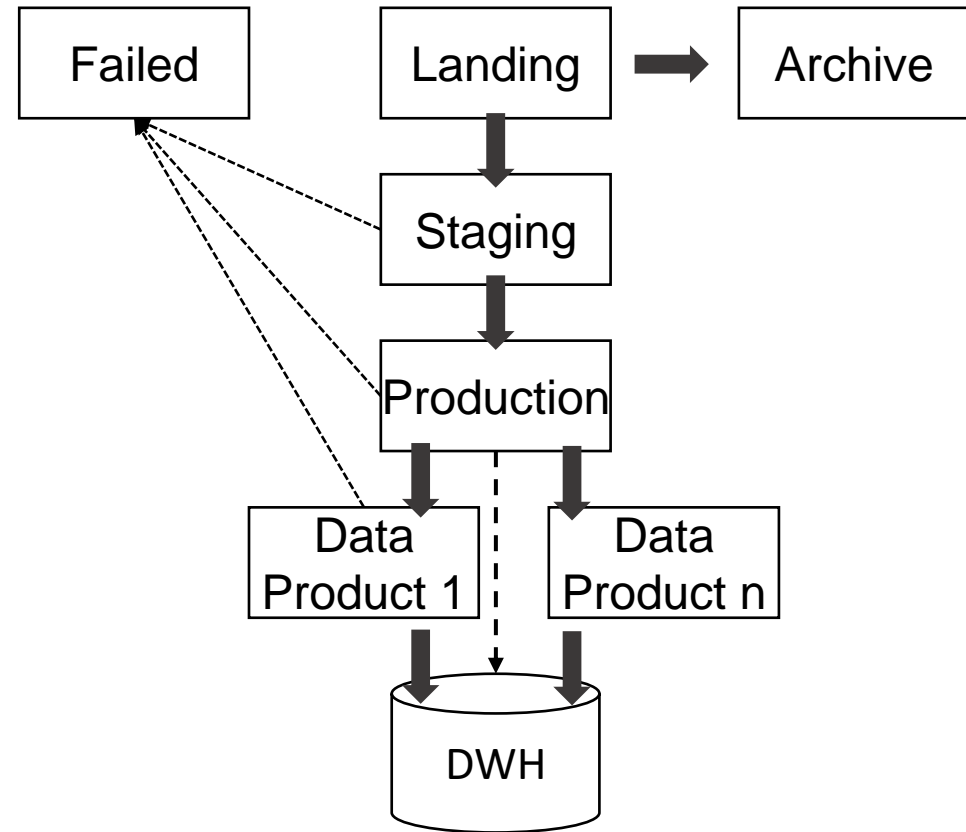
	Retention Period			
	<1 mo	1–3 mo	3–12 mo	>12 mo
>12/yr	Standard	Standard	Standard	Standard
4–12/yr	Standard	Nearline	Nearline	Nearline
1–4/yr	Standard	Nearline	Coldline	Coldline
<1/yr	Standard	Nearline	Coldline	Archive

<https://cloud.google.com/blog/products/storage-data-transfer/archive-storage-class-for-coldest-data-now-available>

Organizing the data lake

Having a consistent principles on how to organize your data is important

- To build standardized pipelines with the same design with regard to where read/write data
- Standardization makes it easier to manage your pipelines at scale
- Helps data users search for data in the storage and understand exactly to find what they need
- Decoupling storage from processing



Organizing the data lake

Landing area (LA)

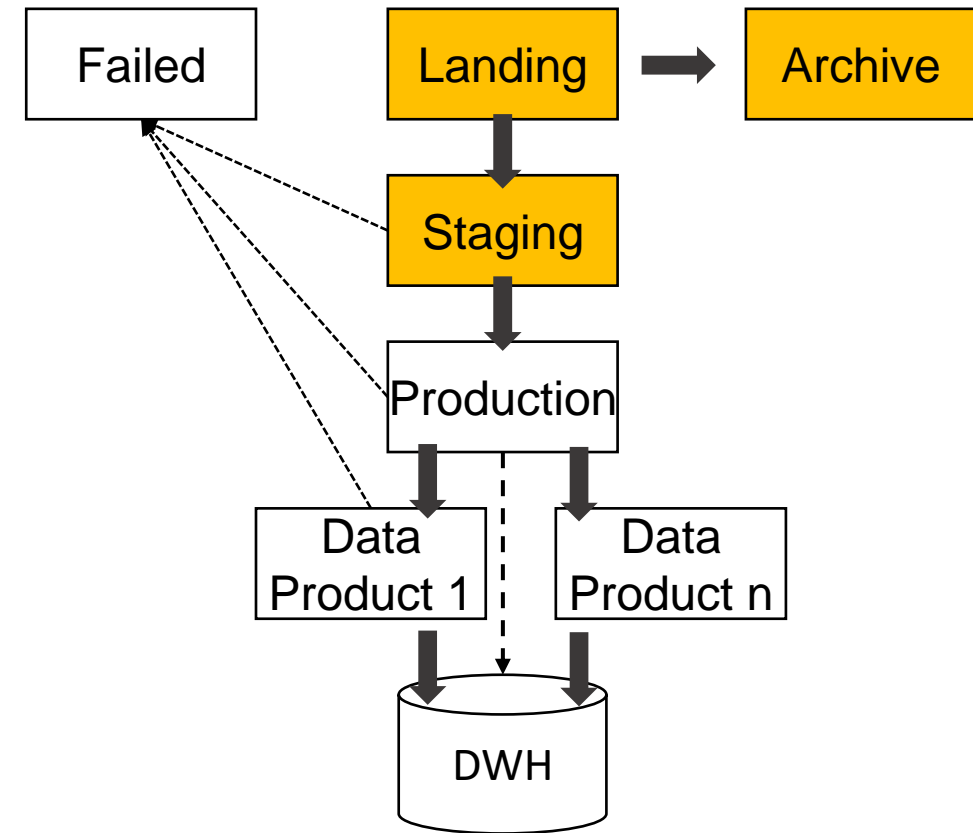
- Save **raw data** from ingestion
- Transient, data is not stored for long term

Staging area (SA)

- Raw data goes through a set of common transformations: ensuring **basic quality** and making sure it **conforms to existing schemas** for this data source and then data is saved into SA

Archive area (A)

- After saving into SA, raw data from LA should be **copied into the archive** to reprocess any given batch of data by simply copying it from AA into LA
- Useful for debugging and testing



Organizing the data lake

Production area (PA)

- Apply the business logic to data from SA

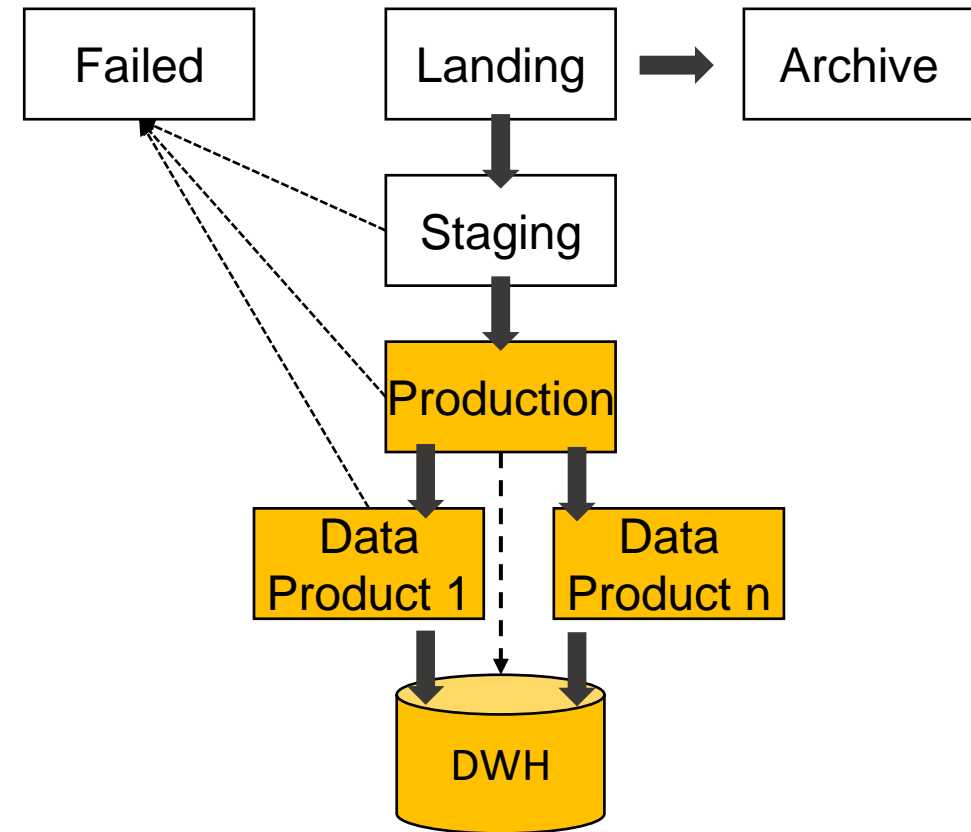
Pass-through job

- Copy data from SA to PA and then into DWH without applying any business logic
- Optional, but having a data set in the data warehouse and PA that is an exact replica can be helpful when debugging any issues with the business logic

Cloud data warehouse (DWH)

Failed area (FA)

- You need to be able to deal with all kinds of errors and failures
- There might be bugs in the pipeline code, cloud resources may fail



Organizing the data lake

Area	Permissions	Tier
Landing	Ingestion applications can write Scheduled pipelines can read Data consumers can't access	Hot
Staging	Scheduled pipelines can read/write Selected data consumers can read	Hot
Production	Scheduled pipelines can read/write Selected data consumers can read	Hot
Archive	Scheduled pipelines can write Dedicated data reprocessing pipelines can read	Cold or archive
Failed	Scheduled pipelines can write Dedicated data reprocessing pipelines can read Data consumers don't have access	Hot

Organizing the data lake

Use folders to organize data inside areas into a logical structure

- **Namespace**
 - Logically group multiple pipelines together.
- **Pipeline name**
 - Each data pipeline should have a name that reflects its purpose. For example
 - A pipeline that takes data from the LA, applies common processing steps, and saves data into SA
 - You will also have one for archiving data into AA
- **Data source name**
 - Ingestion layer will assign a name to each data source you bring into the platform
- **BatchId**
 - Unique identifier for any batch of data that is saved into LA
 - E.g., Since only ingestion can write to LA, it is its responsibility to generate this identifier
 - A common choice for this type of an identifier is a Universally Unique Identifier (UUID)

Different areas will have slightly different folder structures

- /landing/**ETL**/**sales_oracle_ingest**/**customers**/**01DFTFX89YDFAXREPJTR94**