

# BIG DATA AND CLOUD PLATFORMS

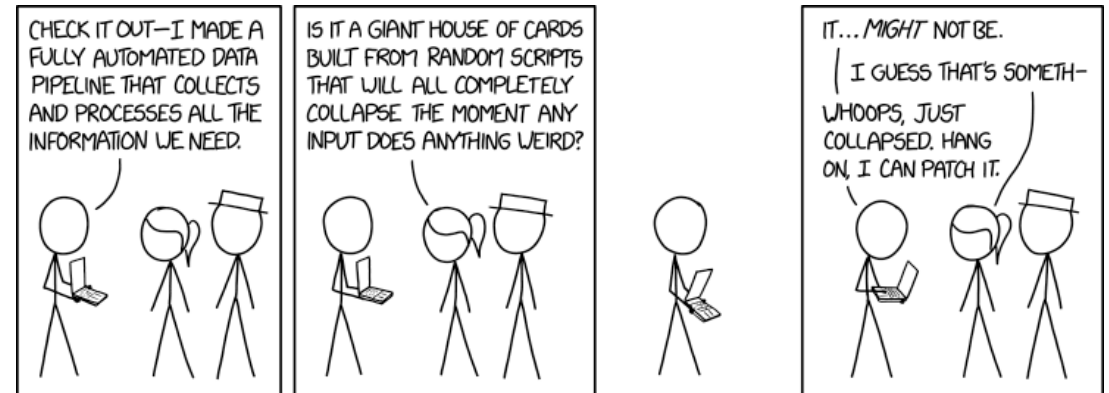
---

Building data pipelines in the cloud

# Data pipeline

## Data pipeline

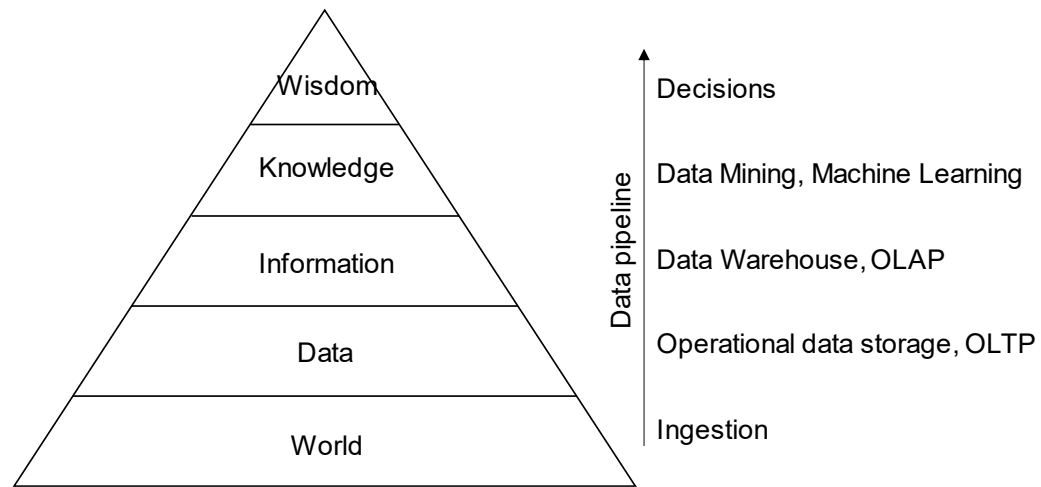
*"A sequence of operations to transform and consume raw data"*



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

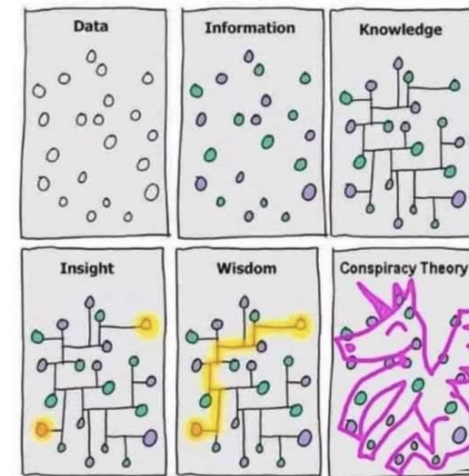
Quemy, Alexandre. "Data Pipeline Selection and Optimization." *DOLAP*. 2019.

# Data pipeline



## DIKW hierarchy

- Layers representing structural relationships between data, information, knowledge, and wisdom



Ackoff, Russell L. "From data to wisdom." Journal of applied systems analysis 16.1 (1989): 3-9.

# Data pipeline

The pyramid abstracts tons of techniques, algorithms, etc.

To provide them as services, architecting data pipelines on cloud requires

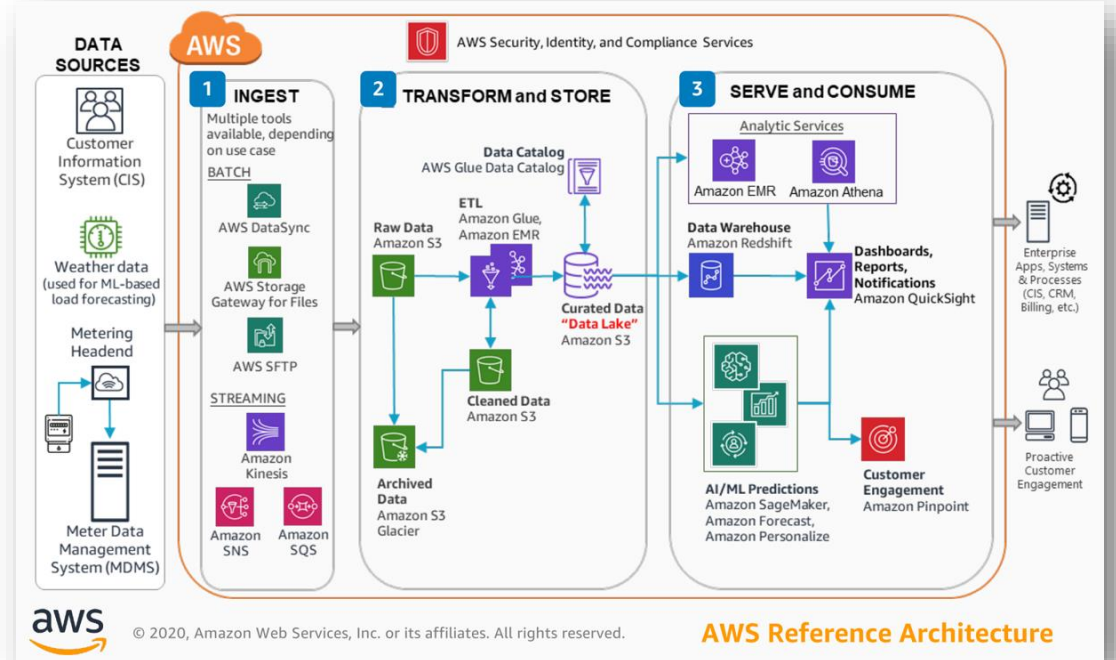
- Standardization (of common services)
- Integration
- Orchestration
- Accessibility through simple APIs

Let us look to data pipelines on different cloud services providers

# 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

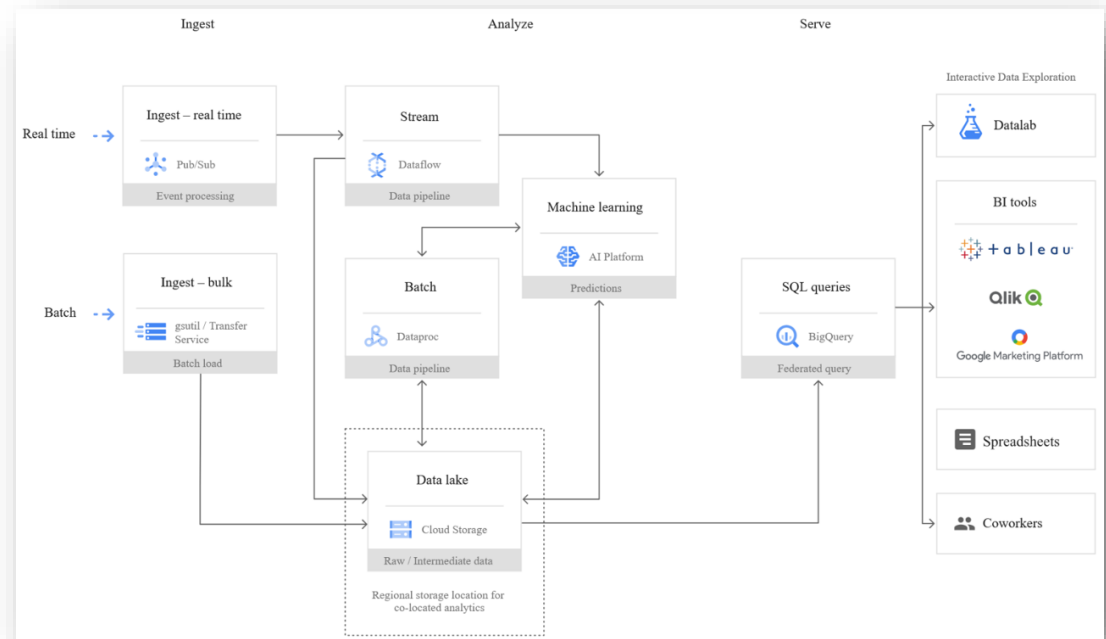


<https://console.aws.amazon.com/console>

# 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

## Common points

## We have services

- Which transforms data
- Which supports data

## Data pipelines are based on

- Ingesting data
- Analyzing data
- Serving data

Two main ways to consume data:

- Querying produces results that are great for quick analysis by data analysts.
- BI tools produce visualizations that are grouped into reports and dashboards to help users explore data

Collecting raw data from

- transactions
- logs
- IoT devices

A good solution allows developers to ingest a wide variety of data at any speed, from batch to streaming

### Data transformation

#### Serve (deciding/consume)

- SQL
- BI tools (e.g., Tableau)

#### Analytics (analyzing/process)

- Processing
  - Batch
  - Streaming
- Machine learning

#### Ingestion (acquiring/collect)

- Batch
- Streaming

Transforming data to make it consumable.

Sorting, aggregating, joining, and applying business logic to produce meaningful analytical datasets.

Load these datasets into a new storage location, like a data lake, database, or data warehouse.

### services

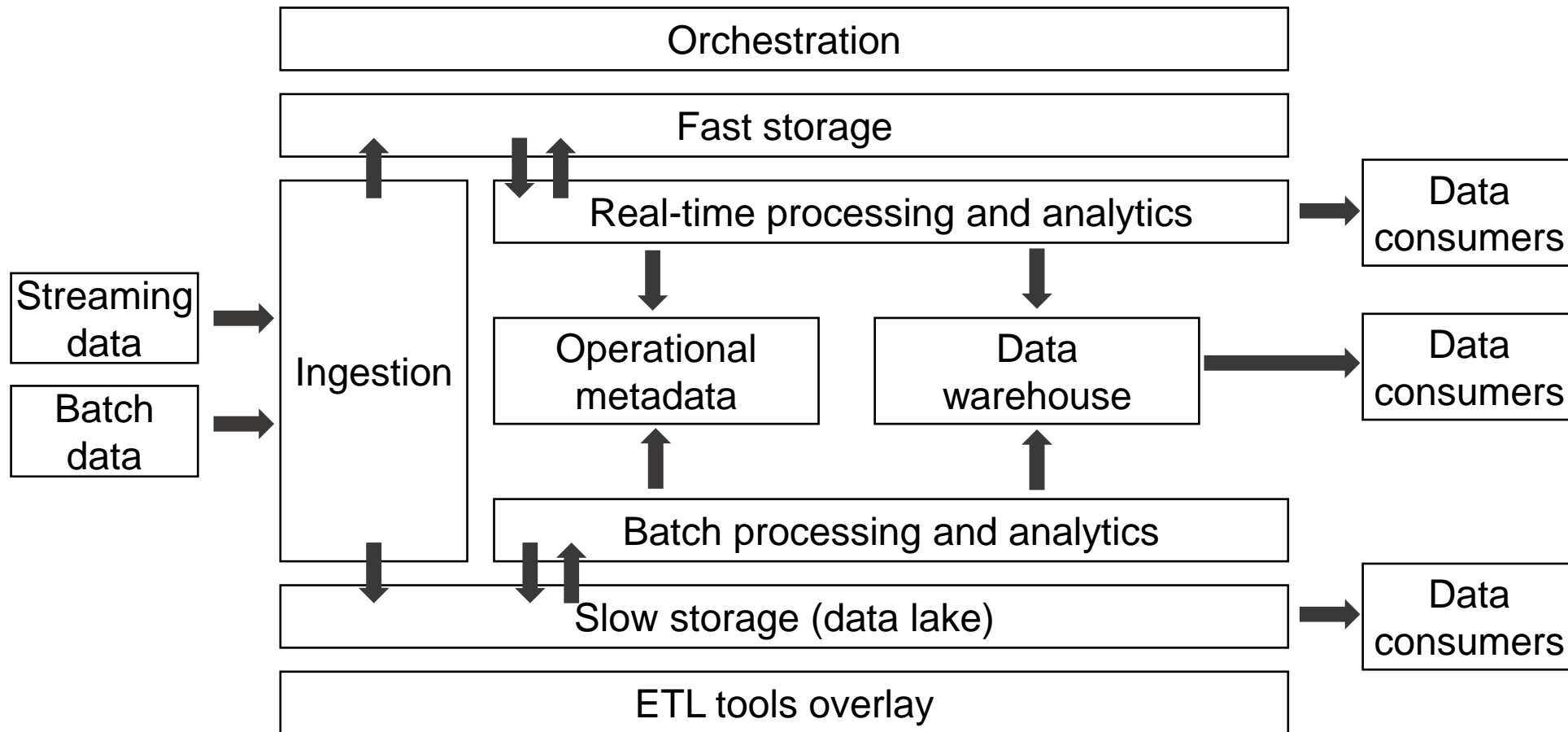


Access Control  
Authorization

Computing

Networking, etc.

# A tentative organization



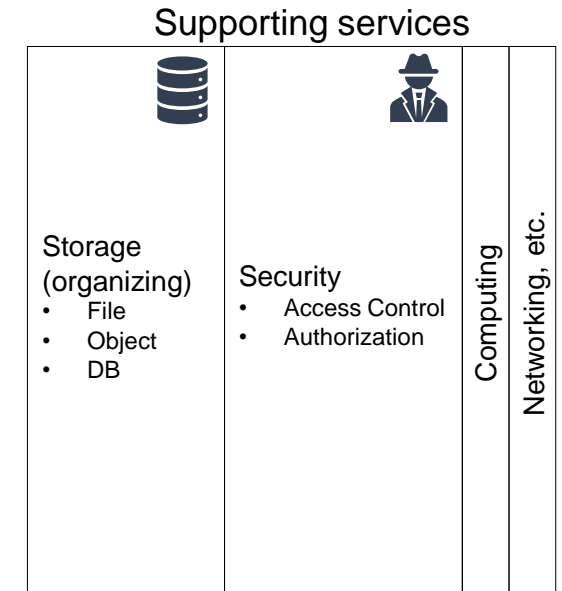
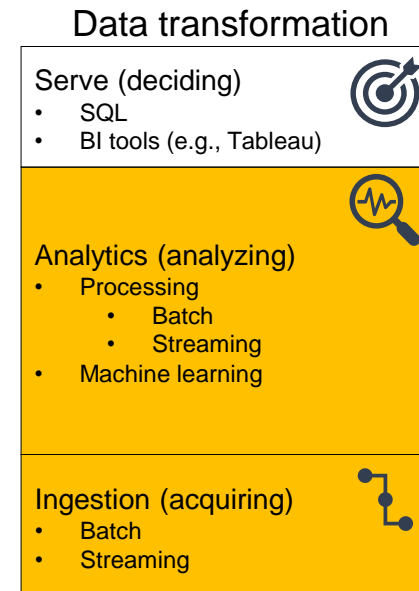


# A tentative organization

This is not a sharp taxonomy

## Ingestion vs Analytics

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

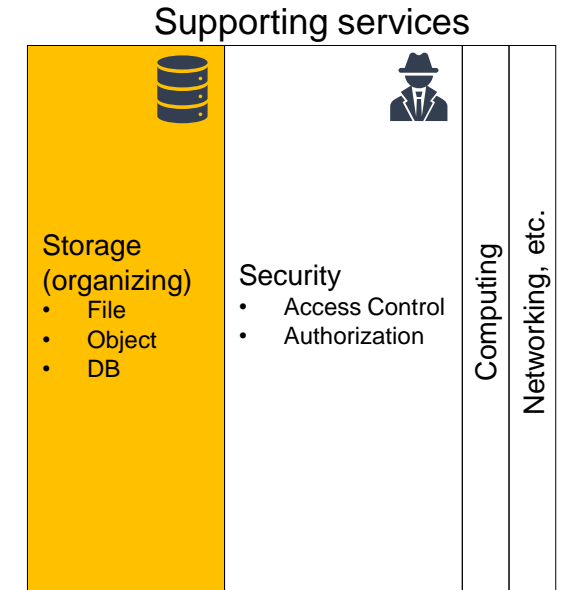
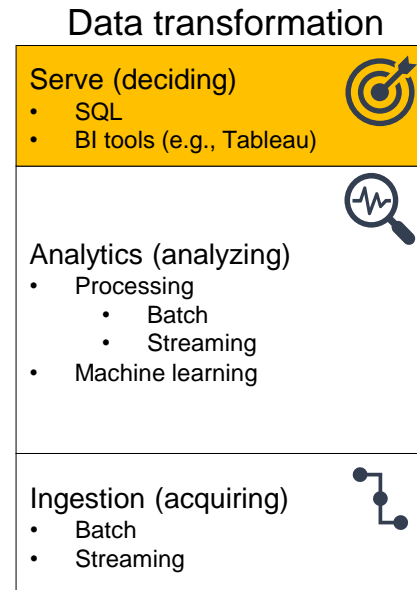


# A tentative organization

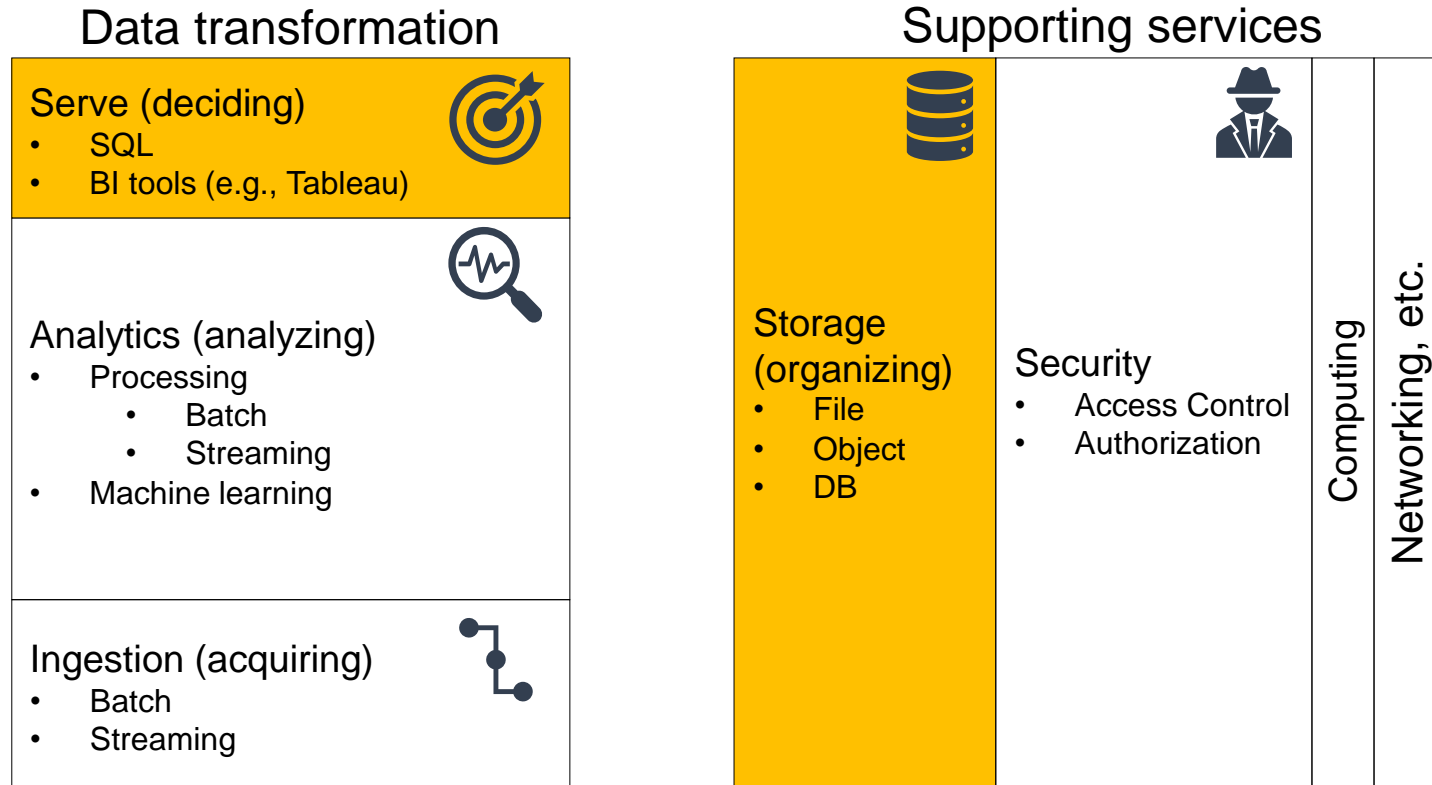
This is not a sharp taxonomy

## Storage vs Serving

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



# A tentative organization



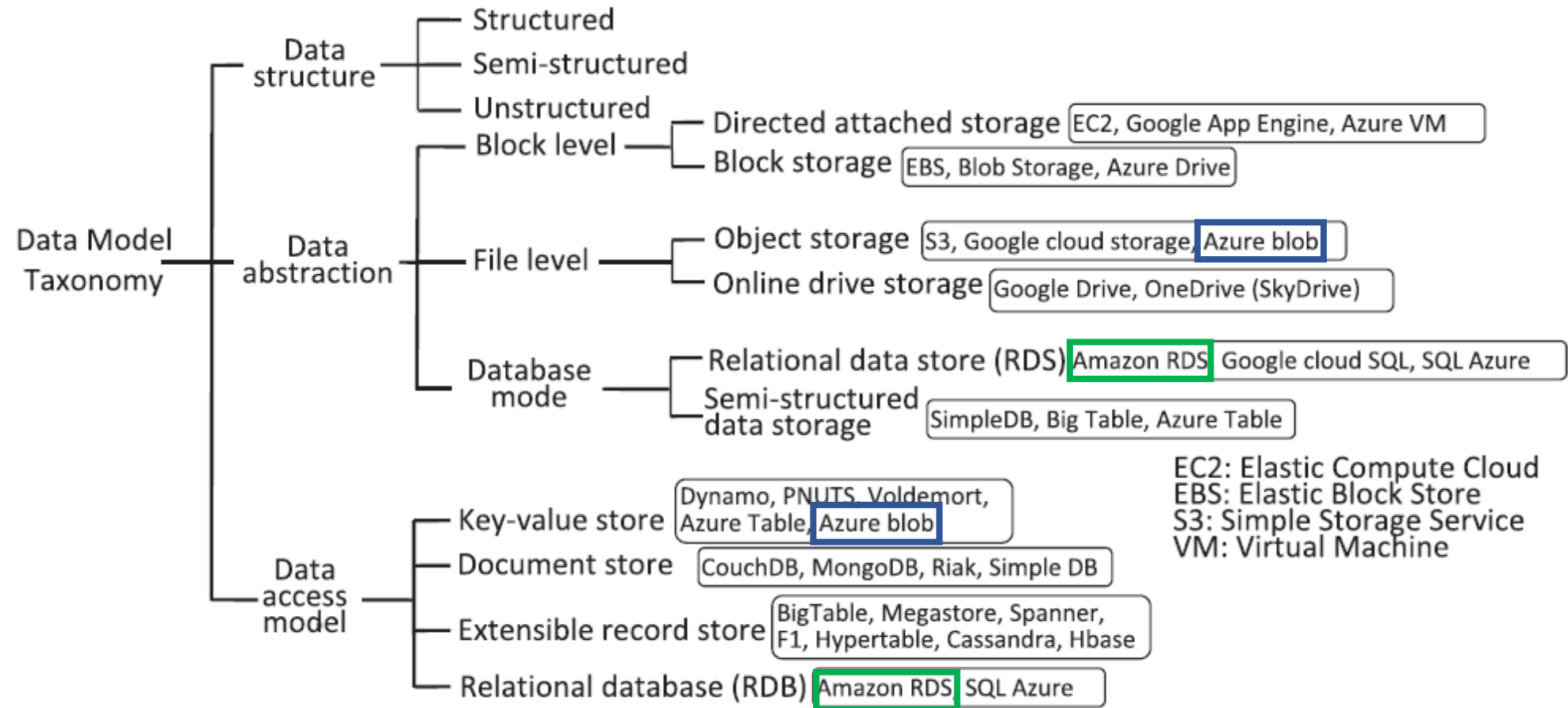
# Storage

**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
  - Once stored, objects are given an **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

- Centralized data architecture (create a unified dataset)
  - 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 as 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

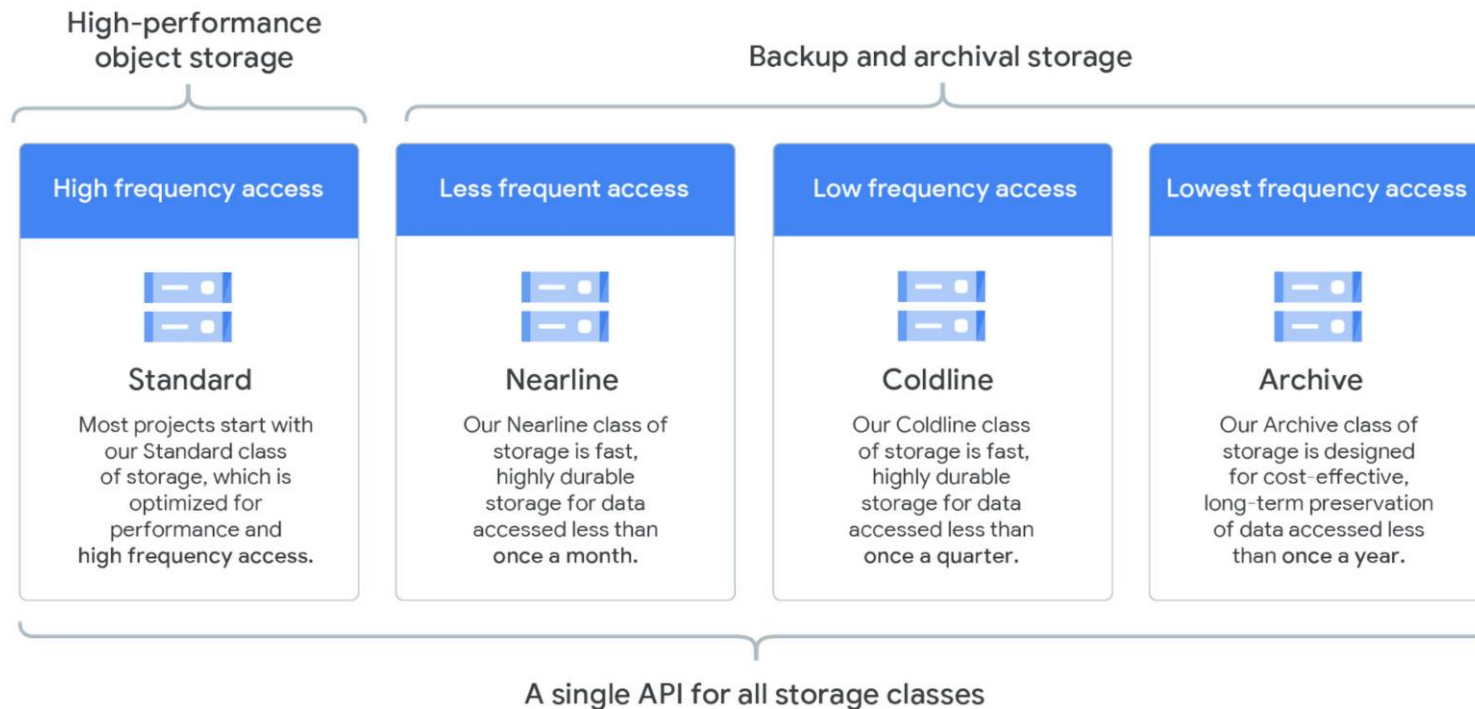
## Two types of actions:

- **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
	<b>Transition</b> →					
Designed for durability	99.999999999% (11 9's)	(11 9's)	(11 9's)	(11 9's)	(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

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

# Storage: access frequency (Google Cloud)



Access Frequency

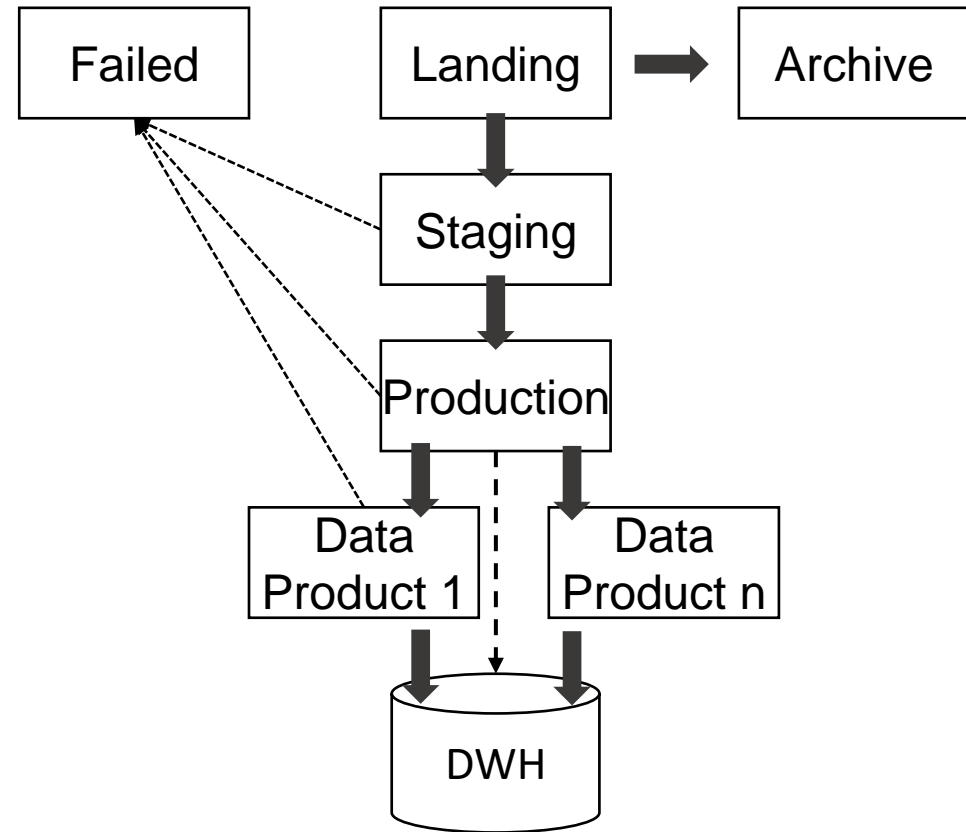
	Retention Period			
	<1 mo	1–3 mo	3–12 mo	>12 mo
>12/yr	Standard	Standard	Standard	Standard
4–12/yr	Standard	Nearline	Nearline	Nearline
1–4/yr	Standard	Nearline	Coldline	Coldline
<1/yr	Standard	Nearline	Coldline	Archive

<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)

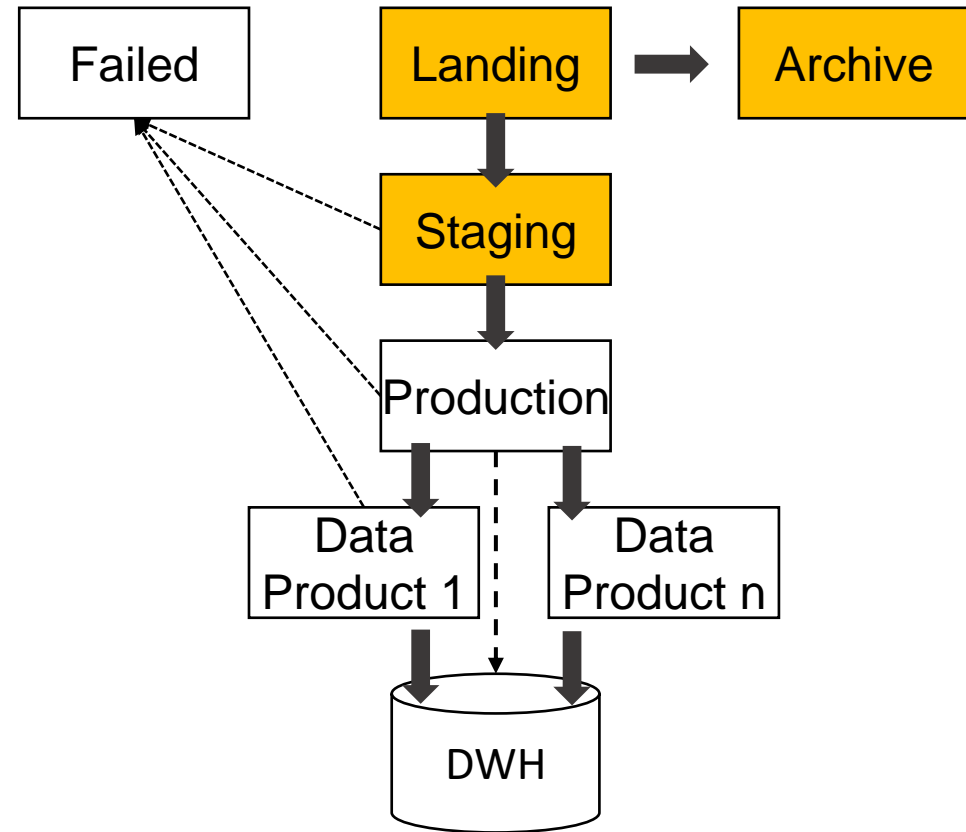
- Save **raw data** from ingestion
- Transient, data is not stored for long term

## 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

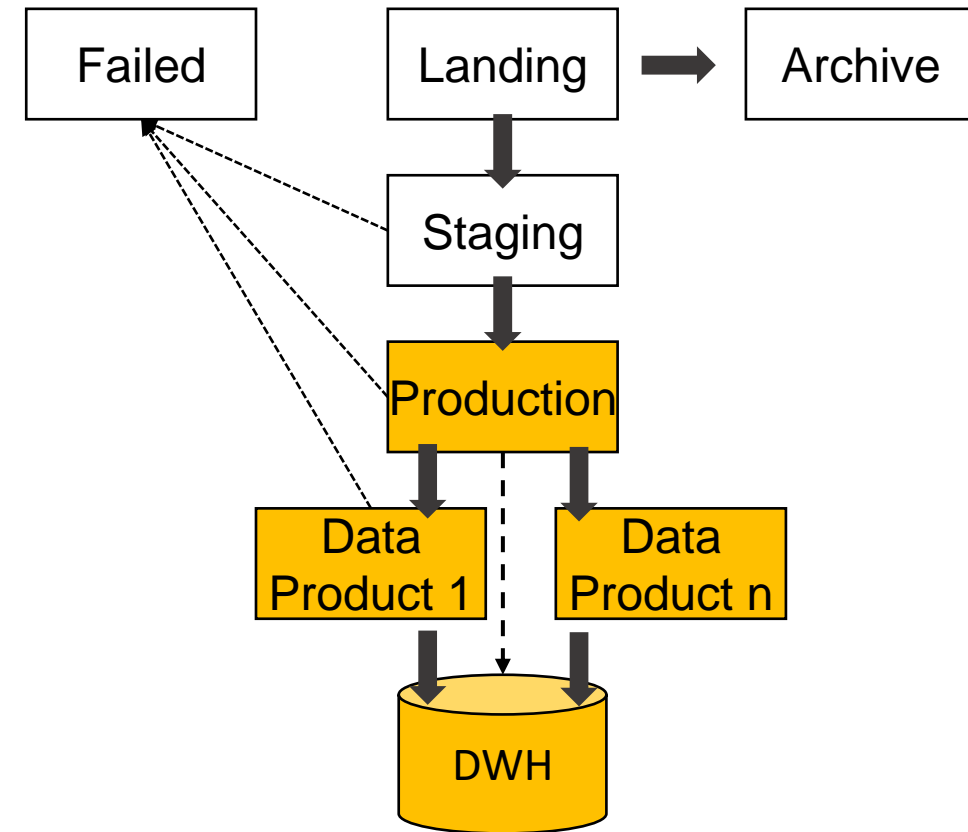
## Pass-through job

- 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

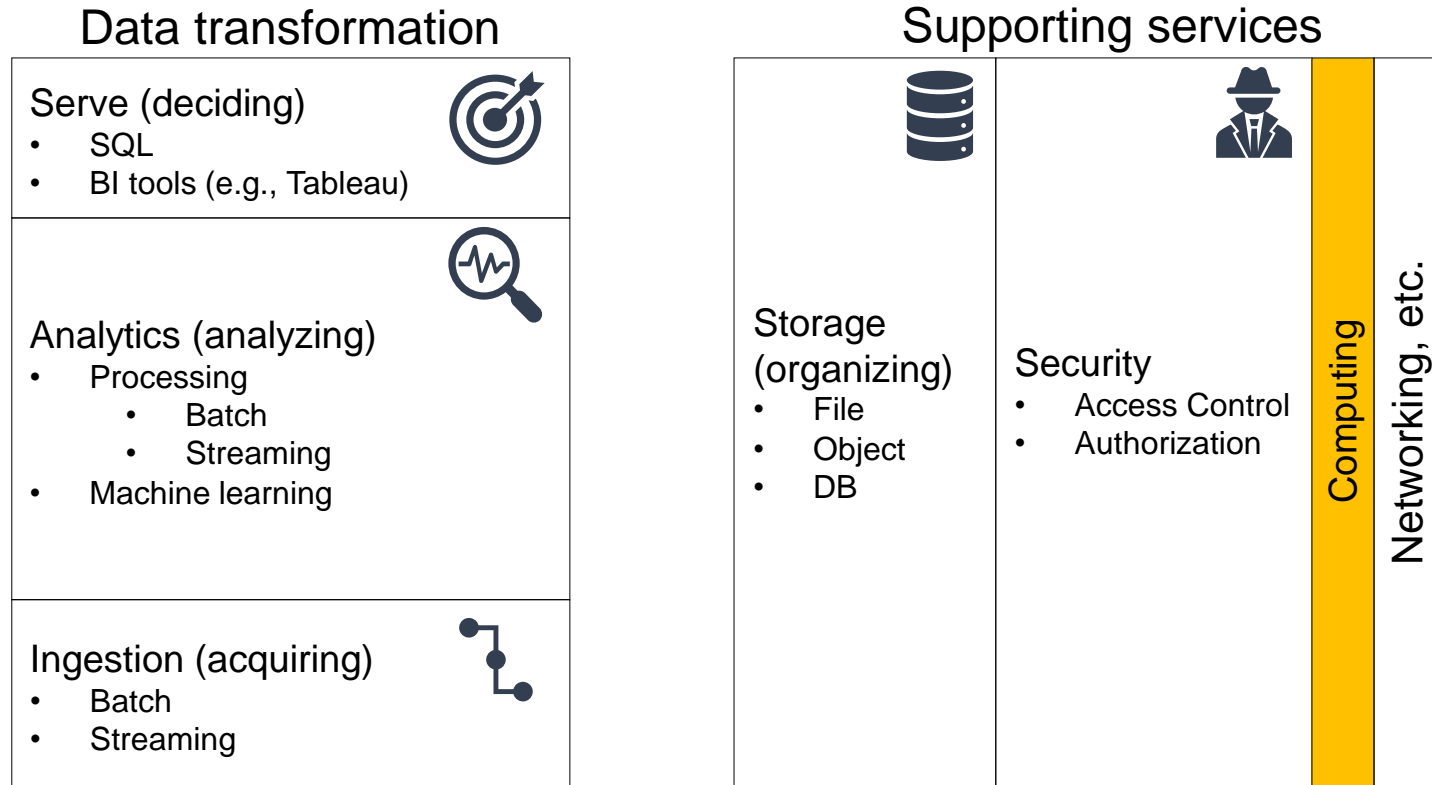
Use folders to organize data inside areas into a logical structure

- **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**

# A tentative organization



# Supporting data pipelines

We can choose the XaaS configuration to build our pipelines

- Here follows some examples

## IaaS

- Outsource virtual machines to the cloud (AWS EC2)
- (You) Manage technological and business challenges

## PaaS

- Outsource the data ecosystem to the cloud (e.g., AWS EMR)
- (You) Manage technological and business challenges



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

# Single instance: AWS EC2

## Amazon Elastic Compute Cloud

- A web service that provides resizable compute capacity
- Complete control of computing resources
  - Processor, storage, networking, OS, and purchase model

## The instance type determines the hardware

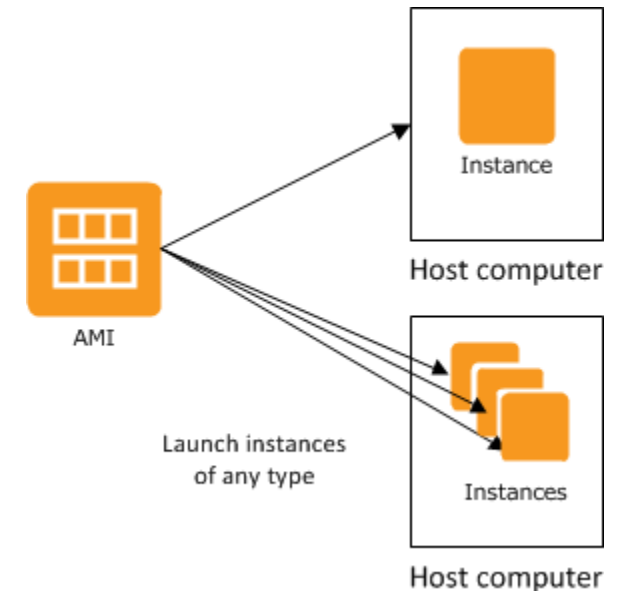
- Different compute and memory capabilities

## Amazon Machine Image is a software template

- The EC2 instance is used for creating the virtual server instance
- The AMI is the EC2 virtual machines image

## Interact with EC2 instance as with any computer

- You have complete control of your instances



<https://aws.amazon.com/ec2/>

<https://aws.amazon.com/ec2/instance-types/>

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/compute-optimized-instances.html> (example of AMI)

# Single instance: AWS EC2

General Purpose

Compute Optimized

Memory Optimized

Accelerated Computing

Storage Optimized

Instance Features

Measuring Instance Performance

MacT4gT3T3aT2M6gM5M5aM5nM5znM4A1

M4 instances provide a balance of compute, memory, and network resources, and it is a good choice for many applications.

Features:

- 2.3 GHz Intel Xeon® E5-2686 v4 (Broadwell) processors or 2.4 GHz Intel Xeon® E5-2676 v3 (Haswell) processors
- EBS-optimized by default at no additional cost
- Support for Enhanced Networking
- Balance of compute, memory, and network resources

Instance	vCPU*	Mem (GiB)	Storage	Dedicated EBS Bandwidth (Mbps)	Network Performance
m4.large	2	8	EBS-only	450	Moderate
m4.xlarge	4	16	EBS-only	750	High
m4.2xlarge	8	32	EBS-only	1,000	High
m4.4xlarge	16	64	EBS-only	2,000	High
m4.10xlarge	40	160	EBS-only	4,000	10 Gigabit
m4.16xlarge	64	256	EBS-only	10,000	25 Gigabit

All instances have the following specs:

- 2.4 GHz Intel Xeon E5-2676 v3\*\* Processor
- Intel AVX†, Intel AVX2†, Intel Turbo
- EBS Optimized
- Enhanced Networking†

General Purpose

Compute Optimized

Memory Optimized

Accelerated Computing

Storage Optimized

Instance Features

Measuring Instance Performance

C6gC6gnC5C5aC5nC4

Amazon EC2 C6g instances are powered by Arm-based AWS Graviton2 processors. They deliver up to 40% better price performance over current generation C5 instances for compute-intensive applications.

Features:

- Custom built AWS Graviton2 Processor with 64-bit Arm Neoverse cores
- Support for Enhanced Networking with Up to 25 Gbps of Network bandwidth
- EBS-optimized by default
- Powered by the AWS Nitro System, a combination of dedicated hardware and lightweight hypervisor
- With C6gd instances, local NVMe-based SSDs are physically connected to the host server and provide block-level storage that is coupled to the lifetime of the instance

Instance Size	vCPU	Memory (GiB)	Instance Storage (GiB)	Network Bandwidth (Gbps)	EBS Bandwidth (Mbps)
c6g.medium	1	2	EBS-Only	Up to 10	Up to 4,750
c6g.large	2	4	EBS-Only	Up to 10	Up to 4,750
c6g.xlarge	4	8	EBS-Only	Up to 10	Up to 4,750
c6g.2xlarge	8	16	EBS-Only	Up to 10	Up to 4,750
c6g.4xlarge	16	32	EBS-Only	Up to 10	4750
c6g.8xlarge	32	64	EBS-Only	12	9000
c6g.12xlarge	48	96	EBS-Only	20	13500
c6g.16xlarge	64	128	EBS-Only	25	19000

<https://aws.amazon.com/ec2/instance-types/>

# Single instance: AWS EC2

AWS uses public-key cryptography to secure the login

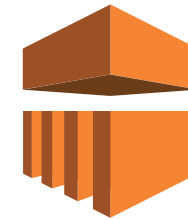
You can create one using the Amazon EC2 console

- Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>
- In the navigation pane, choose `Key Pairs`
- Choose `Create key pair`
- For `Name`, enter a descriptive name for the key pair
- For `File format`, choose the format in which to save the private key
  - OpenSSH, choose `pem` (`chmod 400 my-key-pair.pem`)
  - PuTTY, choose `ppk`
- Choose `Create key pair`
- The private key file is automatically downloaded by your browser

# Cluster: AWS EMR

Amazon EMR is a data platform based on the Hadoop stack

- Apache Spark, Apache Hive, Apache HBase, etc.
- You can run workloads on
  - Amazon EC2 instances
  - Amazon Elastic Kubernetes Service (EKS) clusters



## Example of workload

- Upload input data into Amazon S3
- EMR launches EC2 instances that you specified
- EMR begins the execution while pulling the input data from S3 into the launched instances
- Once the cluster is finished, EMR transfers output data to Amazon S3

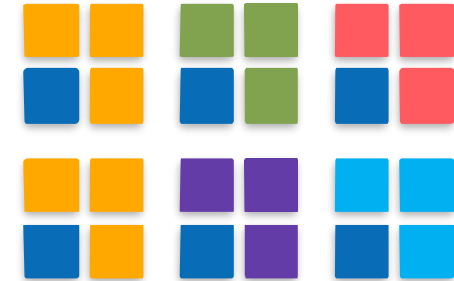


# AWS EMR

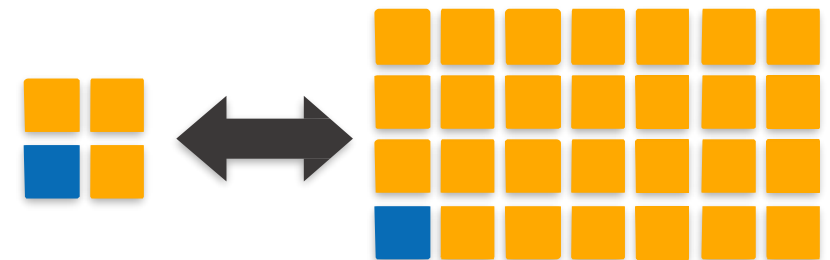
Provision as much capacity as you need

Add or remove capacity at any time

Deploy Multiple Clusters



Resize a Running Cluster



# AWS EMR

## EMR cluster

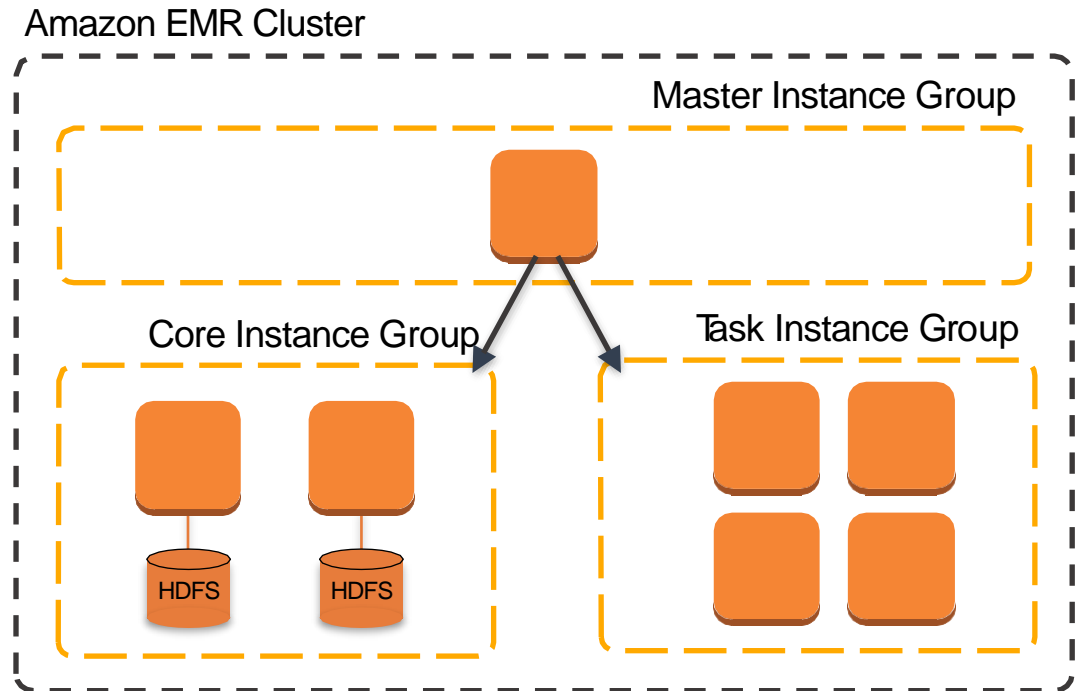
### Master group controls the cluster

- Coordinate the work distribution
- Manage the cluster state

### Core groups

- Core instances run Data Node daemons

### (Optional) Task instances



# AWS EMR

The central component of Amazon EMR is the **cluster**

- A collection of **Amazon Elastic Compute Cloud (Amazon EC2)** instances
- Each instance is called a **node**

The **node type** identifies the role within the cluster

- **Master** node coordinates the distribution of data and tasks among other nodes
  - Every cluster has (at least) a master node
  - Always active
- **Core** node runs tasks and store data in the Hadoop Distributed File System (HDFS)
  - Multi-node clusters have at least one core node
  - Always active, contains the data node daemon
- **Task** node only runs tasks
  - Task nodes are optional
  - Decoupling processing and storage, we lose data locality

# AWS EMR

## On-Demand Instance

- Pay for compute capacity by the hour (minimum of 60 seconds)
- No long-term commitments

## Spot Instance

- Unused EC2 instance that is available for less than the on-demand price
- Hourly price is called *spot price*
  - Adjusted based on long-term supply and demand for spot instances
- Run the instance when capacity is available and price is below threshold
  - When data-center resources are low, spot instances are dropped
  - Mainly suitable for batch workloads

<https://aws.amazon.com/ec2/pricing/>

# AWS EMR

## Spot Instance cost strategies

### Capacity-optimized strategy

- Allocated instances into the most available pools
- Look at real-time capacity data, predict which are the most available
- Works well for workloads such as big data and analytics
- Works well when we have high cost of interruption

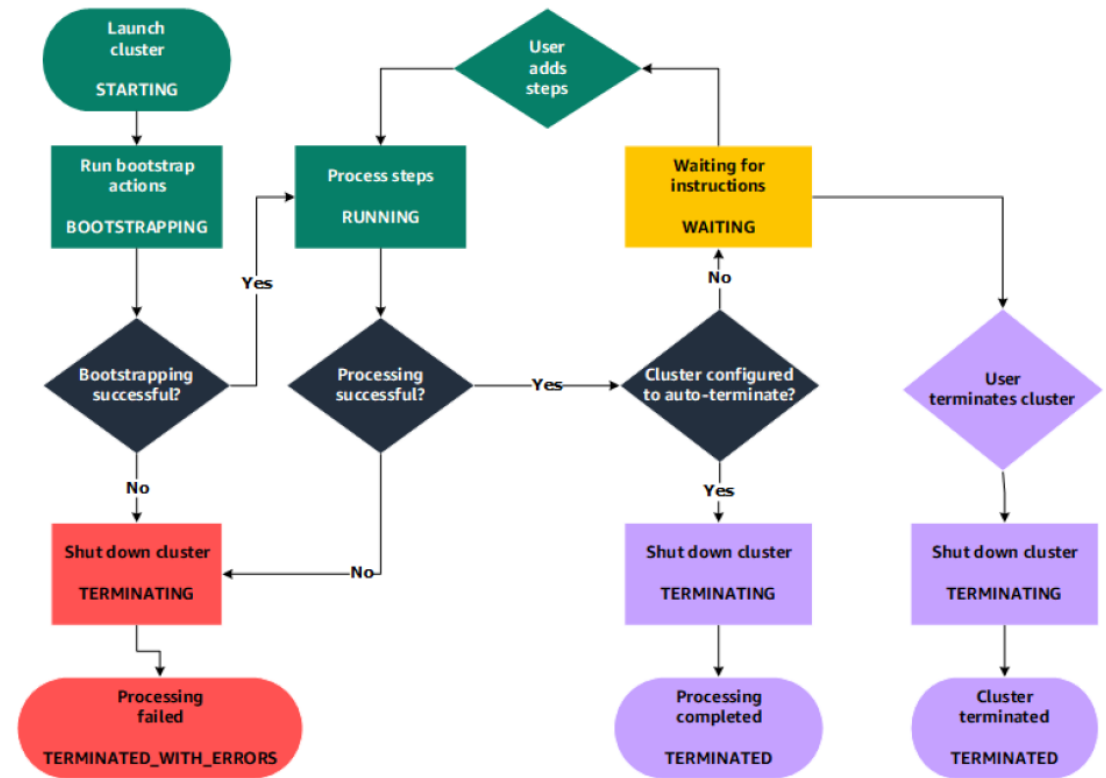
### Lowest-price strategy

- Allocates instances in pools with lowest price at time of fulfillment

# Cluster lifecycle

Creating a cluster (it takes ~10 minutes)

- A cluster cannot be stopped
- It can only be terminated



# Cluster lifecycle

STARTING: EMR provisions EC2 instances for each required instance

BOOTSTRAPPING: EMR runs actions that you specify on each instance

- E.g., install custom applications and perform customizations

Amazon EMR installs the native applications

- E.g., Hive, Hadoop, Spark, and so on

RUNNING: a step for the cluster is currently being run

- Cluster sequentially runs any steps that you specified when you created the cluster

WAITING: after steps run successfully

TERMINATING: after manual shut down

- Any data stored on the cluster is deleted

# Cluster lifecycle

A **step** is a user-defined unit of processing

- E.g., one algorithm that manipulates the data

## Step states

- **PENDING**: The step is waiting to be run
- **RUNNING**: The step is currently running
- **COMPLETED**: The step completed successfully
- **CANCELLED**: The step was cancelled before running because an earlier step failed
- **FAILED**: The step failed while running



# Creating the cluster

Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	<input checked="" type="radio"/> On-demand <sup>i</sup> <div>Current on-demand price \$0.192 per instance/hr</div> <input type="radio"/> Spot <sup>i</sup> Use on-demand as max price
Core Core - 2	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	<input type="text" value="1"/> Instances	<input checked="" type="radio"/> On-demand <sup>i</sup> <input type="radio"/> Spot <sup>i</sup> Use on-demand as max price
Task Task - 3	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	<input type="text" value="1"/> Instances	<input checked="" type="radio"/> On-demand <sup>i</sup> <span>✕</span> <input type="radio"/> Spot <sup>i</sup> Use on-demand as max price

Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	<input type="radio"/> On-demand <sup>i</sup> <input checked="" type="radio"/> Spot <sup>i</sup> <div>Current spot price Availability zone Price us-east-1a \$0.073 us-east-1b \$0.073 us-east-1c <b>\$0.069</b> lowest us-east-1d \$0.074 us-east-1f \$0.072</div> Use on-demand as max price
Core Core - 2	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	<input type="text" value="1"/> Instances	<input checked="" type="radio"/> On-demand <sup>i</sup> <input type="radio"/> Spot <sup>i</sup> Use on-demand as max price
Task Task - 3	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	<input type="text" value="1"/> Instances	<input checked="" type="radio"/> On-demand <sup>i</sup> <span>✕</span> <input type="radio"/> Spot <sup>i</sup> Use on-demand as max price

# Creating the cluster

## Choose to launch **master**, **core**, or **task** on Spot Instances

- The **master** node controls the cluster
  - When terminated, the cluster ends
  - Use *spot instances* if you are running a cluster where sudden termination is acceptable
- **Core** nodes process data and store information using HDFS
  - When terminated, data is lost
  - Use *spot instances* when partial HDFS data loss is tolerable
- **Task** nodes process data but do not hold persistent data in HDFS
  - When terminated, computational capacity is lost
  - The effect of spot instances on the cluster is "minimal"

<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-instances-guidelines.html>

# Creating the cluster

Application Scenario	Master Node Purchasing Option	Core Nodes Purchasing Option	Task Nodes Purchasing Option
Long-Running Clusters and Data Warehouses	On-Demand	On-Demand or instance-fleet mix	Spot or instance-fleet mix
Cost-Driven Workloads	Spot	Spot	Spot
Data-Critical Workloads	On-Demand	On-Demand	Spot or instance-fleet mix
Application Testing	Spot	Spot	Spot

# Creating the cluster

## Amazon EMR provides two main file systems

- **HDFS** and **EMRFS**, specify which file system to use by the prefix
- `hdfs://path` (or just ``path``)
  - HDFS is used by the master and core nodes
  - **AWS EBS volume storage is used for HDFS data**
  - Is fast, best used for caching the results produced by intermediate job-flow steps, **why?**
  - It's ephemeral storage which is reclaimed when the cluster ends
- `s3://DOC-EXAMPLE-BUCKET1/path` (EMRFS)
  - An implementation of the Hadoop file system atop Amazon S3
  - We can avoid EBS storage

<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-storage.html>

# Creating the cluster

Choose the frameworks and applications to install

Data process

- Submit jobs or queries directly to installed applications
- Run steps in the cluster

Submitting jobs

- Connect to the master node over a secure connection
- Access the interfaces and tools that are available on your cluster

# Creating the cluster

Create Cluster - **Advanced Options** [Go to quick options](#)

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

### Software Configuration

Release **emr-6.2.0** ⓘ

<input checked="" type="checkbox"/> Hadoop 3.2.1	<input type="checkbox"/> Zeppelin 0.9.0	<input type="checkbox"/> Livy 0.7.0
<input type="checkbox"/> JupyterHub 1.1.0	<input type="checkbox"/> Tez 0.9.2	<input type="checkbox"/> Flink 1.11.2
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 2.2.6-amzn-0	<input checked="" type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 3.1.2	<input type="checkbox"/> Presto 0.238.3	<input type="checkbox"/> PrestoSQL 343
<input type="checkbox"/> ZooKeeper 3.4.14	<input checked="" type="checkbox"/> JupyterEnterpriseGateway 2.1.0	<input type="checkbox"/> MXNet 1.7.0
<input type="checkbox"/> Sqoop 1.4.7	<input checked="" type="checkbox"/> Hue 4.8.0	<input type="checkbox"/> Phoenix 5.0.0
<input type="checkbox"/> Oozie 5.2.0	<input checked="" type="checkbox"/> Spark 3.0.1	<input type="checkbox"/> HCatalog 3.1.2
<input type="checkbox"/> TensorFlow 2.3.1		

Multiple master nodes (optional)

☐ Use multiple master nodes to improve cluster availability. [Learn more](#) ⓘ

AWS Glue Data Catalog settings (optional)

☐ Use for Hive table metadata ⓘ

☐ Use for Spark table metadata ⓘ

Edit software settings ⓘ

☒ Enter configuration ☐ Load JSON from S3

```
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
```

### Steps (optional)

A step is a unit of work you submit to the cluster. For instance, a step might contain one or more Hadoop or Spark jobs. You can also submit additional steps to a cluster after it is running. [Learn more](#) ⓘ

Concurrency: ☐ Run multiple steps at the same time to improve cluster utilization

After last step completes: ☒ Clusters enters waiting state

☐ Cluster auto-terminates

Step type **Select a step** [Add step](#)

# Creating the cluster

## Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

Node type	Instance type	Instance count	Purchasing option
<b>Master</b> Master - 1	<b>m5.xlarge</b> 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	<input type="radio"/> On-demand <input checked="" type="radio"/> Spot Use on-demand as max price
<b>Core</b> Core - 2	<b>m5.xlarge</b> 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price
<b>Task</b> Task - 3	<b>m5.xlarge</b> 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB Add configuration settings	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price

Current spot price

Availability zone	Price
us-east-1a	\$0.073
us-east-1b	\$0.073
us-east-1c	\$0.069 lowest
us-east-1d	\$0.074
us-east-1f	\$0.072

# Creating the cluster

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps  
Step 2: Hardware  
**Step 3: General Cluster Settings**  
Step 4: Security

### General Options

Cluster name

☒ Logging ⓘ  
S3 folder  ⓘ

☐ Log encryption ⓘ

☒ Debugging ⓘ

☒ Termination protection ⓘ

### Tags ⓘ

Key	Value (optional)
<input type="text" value="Add a key to create a tag"/>	<input type="text"/>

### Additional Options

☐ EMRFS consistent view ⓘ

Custom AMI ID  ⓘ

► Bootstrap Actions

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



# Creating the cluster

## Create Cluster - Advanced Options

[Go to quick options](#)

[Step 1: Software and Steps](#)  
[Step 2: Hardware](#)  
[Step 3: General Cluster Settings](#)  
**| Step 4: Security**

### Security Options

EC2 key pair  [i](#)

☒ Cluster visible to all IAM users in account [i](#)

#### Permissions [i](#)

☒ Default ☐ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role [EMR\\_DefaultRole](#) [i](#)

EC2 instance profile [EMR\\_EC2\\_DefaultRole](#) [i](#)

Auto Scaling role [EMR\\_AutoScaling\\_DefaultRole](#) [i](#)

[▶ Security Configuration](#)

[▶ EC2 security groups](#)

[Cancel](#) [Previous](#) [Create cluster](#)

Allows EMR to call other AWS Services such as EC2 on your behalf.

Provides access to other AWS services such as S3, DynamoDB from EC2 instances that are launched by EMR.

# Creating the cluster

## Using CLI (command line interface)

```
aws emr create-cluster --auto-scaling-role EMR_AutoScaling_DefaultRole --termination-protected --
applications Name=Hadoop Name=Hive Name=Hue Name=JupyterEnterpriseGateway Name=Spark --ebs-root-volume-
size 10 --ec2-attributes
'{"KeyName":"bigdata","InstanceProfile":"EMR_EC2_DefaultRole","SubnetId":"subnet-
5fa2f912","EmrManagedSlaveSecurityGroup":"sg-07818b5690a50b3f1","EmrManagedMasterSecurityGroup":"sg-
0e2f5550a2cb98f79"}' --service-role EMR_DefaultRole --enable-debugging --release-label emr-6.2.0 --log-
uri 's3n://aws-logs-604905954159-us-east-1/elasticmapreduce/' --name 'BigData' --instance-groups
'[{"InstanceCount":1,"BidPrice":"OnDemandPrice","EbsConfiguration":{"EbsBlockDeviceConfigs":[{"VolumeSpe
cification":{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"InstanceGroupType":"MASTER","I
nstanceType":"m4.xlarge","Name":"Master -
1"}, {"InstanceCount":1,"BidPrice":"OnDemandPrice","EbsConfiguration":{"EbsBlockDeviceConfigs":[{"VolumeS
pecification":{"SizeInGB":32,"VolumeType":"gp2"},"VolumesPerInstance":2}]},"InstanceGroupType":"CORE","I
nstanceType":"m4.xlarge","Name":"Core - 2"}]' --scale-down-behavior TERMINATE_AT_TASK_COMPLETION --
region us-east-1
```

# Running the cluster

Amazon EMR

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on EKS

Virtual clusters

Help

What's new

Clone Terminate AWS CLI export

Cluster: **BigData** Starting

Summary Application user interfaces Monitoring Hardware Configurations Events Steps Bootstrap actions

Summary

ID: j-EUO6QT8VQRA1

Creation date: 2021-03-22 15:14 (UTC+1)

Elapsed time: 0 seconds

After last step completes: Cluster waits

Termination protection: On [Change](#)

Tags: -- [View All](#) / [Edit](#)

Master public DNS: --

Network and hardware

Availability zone: --

Subnet ID: [subnet-5fa2f912](#)

Master: Provisioning 1 m5.xlarge  
Spot (max on-demand)

Core: Provisioning 1 m5.xlarge

Task: Provisioning 1 m5.xlarge

Cluster scaling: Not enabled

Configuration details

Release label: emr-6.2.0

Hadoop distribution: Amazon 3.2.1

Applications: Hive 3.1.2, Hue 4.8.0, JupyterEnterpriseGateway 2.1.0, Spark 3.0.1

Log URI: s3://aws-logs-604905954159-us-east-1/elasticmapreduce/

EMRFS consistent view: Disabled

Custom AMI ID: --

Security and access

Key name: bigdata

EC2 instance profile: EMR\_EC2\_DefaultRole

EMR role: EMR\_DefaultRole

Auto Scaling role: EMR\_AutoScaling\_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master: [sg-0e2f5550a2cb98f79](#) (ElasticMapReduce-master)

Security groups for Core & Task: [sg-07818b5690a50b3f1](#) (ElasticMapReduce-slave)

Application user interfaces

Persistent user interfaces

On-cluster user interfaces

# Running the cluster

The screenshot displays the Amazon EMR console interface. On the left, a navigation sidebar lists various services under 'Amazon EMR' and 'EMR on EC2'. The main content area shows the details for a cluster named 'BigData', which is currently in a 'Waiting' state. A blue callout box with the text 'DNS name' points to the 'Master public DNS' field in the 'Summary' tab. The console is divided into several sections: 'Summary', 'Configuration details', 'Application user interfaces', 'Network and hardware', and 'Security and access'. Each section provides specific information about the cluster's configuration and status.

**Amazon EMR**

EMR on EC2

- Clusters
- Notebooks
- Git repositories
- Security configurations
- Block public access
- VPC subnets
- Events
- EMR on EKS
- Virtual clusters

Help

What's new

Cluster: BigData **Waiting** Cluster ready after last step completed.

**Summary** | Application user interfaces | Monitoring | Hardware | Configurations | Events | Steps | Bootstrap actions

**Summary**

ID: j-EUO6QT8VQRA1

Creation date: 2021-03-22 15:14 (UTC+1)

Elapsed time: 2 hours, 33 minutes

After last step completes: Cluster waits

Termination protection: On [Change](#)

Tags: -- [View All / Edit](#)

**Master public DNS:** ec2-54-227-86-20.compute-1.amazonaws.com [Copy](#)  
Connect to the Master Node Using SSH

**Configuration details**

Release label: emr-6.2.0

Hadoop distribution: Amazon 3.2.1

Applications: Hive 3.1.2, Hue 4.8.0, JupyterEnterpriseGateway 2.1.0, Spark 3.0.1

Log URI: s3://aws-logs-604905954159-us-east-1/elasticmapreduce/ [View](#)

EMRFS consistent view: Disabled

Custom AMI ID: --

**Application user interfaces**

Persistent user interfaces [View](#): Spark history server, YARN timeline server, Tez UI

On-cluster user interfaces [View](#): Not Enabled [Enable an SSH Connection](#)

**Network and hardware**

Availability zone: us-east-1b

Subnet ID: [subnet-5fa2f912](#) [View](#)

Master: **Running** 1 m5.xlarge Spot (max on-demand)

Core: **Running** 1 m5.xlarge

Task: **Running** 1 m5.xlarge

Cluster scaling: Not enabled

**Security and access**

Key name: bigdata

EC2 instance profile: EMR\_EC2\_DefaultRole

EMR role: EMR\_DefaultRole

Auto Scaling role: EMR\_AutoScaling\_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master: [sg-0e2f5550a2cb98f79](#) [View](#) (ElasticMapReduce-master)

Security groups for Core & Task: [sg-07818b5690a50b3f1](#) [View](#) (ElasticMapReduce-slave)

# Running a notebook

The screenshot shows the 'Create notebook' page in the Amazon EMR console. The left sidebar contains navigation links for Amazon EMR, EMR on EC2 (Clusters, Notebooks, Git repositories, Security configurations, Block public access, VPC subnets, Events), EMR on EKS (Virtual clusters), Help, and What's new. The main content area is titled 'Create notebook' and 'Name and configure your notebook'. It includes a sub-header 'Name your notebook, choose a cluster or create one, and customize configuration options if desired. [Learn more](#)'. The form fields are: 'Notebook name\*' (MyNotebook), 'Description' (256 characters max), 'Cluster\*' (Choose an existing cluster, Choose, BigData, j-EUO6QT8VQRA1, Create a cluster), 'Security groups' (Use default security groups, Choose security groups (vpc-2af45357)), 'AWS service role\*' (EMR\_Notebooks\_DefaultRole), 'Notebook location\*' (Use the default S3 location, s3://aws-emr-resources-604905954159-us-east-1/notebooks/, Choose an existing S3 location in us-east-1), 'Git repository' (Link to a Git repository, Choose repository), and 'Tags'. At the bottom, there is a '\* Required' label and 'Cancel' and 'Create notebook' buttons.

Amazon EMR

EMR on EC2

- Clusters
- Notebooks
- Git repositories
- Security configurations
- Block public access
- VPC subnets
- Events

EMR on EKS

- Virtual clusters

Help

What's new

## Create notebook

### Name and configure your notebook

Name your notebook, choose a cluster or create one, and customize configuration options if desired. [Learn more](#)

**Notebook name\*** MyNotebook  
Names may only contain alphanumeric characters, hyphens (-), or underscores (\_).

**Description**  
256 characters max.

**Cluster\*** ☒ Choose an existing cluster  
 BigData j-EUO6QT8VQRA1 [?](#)

☐ Create a cluster [?](#)

**Security groups** ☒ Use default security groups [?](#)  
☐ Choose security groups (vpc-2af45357)

**AWS service role\*** EMR\_Notebooks\_DefaultRole [?](#)

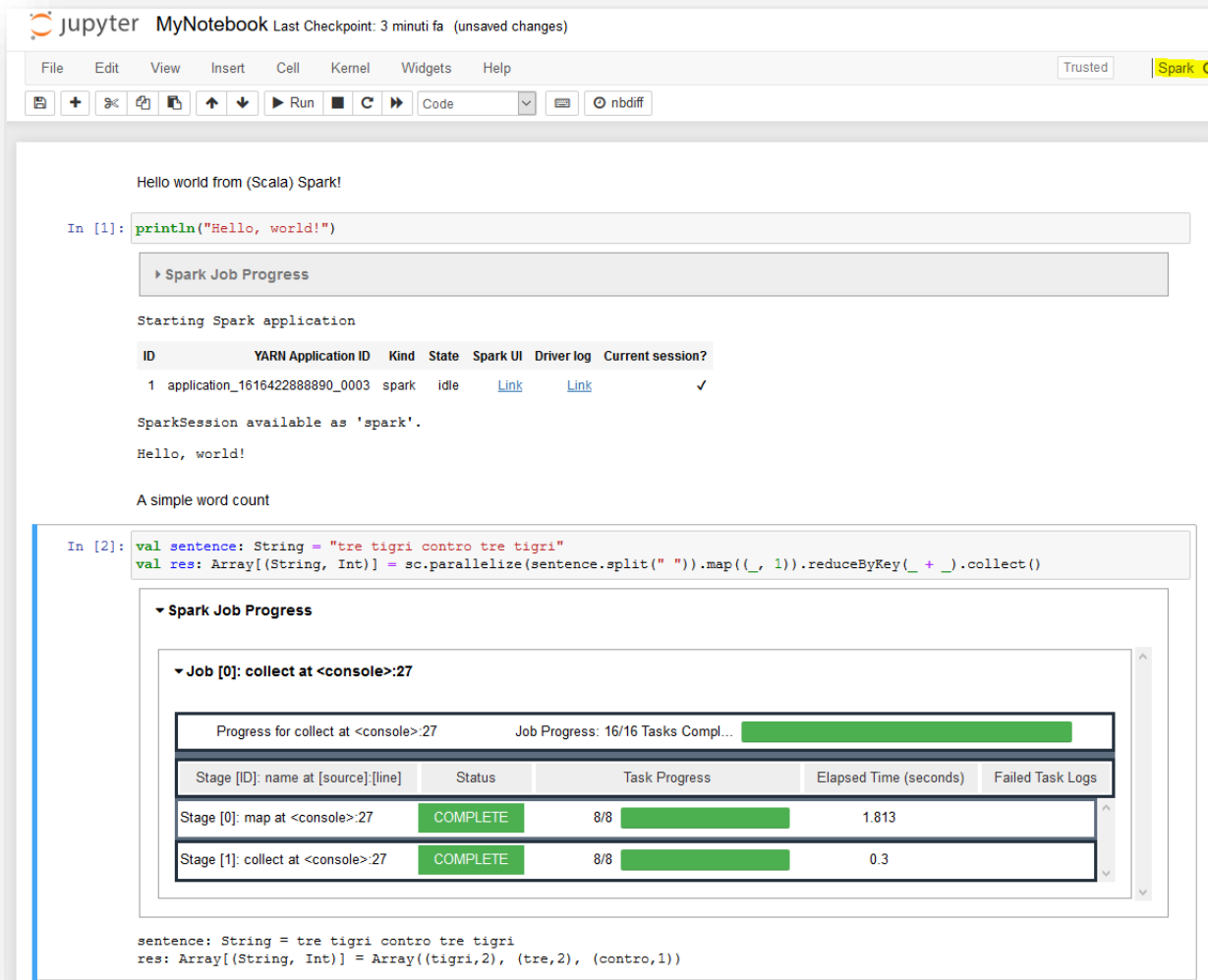
**Notebook location\*** Choose an S3 location where files for this notebook are saved.  
☒ Use the default S3 location  
s3://aws-emr-resources-604905954159-us-east-1/notebooks/  
☐ Choose an existing S3 location in us-east-1

**Git repository**

**Tags** [?](#)

\* Required

# Running a notebook



MyNotebook Last Checkpoint: 3 minuti fa (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Spark

Code nbdiff

Hello world from (Scala) Spark!

```
In [1]: println("Hello, world!")
```

Spark Job Progress

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
1	application_1616422888890_0003	spark	idle	<a href="#">Link</a>	<a href="#">Link</a>	✓

SparkSession available as 'spark'.

Hello, world!

A simple word count

```
In [2]: val sentence: String = "tre tigri contro tre tigri"
val res: Array[(String, Int)] = sc.parallelize(sentence.split(" ")).map(_._1).reduceByKey(_+_).collect()
```

Spark Job Progress

Job [0]: collect at <console>:27

Progress for collect at <console>:27		Job Progress: 16/16 Tasks Compl...		
Stage [ID]: name at [source]:[line]	Status	Task Progress	Elapsed Time (seconds)	Failed Task Logs
Stage [0]: map at <console>:27	COMPLETE	8/8	1.813	
Stage [1]: collect at <console>:27	COMPLETE	8/8	0.3	

sentence: String = tre tigri contro tre tigri  
res: Array[(String, Int)] = Array((tigri,2), (tre,2), (contro,1))

# Running a notebook

Save the result to HDFS

```
import org.apache.hadoop.fs.{FileSystem, Path}
val fs = FileSystem.get(sc.hadoopConfiguration) // get the file system
val outPutPath = new Path(path)
if (fs.exists(outPutPath)) { // delete the HDFS folder if exists
  fs.delete(outPutPath, true)
}

val hdfsPath: String = "wordcount" // HDFS path
def writeandread(path: String) = {
  sc.parallelize(res).saveAsTextFile(path) // save the RDD
  val rdd = sc.textFile(path) // read it back
  rdd.collect() // print it
}

writeandread(hdfsPath)
```

► Spark Job Progress

```
import org.apache.hadoop.fs.{FileSystem, Path}
fs: org.apache.hadoop.fs.FileSystem = DFS[DFSClient[clientName=DFSClient_NONMAPREDUCE_1600703682_22, ugi=livy (auth:SIMPLE)]]
outPutPath: org.apache.hadoop.fs.Path = wordcount
res28: AnyVal = true
hdfsPath: String = wordcount
writeandread: (path: String)Array[String]
res32: Array[String] = Array((tigri,2), (tre,2), (contro,1))
```

... and to S3 as well

```
val s3bucket: String = "s3://aws-emr-resources-604905954159-us-east-1/wordcount"
writeandread(s3bucket)
```