HADOOP ARCHITECTURAL PATTERNS

Module 08

Frameworks for Thinking About Big Data Technology

Mid-Term Exam

Overview

- When
  - The exam will be made available on the blackboard on Thursday evening 10/21 at the start of class, 6:45 pm Chicago time
  - It must be completed and submitted or grading by no later than than Friday evening 10/22 at 6:45 pm Chicago time
  - Of course, please adjust this for your time zone to what makes the most sense
- Where
  - If you are registered for the in-person session take the exam in class
  - Otherwise if you are remote, or have permission from me, take the exam via the blackboard any time during the above period
- Duration
  - The exam will be timed to 2 hours and 45 minutes

Mid-Term Exam

Taking the Exam

- Go to the blackboard section “Quizzes and Exams”
- Pick one of two exams, based on your student id number
- Open the exam in a browser and NEVER leave the exam page
- Log on to the Blackboard again and open up a separate session when attempting to access any other blackboard materials
- The exam is “open book” similar to our quizzes. You may refer to your notes or any materials on our blackboard site
- By departmental policy you cannot use Google search for looking up answers
- IN THE CASE OF COMPUTER OR OTHER TECHNICAL ISSUES- contact me via email ASAP

Mid-Term Exam

Question Types

- Mostly (usually worth 3 points per question)
  - Multiple choice
  - Fill in the blank
  - A lot like quiz questions
- Some (worth 3 or 4 points per question)
  - Short essay
    - Just 2-4 sentences
    - I don't worry about elegance or grammar
  - Code
    - Just a couple of lines
    - I don't worry about missing semi-colons or similar
- I grade the short essay and code questions manually, so you don't get a full score on your exam automatically

Code Question

Example

- Assume you have an input DataFrame, called inDF, which holds the following:

name	age
Afzal	20
Jim	30
Ayesha	40

- Write one line of code, not using SparkSQL, which produces an output DataFrame, called outDF, which holds (copies of) records from inDF having age is less than 35

Mid-Term Exam

Question Categories

- Each question is labeled with one of two phrases
  - MUST ANSWER
  - CAN SKIP

Mid-Term Exam

Question Categories

- MUST ANSWER
  - You must provide an answer to every question in this category
  - If you do not provide an answer, the question is marked incorrect

- CAN SKIP
  - Many multiple choice questions are of this category
  - IT DOES NOT MEAN YOU CAN SKIP THEM ALL
  - You MUST skip exactly four of the questions in this category
    - It is your job to keep count of how many questions you have or need to skip
    - You can't answer all the questions and have me drop four wrong ones
  - This allows you to avoid questions which you might find ambiguous or otherwise problematic
  - If you do not skip four questions of this category…
    - I pick the last four questions you can skip, whether the answer is right or wrong, and skip them for you

Mid-Term Exam
CAN SKIP Questions

- Example

(CAN SKIP) All Professor Rosen's jokes are quite funny

(a) True

(b) False

(c) I want to skip this question

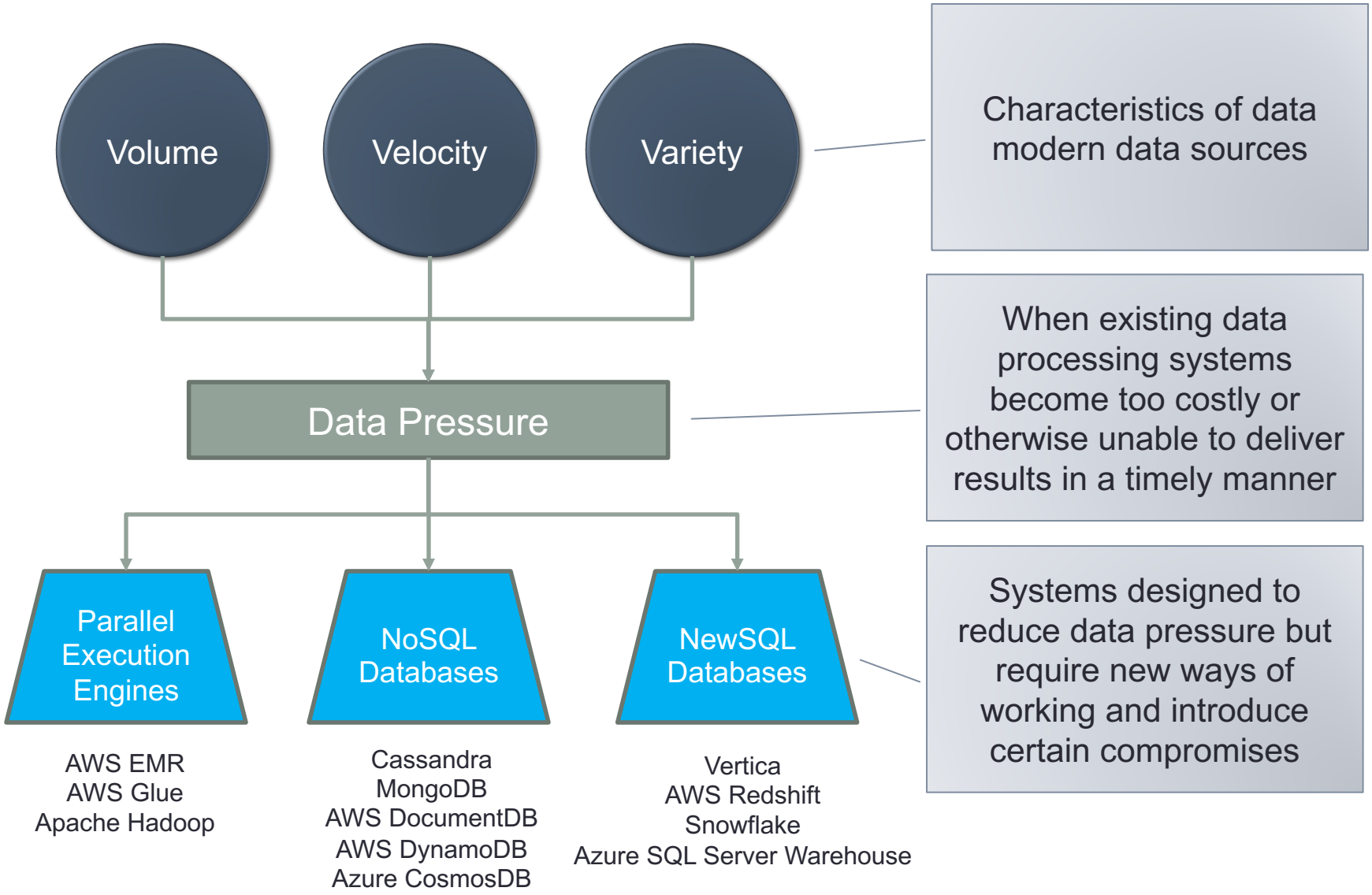
- If you answer (c) it is as if this question disappears from the exam

Mid-Term Exam

Scoring Issues

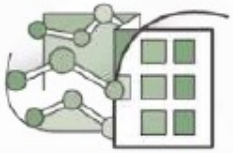
- The Blackboard exam software does not understand about CAN SKIP questions
- And the blackboard thinks that any question answered "I want to skip this question" is wrong and will take away points
  - Don't worry, I fix this during grading
- If the exam has a total of 40 questions, of which you must skip 4…
  - The blackboard still sees 40 questions
  - So it miscalculates the maximum score for the quiz
    - Assume each question is worth 3 points
    - For a 40 question exam, the blackboard thinks the maximum score is then 120
    - But, since you must skip four questions, the maximum score I use for grading is 120 – (4 questions * 3 points/question) = 108
- So I treat a 40 question exam as if it only had 36 questions

Data Pressure: Its Causes and Results



Transient and Persistent Cluster Pattern

Why Amazon EMR?



Easy to Use
Launch a cluster in minutes

Low Cost
Pay an hourly rate

Elastic
Easily add or remove capacity

Reliable
Spend less time monitoring

Secure
Managed firewalls

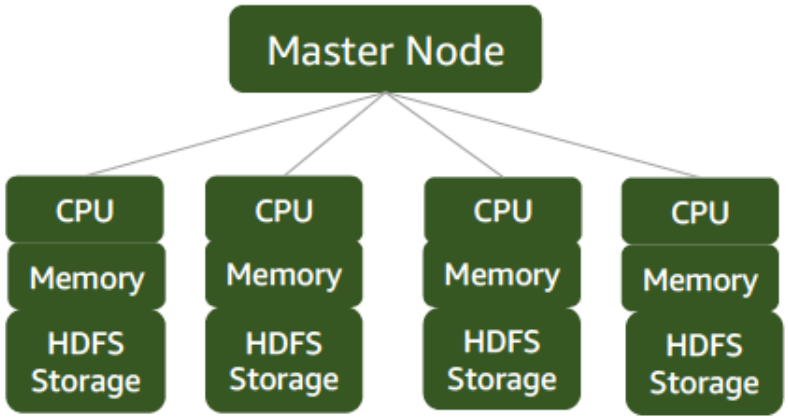
Flexible
You control the cluster

Storage & Compute

Coupled vs. Decoupled

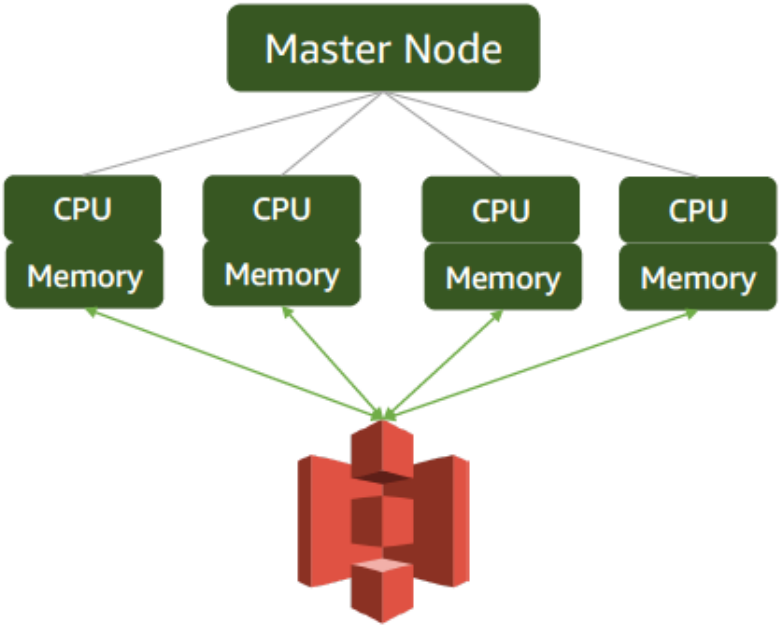
Understanding Decoupled Storage & Compute

Old Clustering / Localized Model



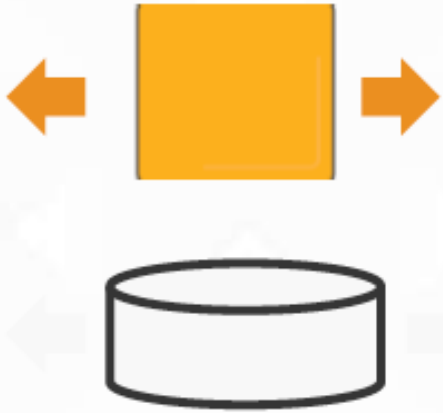
HDFS = 3X Replication
500 TB Dataset = 1.5 PB cluster with replication

Amazon EMR Decoupled Model



S3 as Streaming HDFS via EMRFS

EMR Decouples Storage & Compute



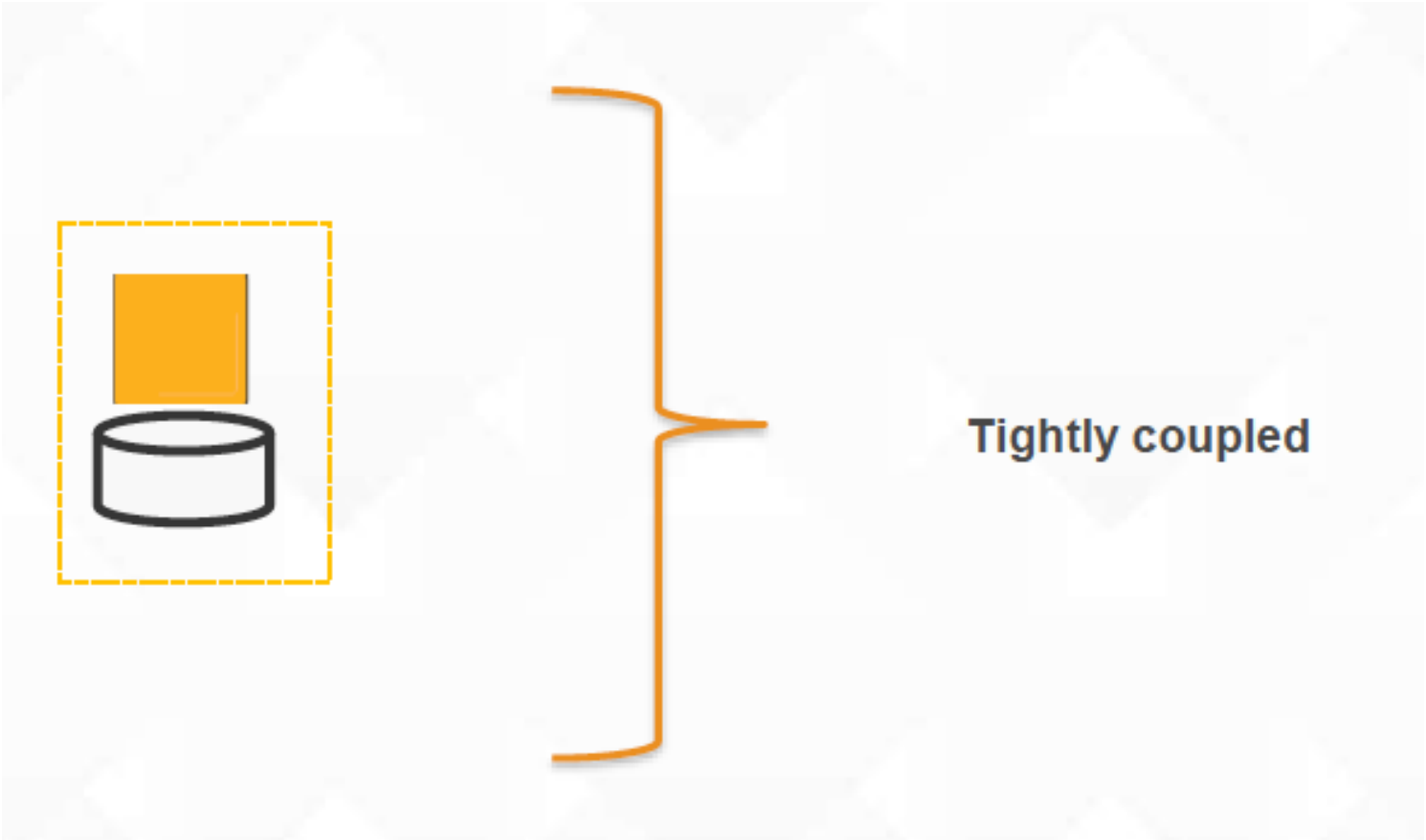
Traditional Hadoop

Tightly-coupled compute & storage
➔ inflexibility

Amazon EMR

Decoupled compute & storage
➔ flexible storage
➔ Right-sizing compute

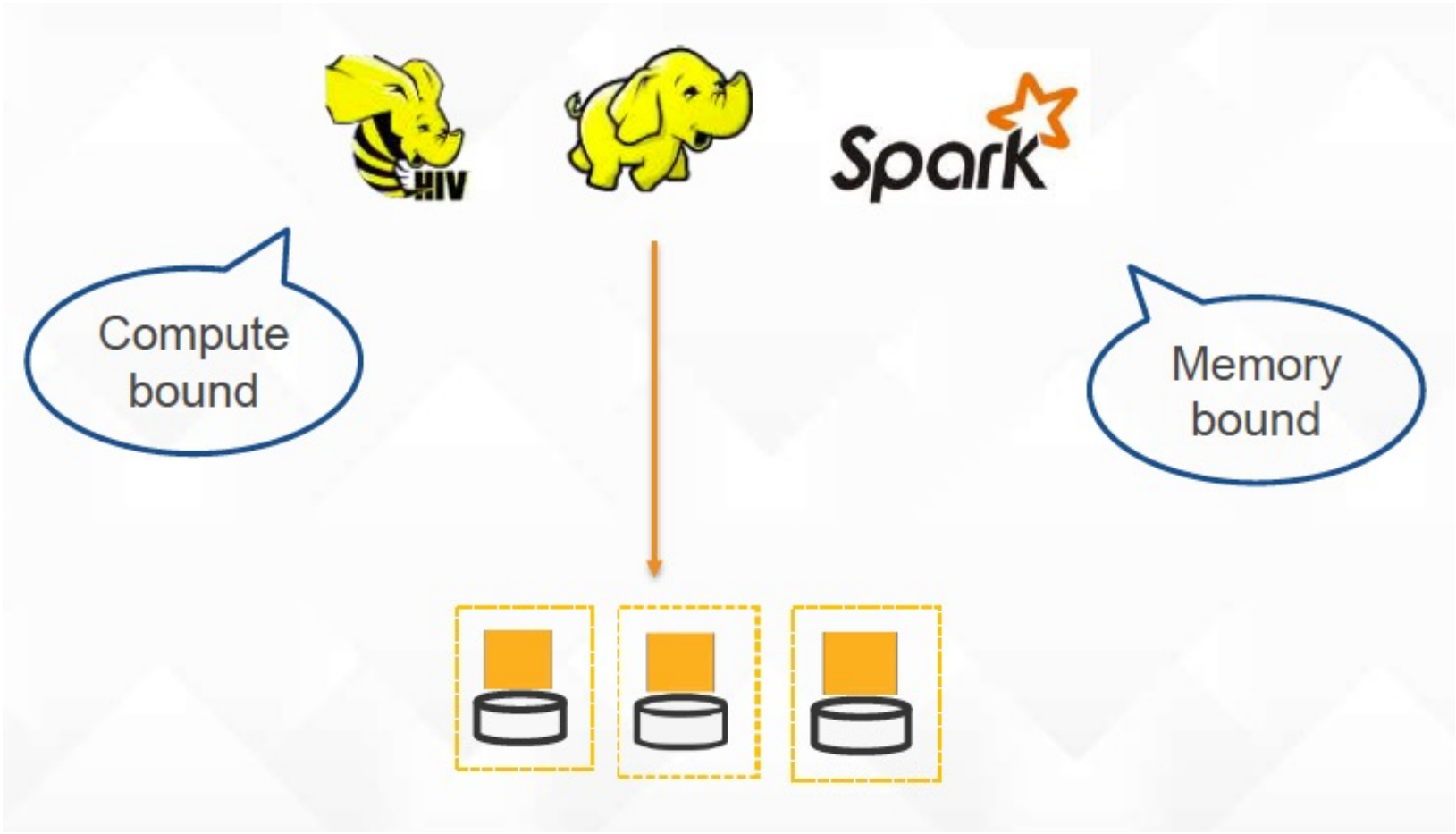
On-Premises Environment or Self-Managed Hadoop Cluster on AWS



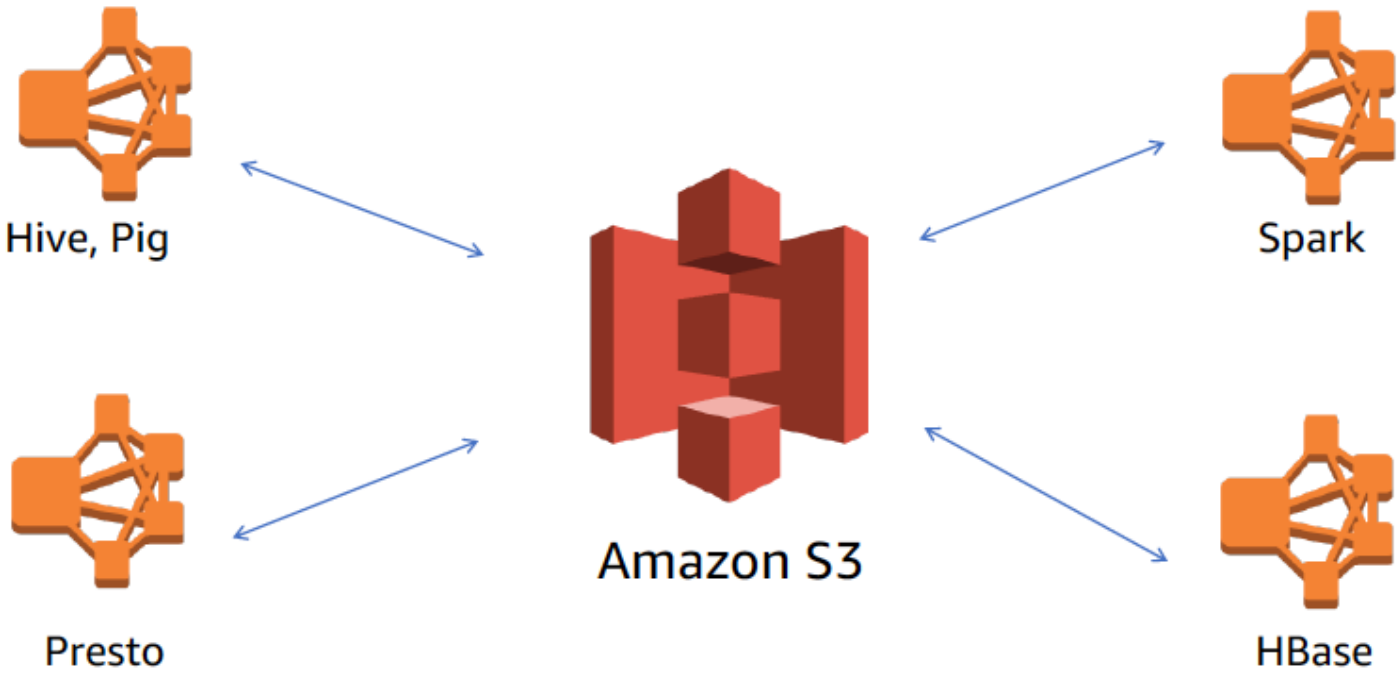
Compute and Storage Grow Together

- Storage grows along with compute
- Compute requirements vary

Contention for Same Resources

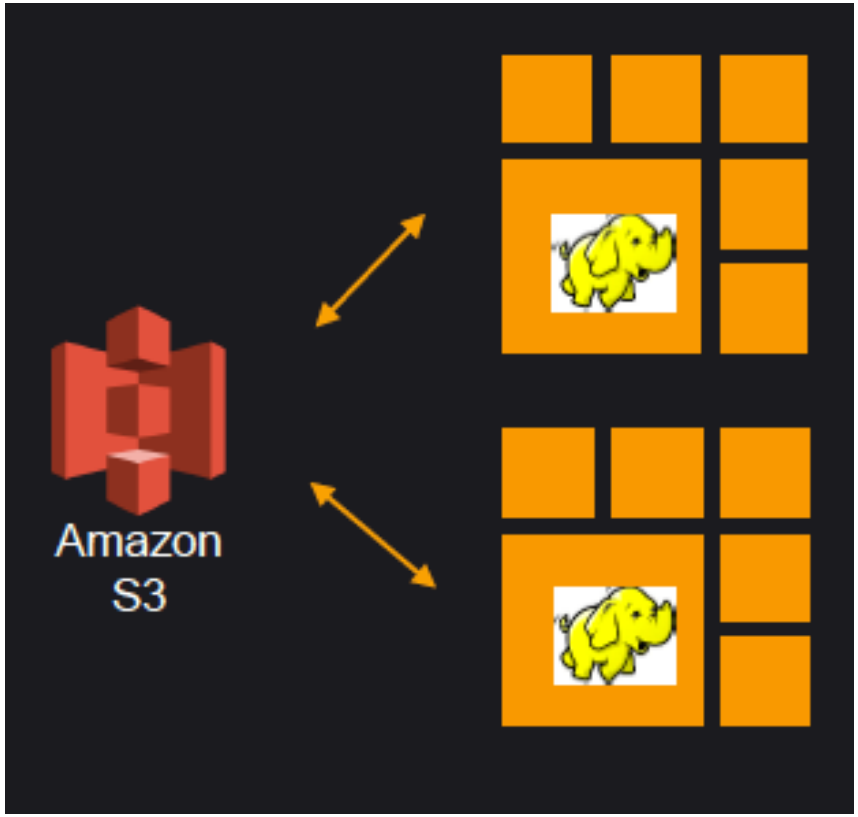


EMR with Amazon S3



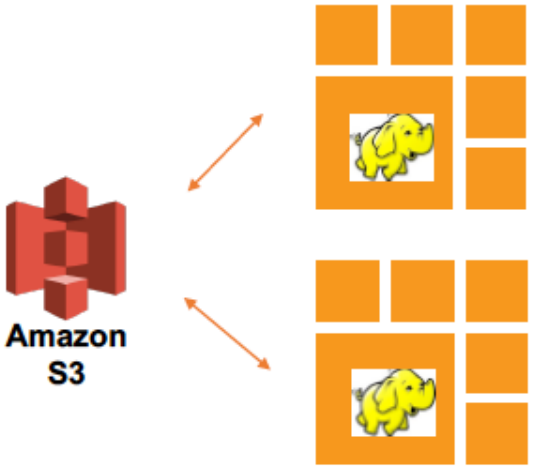
Amazon S3 as your persistent data store

- Separate compute and storage
- Resize and shut down Amazon EMR clusters with no data loss
- Point multiple Amazon EMR clusters at the same data in Amazon S3
- Easily evolve your analytic infrastructure as technology evolves

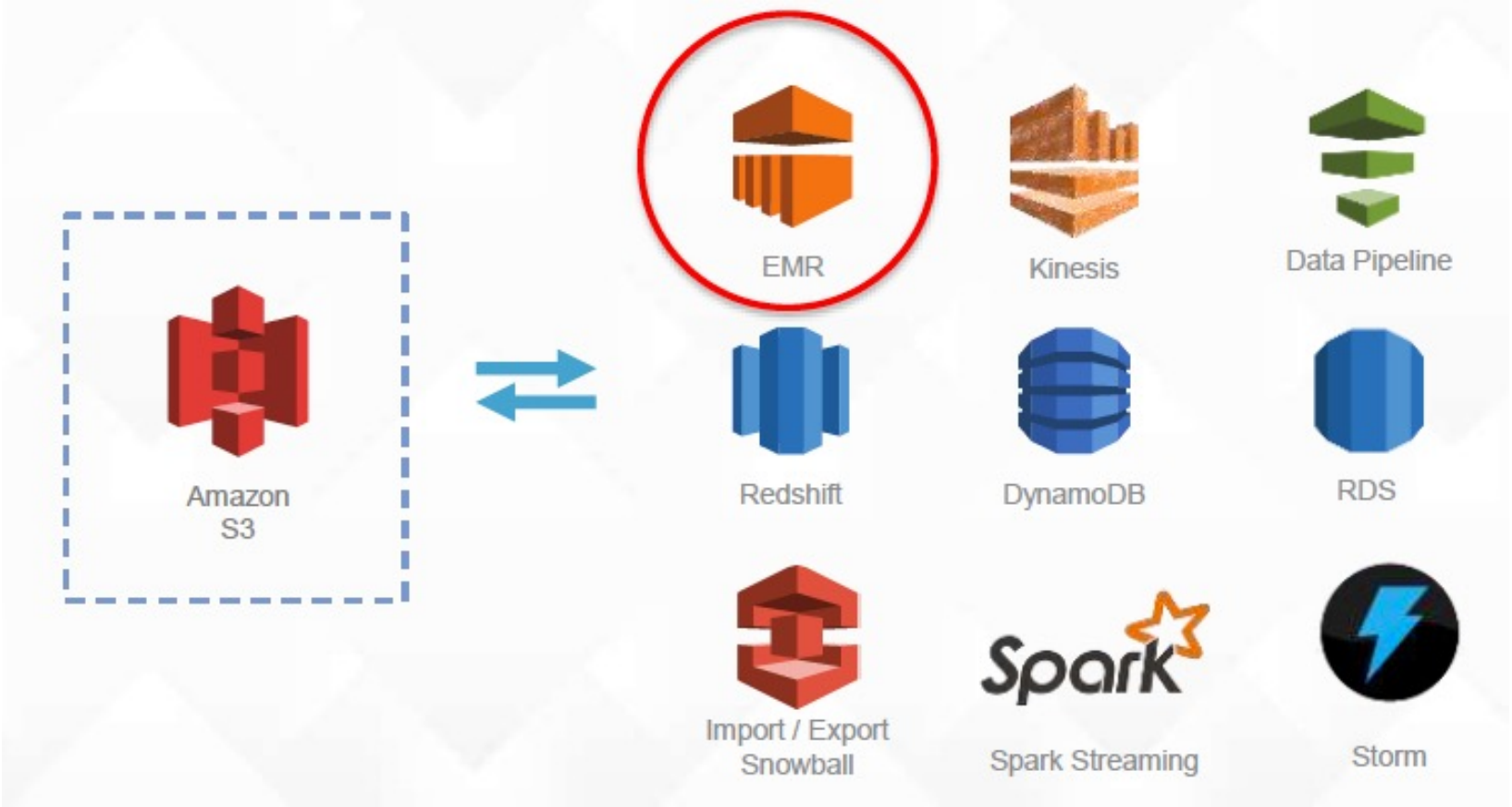


Amazon S3 as your persistent data store

- Designed for 99.999999999% durability
- Separate compute and storage
- Resize and shut down clusters with no data loss
- Point multiple clusters at the same data in S3
- Easily evolve your analytics infrastructure as technology evolves



EMR: Persist Data in S3
Also supported by other services



EMRFS makes it easier to use Amazon S3

- Read-after-write consistency
- Very fast list operations
- Error handling options
- Support for Amazon S3 encryption
- Transparent to applications: s3://
- The EMR File System (EMRFS) is an implementation of HDFS that all Amazon EMR clusters use for reading and writing regular files from Amazon EMR directly to Amazon S3
- EMRFS provides the convenience of storing persistent data in Amazon S3 for use with Hadoop while also providing features like consistent view and data encryption

Going from HDFS to Amazon S3

```
CREATE EXTERNAL TABLE serde_regex (
            host STRING,
            referer STRING,
            agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe')
LOCATION ‘hdfs:///user/samples/pig-apache/input/'
```

Going from HDFS to Amazon S3

```
CREATE EXTERNAL TABLE serde_regex(
            host STRING,
            referer STRING,
            agent STRING)
ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.RegexSerDe')
LOCATION 's3://elasticmapreduce.samples.bucket/pig-apache/input/'
```

HDFS is still there if you need it

- Iterative workloads
  - If you're processing the same dataset more than once
  - Consider using Spark & RDDs for this too
- Disk I/O intensive workloads
  - Persist data on Amazon S3 and use S3DistCp to copy to/from HDFS for processing

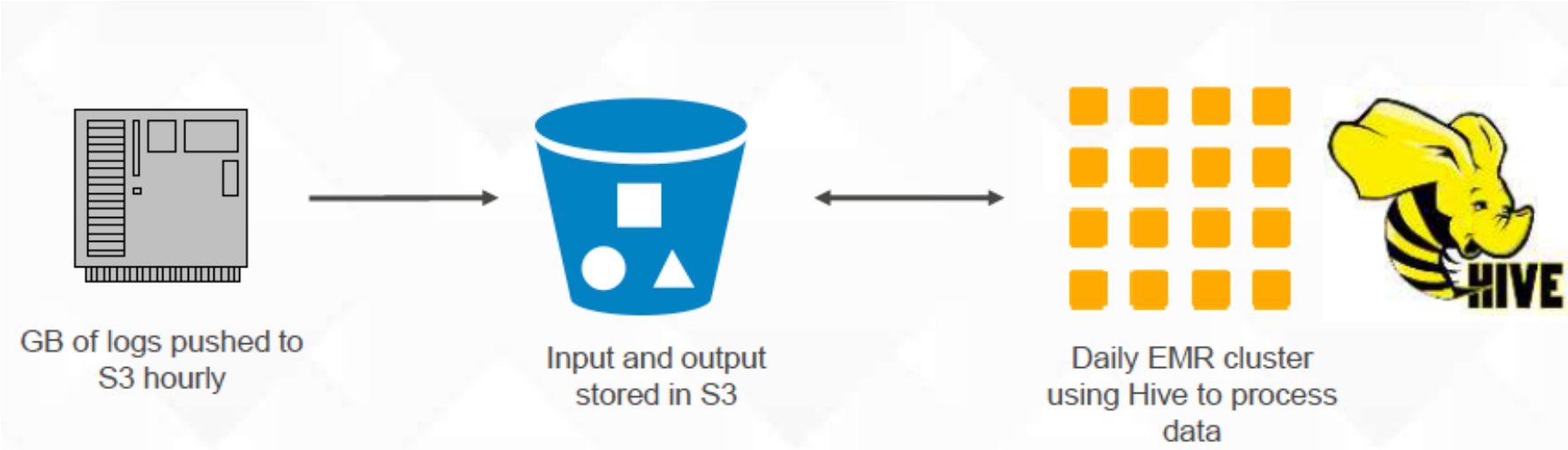
S3DistCp (s3-dist-cp)

- Apache DistCp is an open-source tool you can use to copy large amounts of data.
- *S3DistCp* is an extension of DistCp that is optimized to work with AWS, particularly Amazon S3.
- The command for S3DistCp in Amazon EMR is s3-dist-cp, which you add as a step in a cluster or at the command line
- Using S3DistCp, you can efficiently copy large amounts of data from Amazon S3 into HDFS where it can be processed by subsequent steps in your Amazon EMR cluster.
- You can also use S3DistCp to copy data between Amazon S3 buckets or from HDFS to Amazon S3
- S3DistCp is more scalable and efficient for parallel copying large numbers of objects across buckets and across AWS accounts

Examples

- HDFS was cost prohibitive for our major company use cases
  - Need 30 D2.8XL's just to store two of our tables: ~$1.5M/yr on HDFS vs ~$120K/yr on S3
  - Need 90 D2.8XL's to store all queryable data: ~$4.5M/yr on HDFS vs. $360K/yr on S3
- Data locality is desirable but not practical for their scale
  - EMR & S3 with partitioned data is a great fit
  - Tuned queries & data structures on S3 take ~2X if on HDFS under perfect locality conditions
- Localize data into HDFS on Core nodes using S3DistCp if making 3 or more passes
- Consider tiered storage
  - External tables in Hive can have a blend of some partitions in HDFS and others in S3
  - Introduces operational complexity for partition maintenance

EMR example #1: Batch Processing

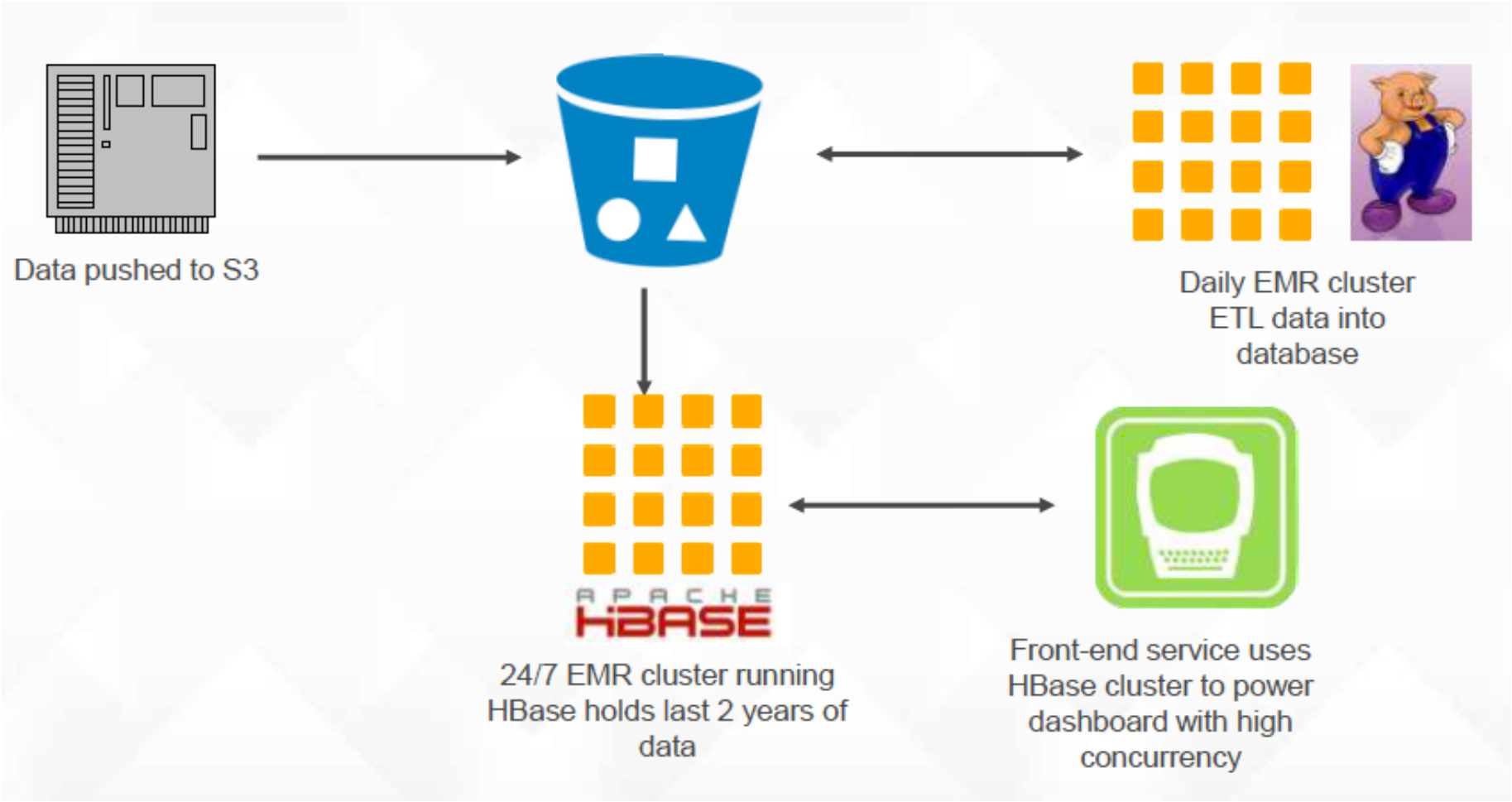


yelp

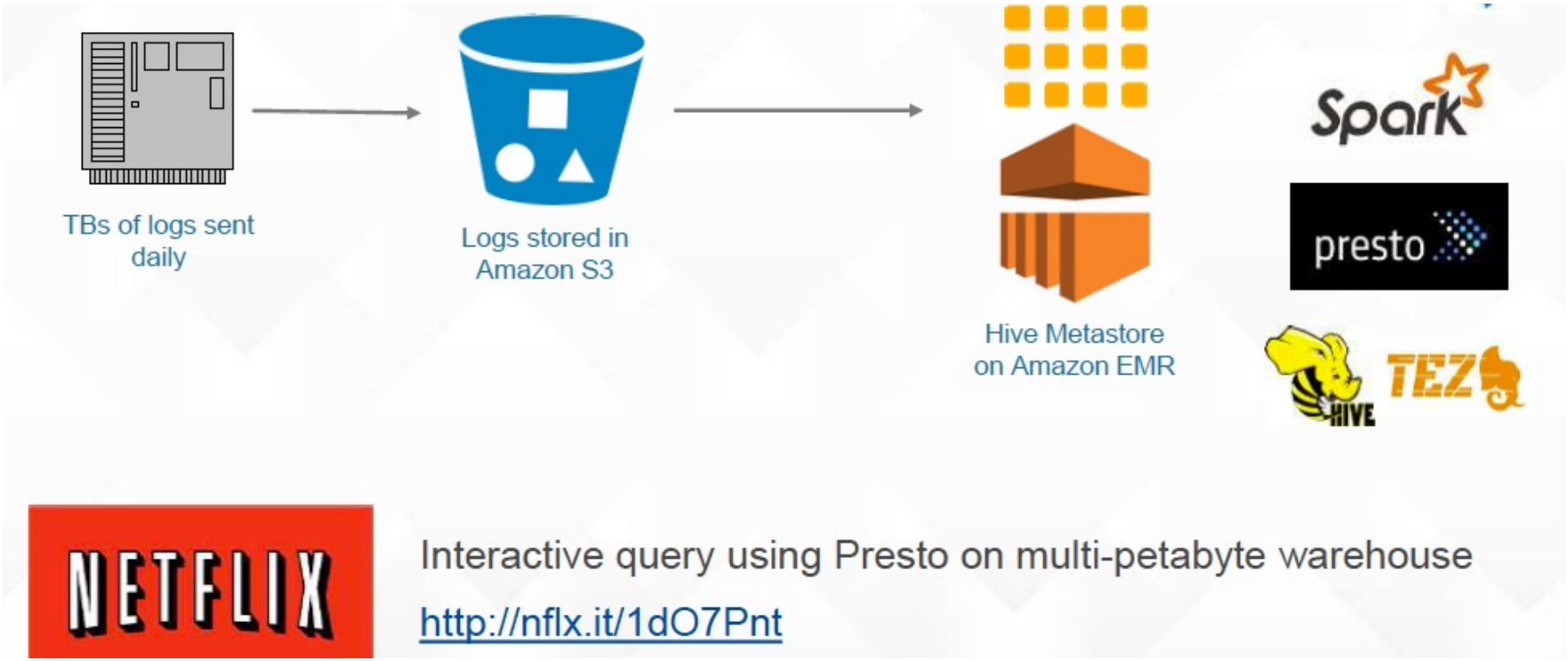
250 Amazon EMR jobs per day, processing 30 TB of data

http://aws.amazon.com/solutions/case-studies/yelp/

EMR example #2: Long-running cluster



EMR example #3: Interactive query



Two architectural patterns

- Transient Clusters
  - Large-scale transformation
  - ETL to other DWH or S3
  - Executing machine learning jobs
- Persistent Clusters
  - Notebooks
  - Experimentation
  - Ad-hoc jobs
  - Streaming
  - Continuous transformation

Transient Clusters

- Cluster lives only for the duration of the job
  - Shut down the cluster when the job is done
- Data persisted on Amazon S3
  - Input & output

amazon

Benefits of Transient Clusters

1. Control your cost
2. Minimum maintenance
   - Cluster goes away when job is done
3. Best Flexibility of Tools
4. Practice cloud architecture
   - Pay for what you use
   - Data processing as a workflow

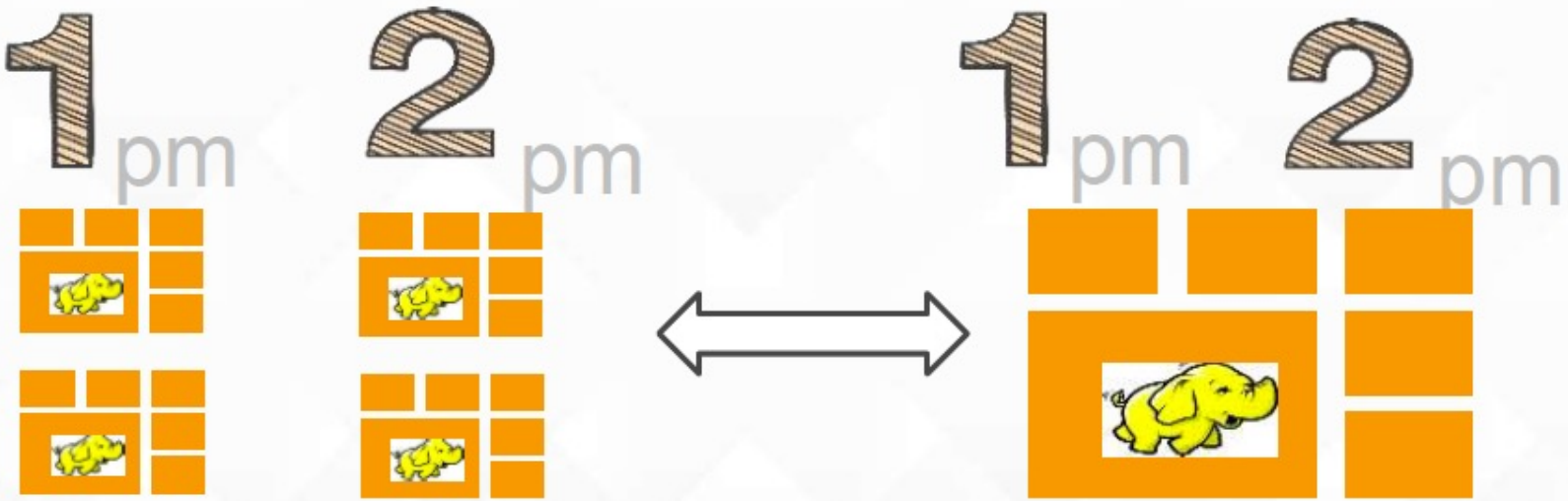
amazon web services

Benefits of Transient Clusters

- Control your cost
- Minimum maintenance
  - Cluster goes away when job is done
  - Best Flexibility of Tools
  - Practice cloud architecture
- Pay for what you use
- Data processing as a workflow

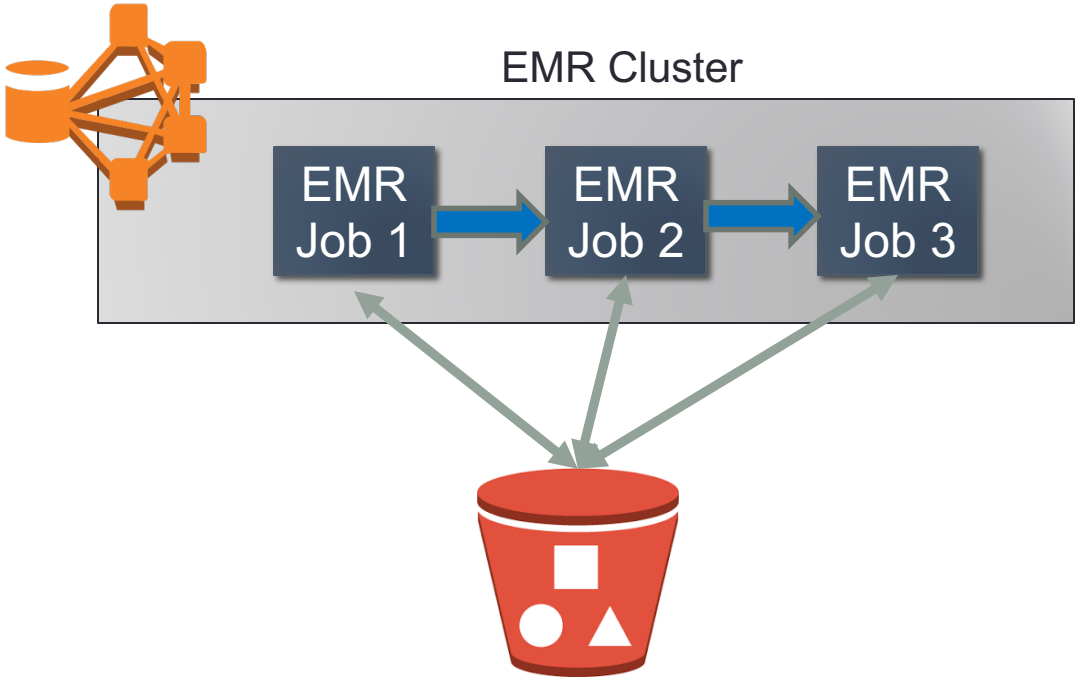
Benefit of Persistent Clusters

- Cost effective for repetitive jobs



Benefits of Persistent Clusters

- Ability to share data between multiple jobs
- Always On for Analyst Access



When to use Persistent clusters?

If (Data Load Time + Processing Time) x Number Of Jobs > 24 hours

Use Persistent Clusters

Else

Use Transient Clusters

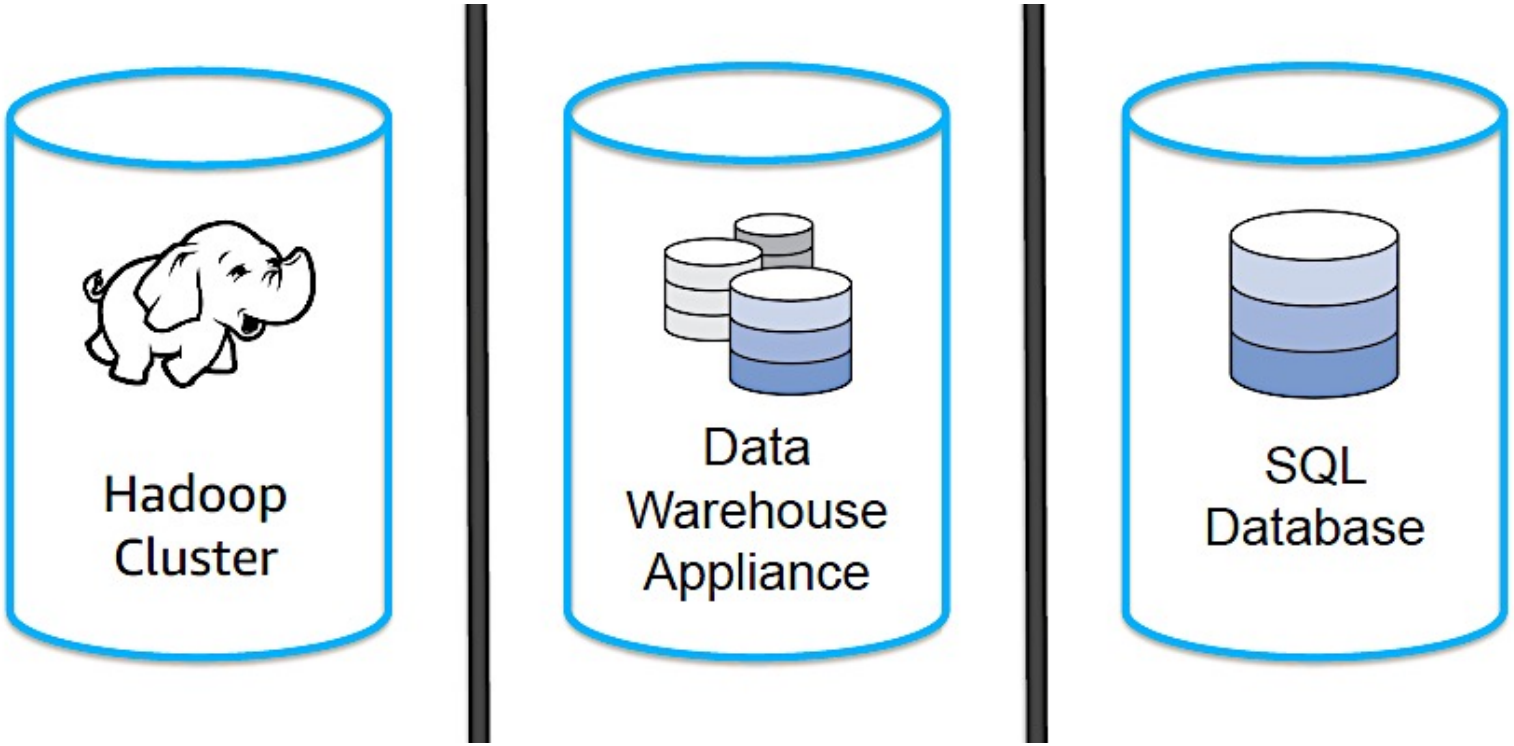
Example

(20min data load + 1 hour Processing time) x 20 jobs = 26 hours

This is > 24 hour, therefore use Persistent Clusters

Data Lake Pattern

Legacy Data Architectures Exist as Isolated Data Silos

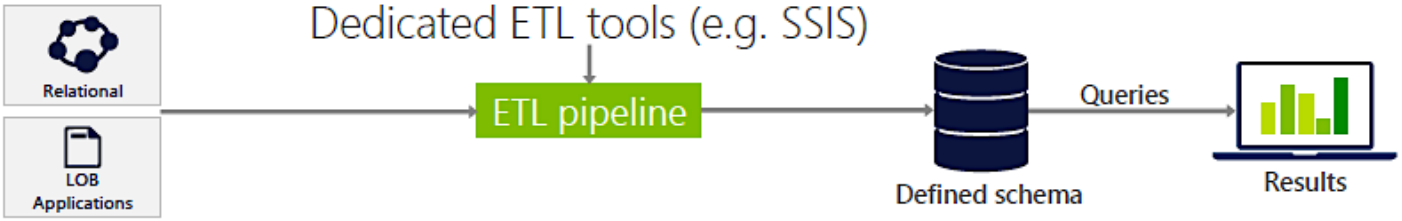


Challenges with Legacy Data Architectures

- Can't move data across silos
- Can't deal with dynamic data and real-time processing
- Can't deal with format diversity and change rate
- Complex ETL processes
  - ETL = Extract (from input files), transform (to desired format), load (to database or data warehouse)
  - Process to ingest, clean, reformat, and store data for analysis
- Difficult to find people with adequate skills to configure and manage these systems
- Can't integrate with the explosion of available social and behavior tracking data

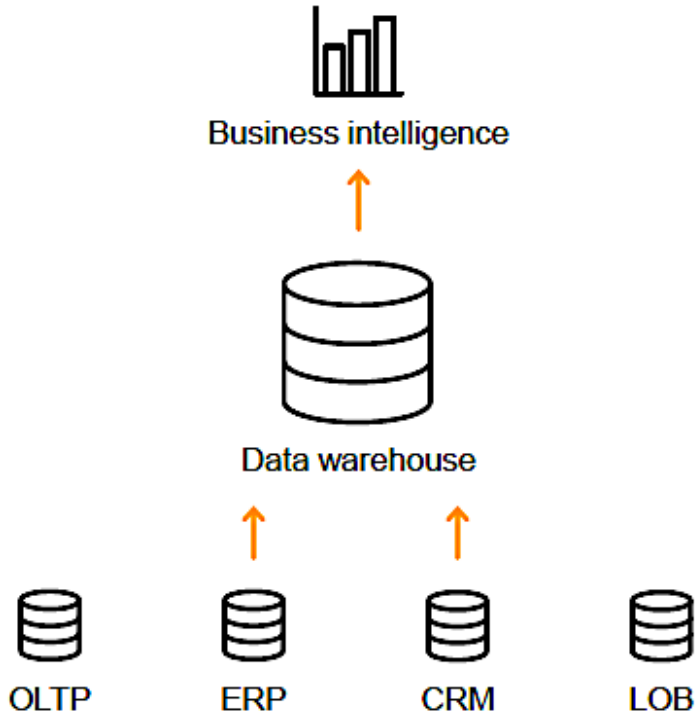
Traditional business analytics process

1. Start with end-user requirements to identify desired reports and analysis
2. Define corresponding database schema and queries
3. Identify the required data sources
4. Create a Extract-Transform-Load (ETL) pipeline to extract required data (curation) and transform it to target schema (*'schema-on-write'*)
5. Create reports. Analyze data



All data not immediately required is discarded or archived

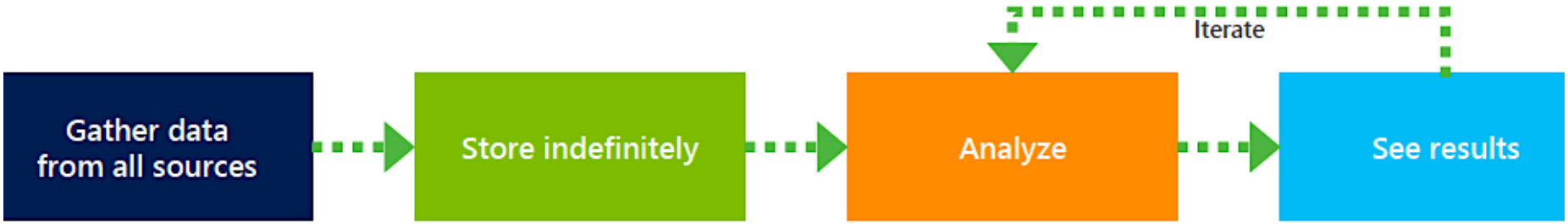
Traditionally, Analytics Used to Look Like This



- Relational data
- TBs–PBs scale
- Schema defined prior to data load
- Operational reporting and ad hoc
- Large initial CAPEX + $10K–$50K/TB/year

New big data thinking: All data has value

- All data has potential value
- Data hoarding
- No defined schema—stored in native format
- Schema is imposed and transformations are done at query time *(schema-on-read)*.
- Apps and users interpret the data as they see fit



Enter Data Lake Architectures

Data lake is a new and increasingly popular architecture to store and analyze massive volumes and heterogeneous types of data

What is a Data Lake

A

data lake

is the combination of

(1) A centralized repository

that allows you to store

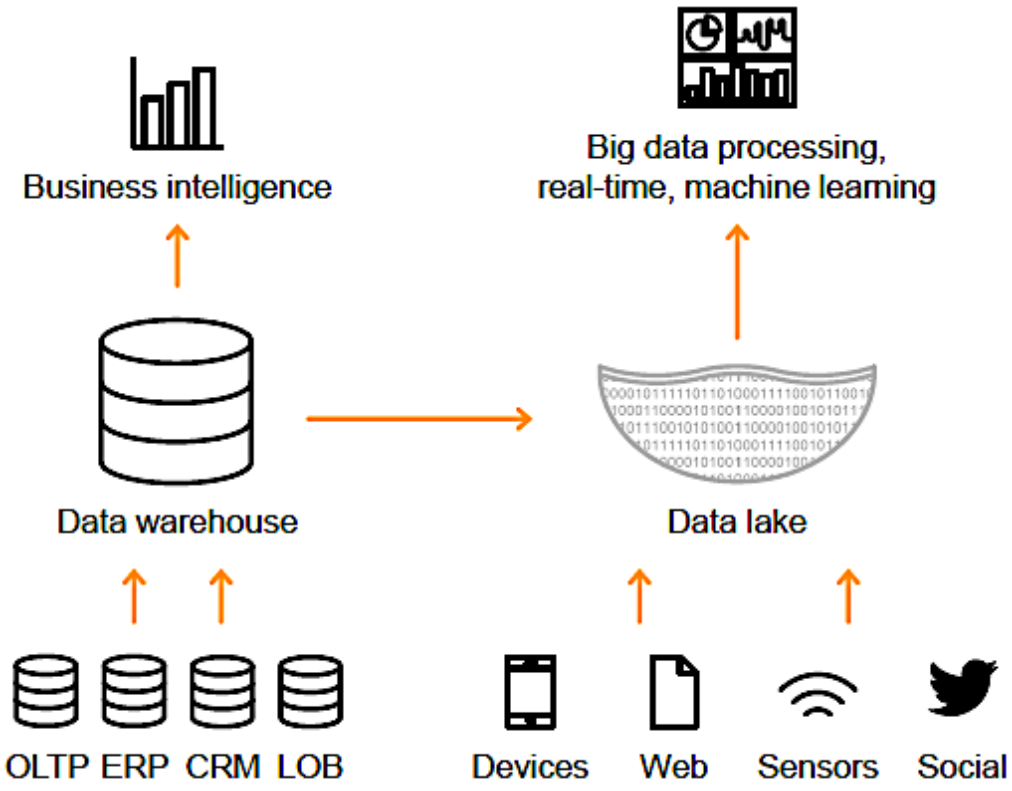
all your structured and unstructured data

at any scale

(2) A data transformation toolkit

able analyze this data

Data Lakes Extend the Traditional Approach



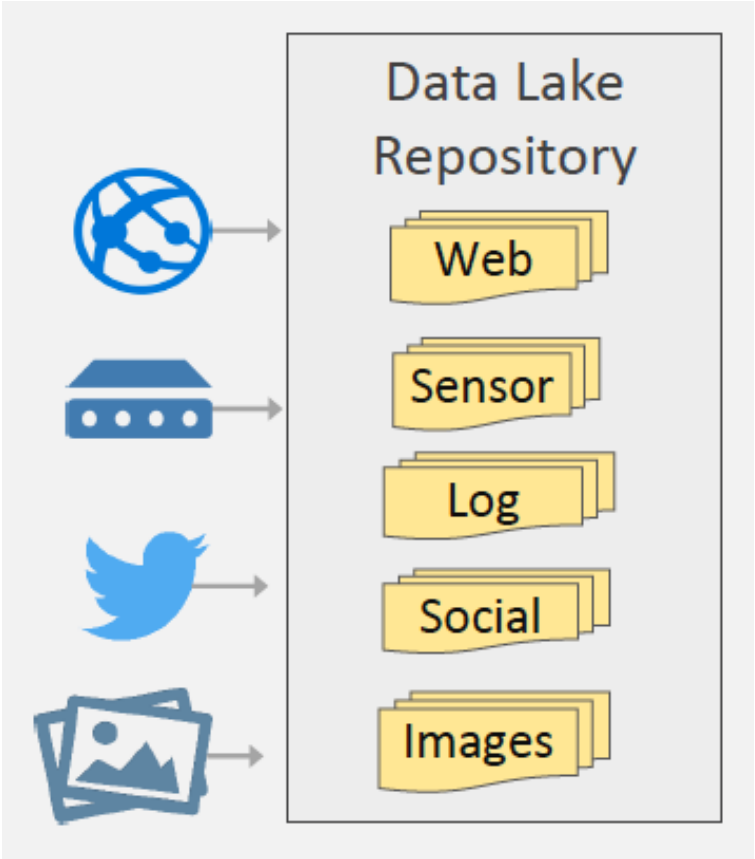
- Relational and nonrelational data
- TBs–EBs scale
- Diverse analytical engines
- Low-cost storage & analytics

The concept of a Data Lake

- All data in one place, a single source of truth
- Handles structured/semi-structured/unstructured/raw data
- Supports fast ingestion and consumption
- Schema on read
- Designed for low-cost storage
- Decouples storage and compute
- Supports protection and security rules

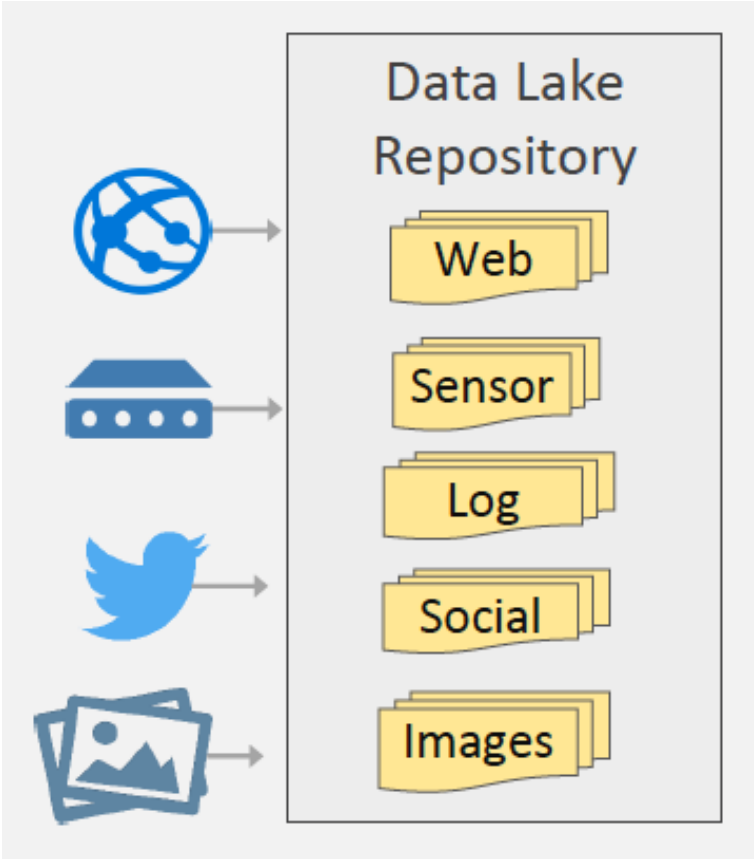
Data Lake

- A repository for analyzing large quantities of disparate sources of data in its native format
- One architectural platform to house all types of data:
  - Machine-generated data (ex: IoT, logs)
  - Human-generated data (ex: tweets, e-mail)
  - Traditional operational data (ex: sales, inventory)



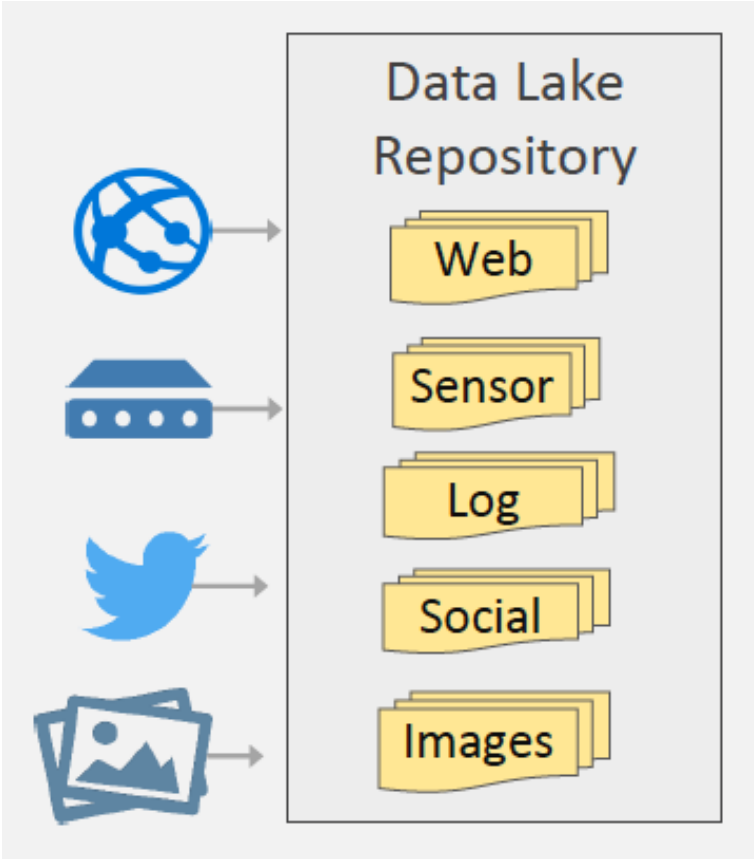
Objectives of a Data Lake

- Reduce up-front effort by ingesting data in any format without requiring a schema initially
- Make acquiring new data easy, so it can be available for data science & analysis quickly
- Store large volume of multi-structured data in its native format



Objectives of a Data Lake

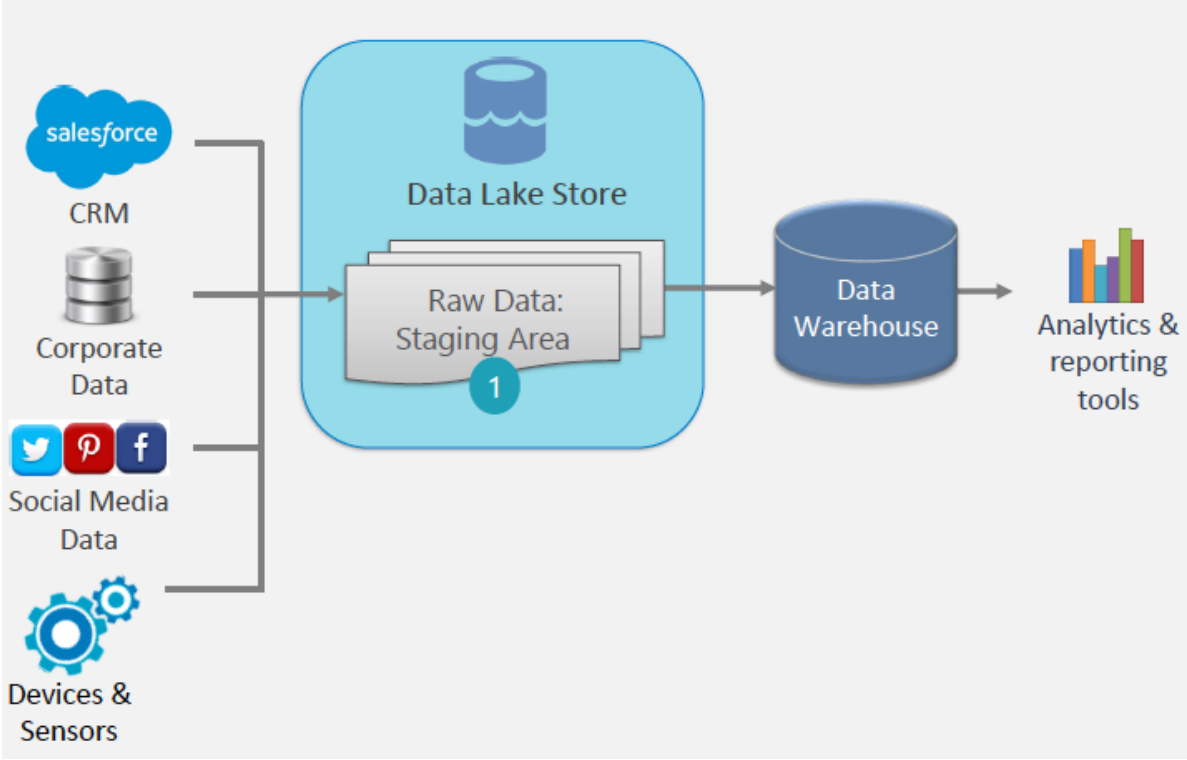
- Defer work to apply schemas to data until after requirements are known
- Achieve agility faster than a traditional data warehouse
- Speed up decision-making ability
- Storage for additional types of data which were historically difficult to obtain



Data Lake as a Staging Area for DW

- Strategy:
  - Reduce storage needs in data warehouse
  - Practical use for data stored in the data lake

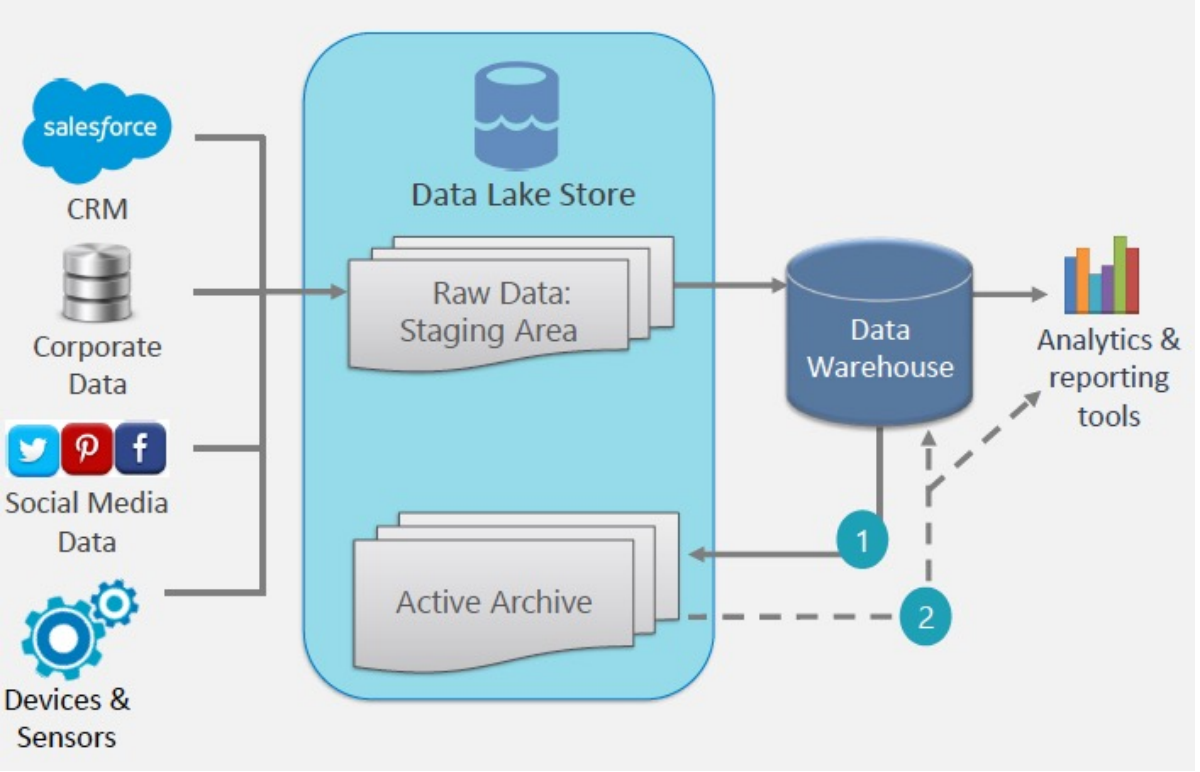
1. Utilize the data lake as a landing area for DW staging area, instead of the relational database



Data Lake for Active Archiving

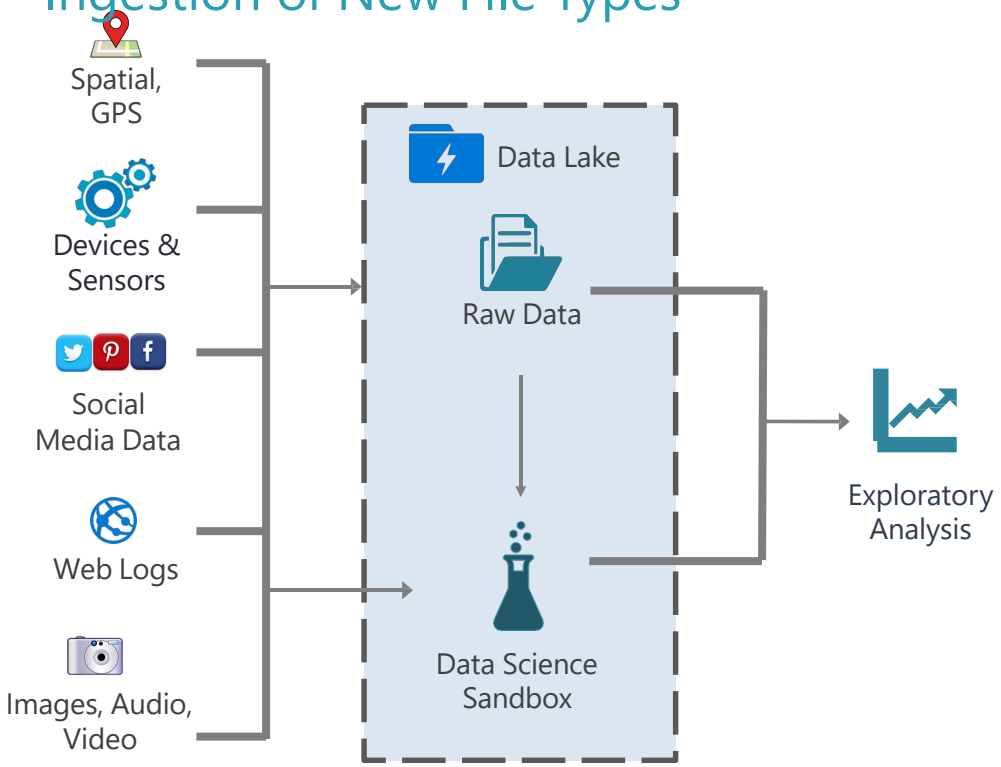
- Strategy:
  - Data archival, with query ability available when needed

1. Archival process based on data retention policy
2. Federated query to access current & historical data



Data Lake Use Cases

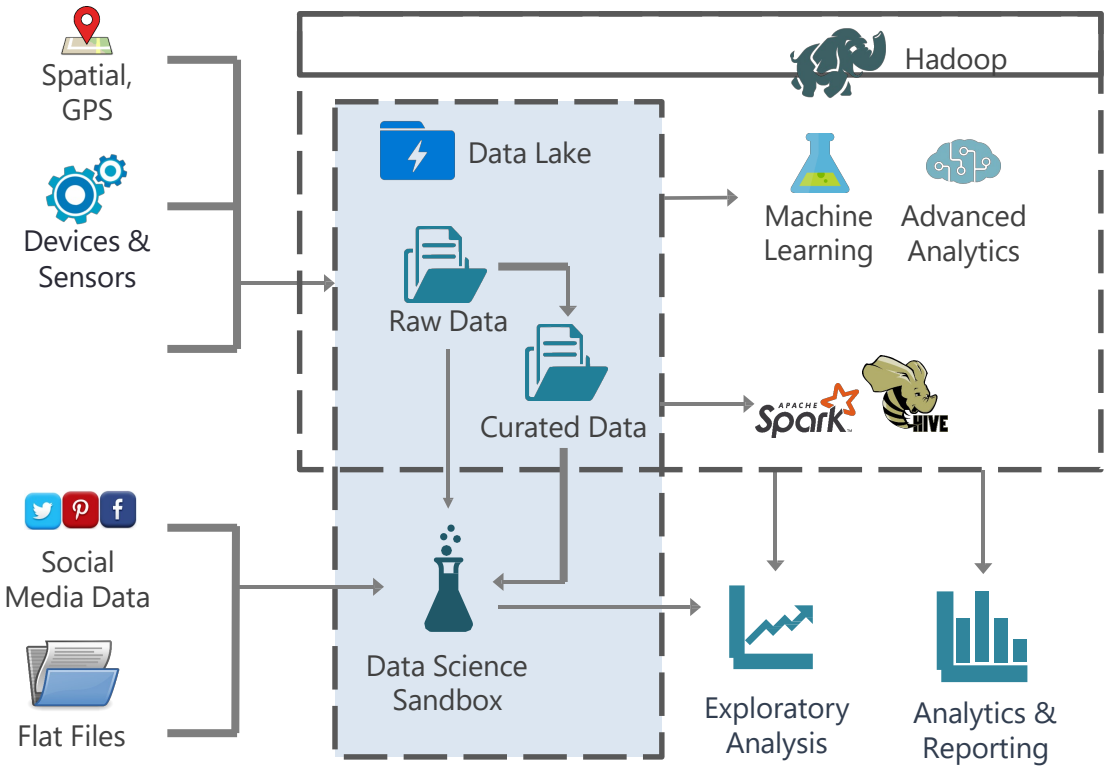
Ingestion of New File Types



- ✓ Preparatory file storage for multi-structured data
- ✓ Exploratory analysis to determine value of new data types & sources
- ✓ Affords additional time for longer-term planning while accumulating data or handling an influx of data

Data Lake Use Cases

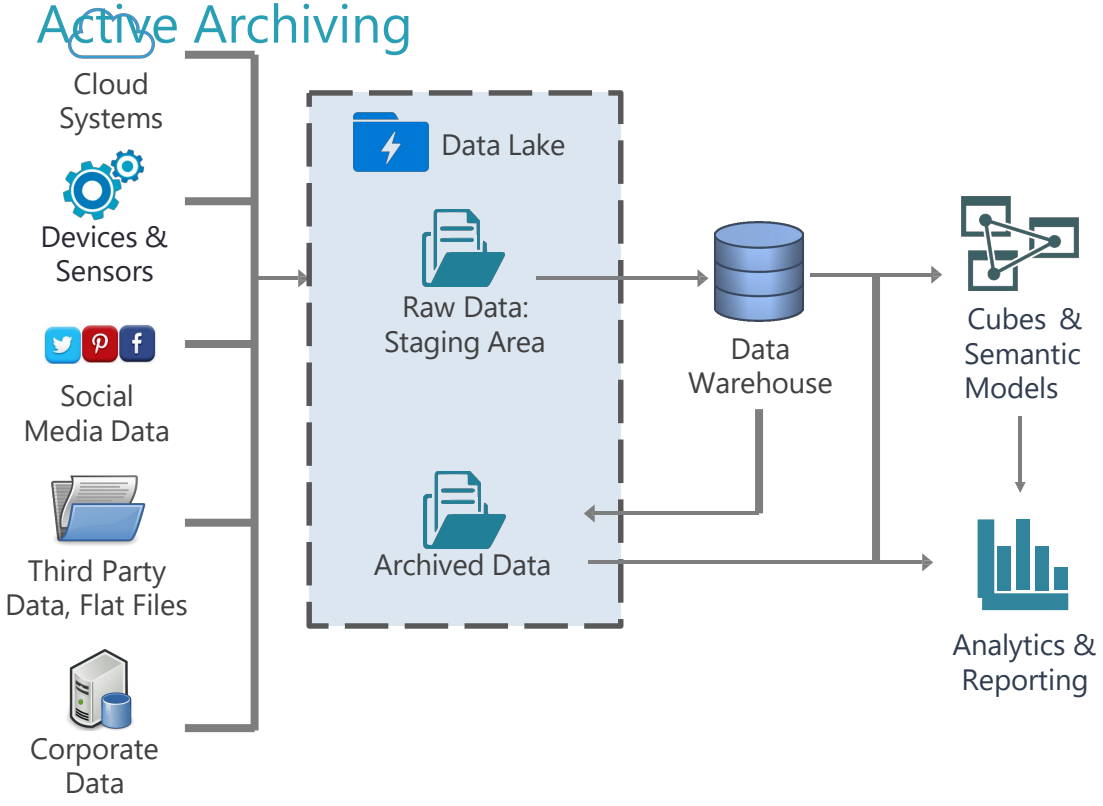
Data Science Experimentation | Hadoop Integration



- ✓ Big data clusters
- ✓ SQL-on-Hadoop solutions
- ✓ Integrate with open source projects such as Hive, Spark, Storm, Kafka, etc.
- ✓ Sandbox solutions for initial data prep, experimentation, and analysis
- ✓ Migrate from proof of concept to operationalized solution

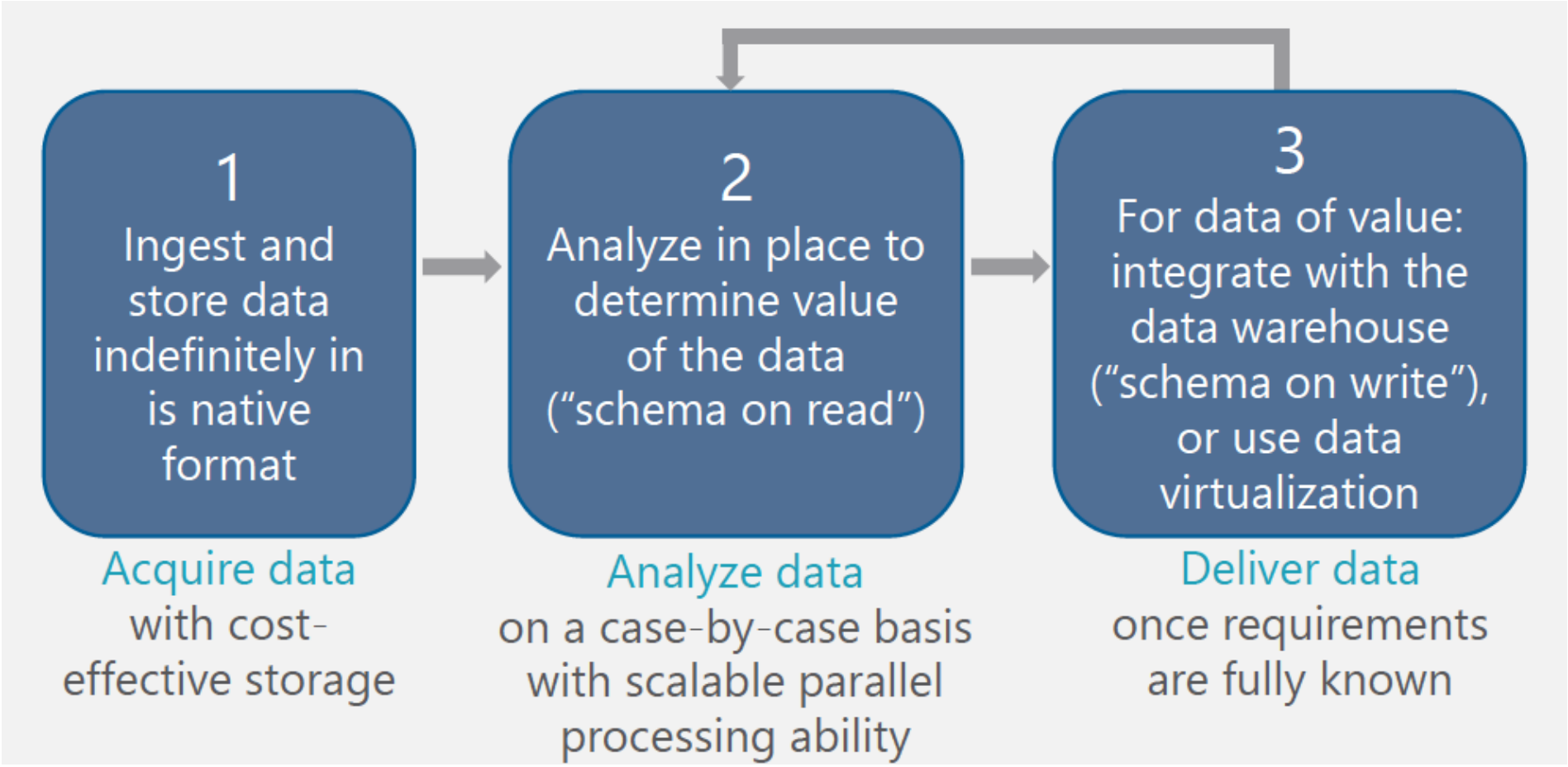
Data Lake Use Cases

Active Archiving



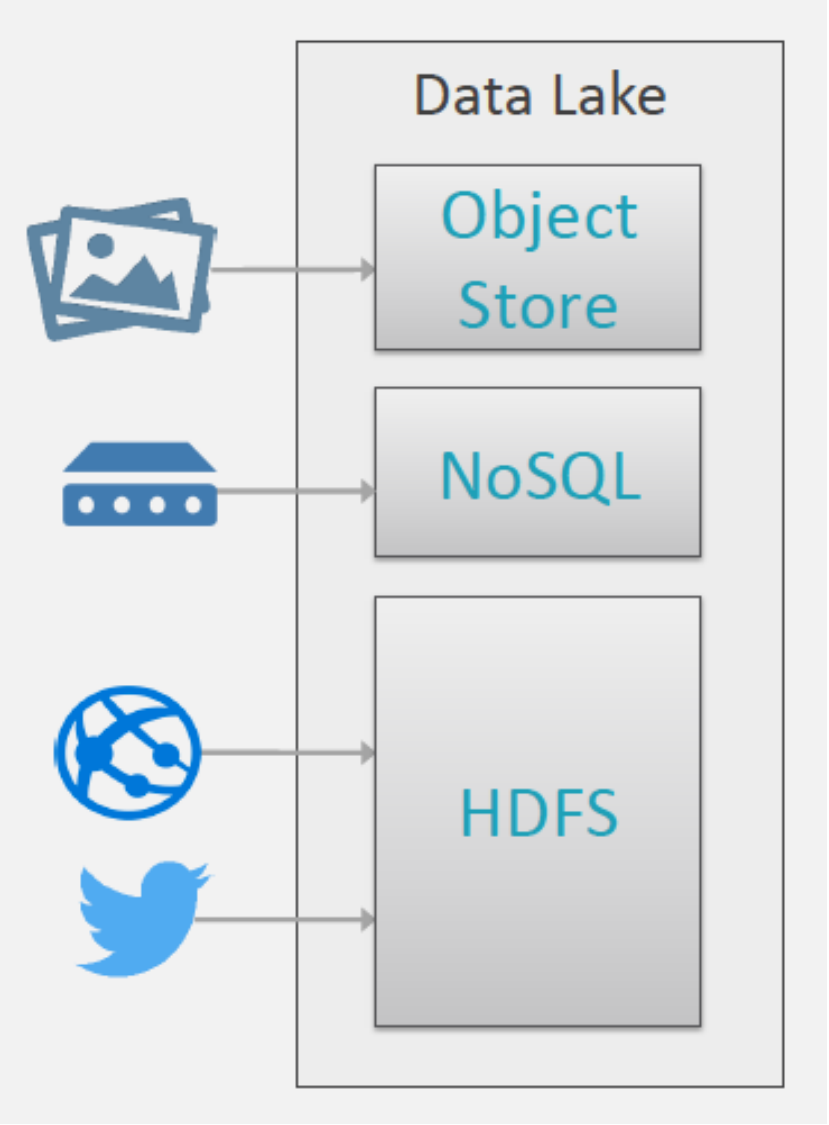
- ✓ Offload aged data from data warehouse back to the data lake
- ✓ An "active archive" available for querying when needed
- ✓ Federated queries to access:
  current data in the DW + archive data in the data lake

Iterative Data Lake Pattern



Data Lake Implementation

- A data lake is a conceptual idea. It can be implemented with **one or more** technologies.
  - **HDFS**(Hadoop Distributed File Storage) is a very common option for data lake storage. However, Hadoop is not a requirement for a data lake. A data lake may also span > 1 Hadoop cluster.
  - **NoSQL** databases are also very common.
  - **Object stores** (like Amazon S3 or Azure Blob Storage) can also be used.



Benefits of a Data Lake—All Data is in One Place

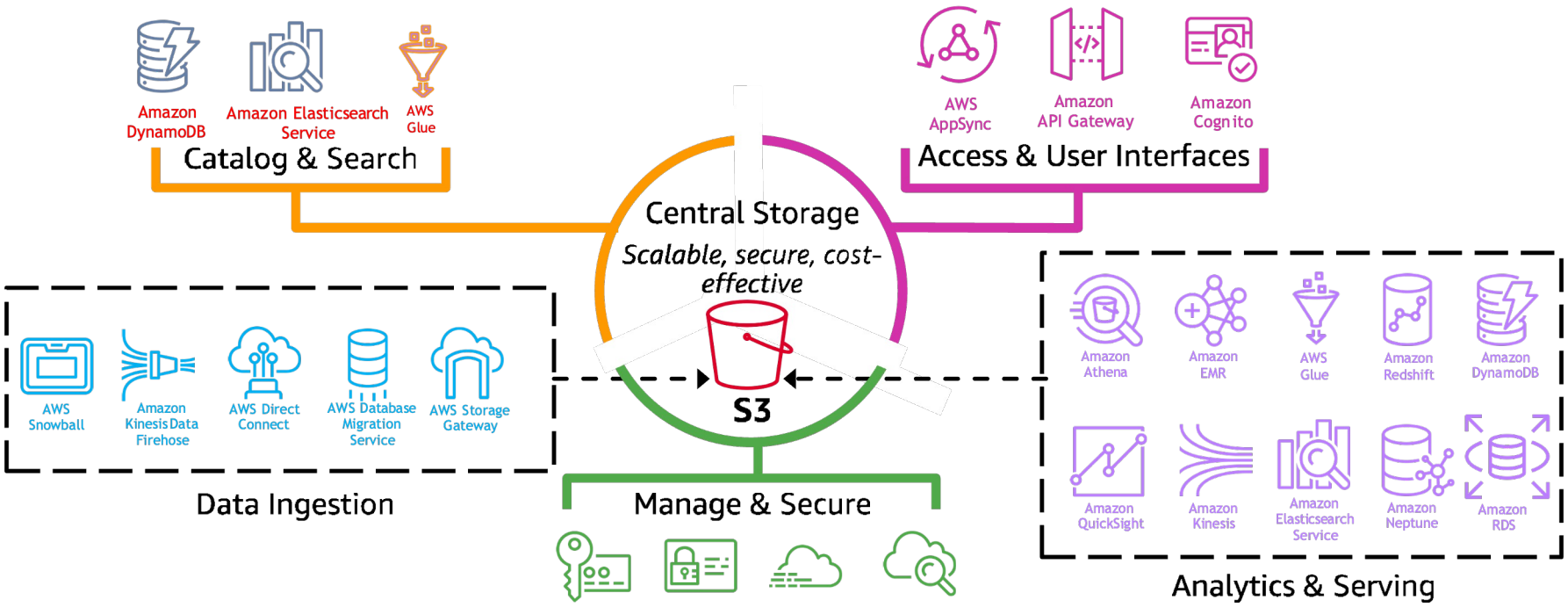
"Why is the data distributed in many locations? Where is the single source of truth?"

Analyze all of your data, from all of your sources, in one stored location

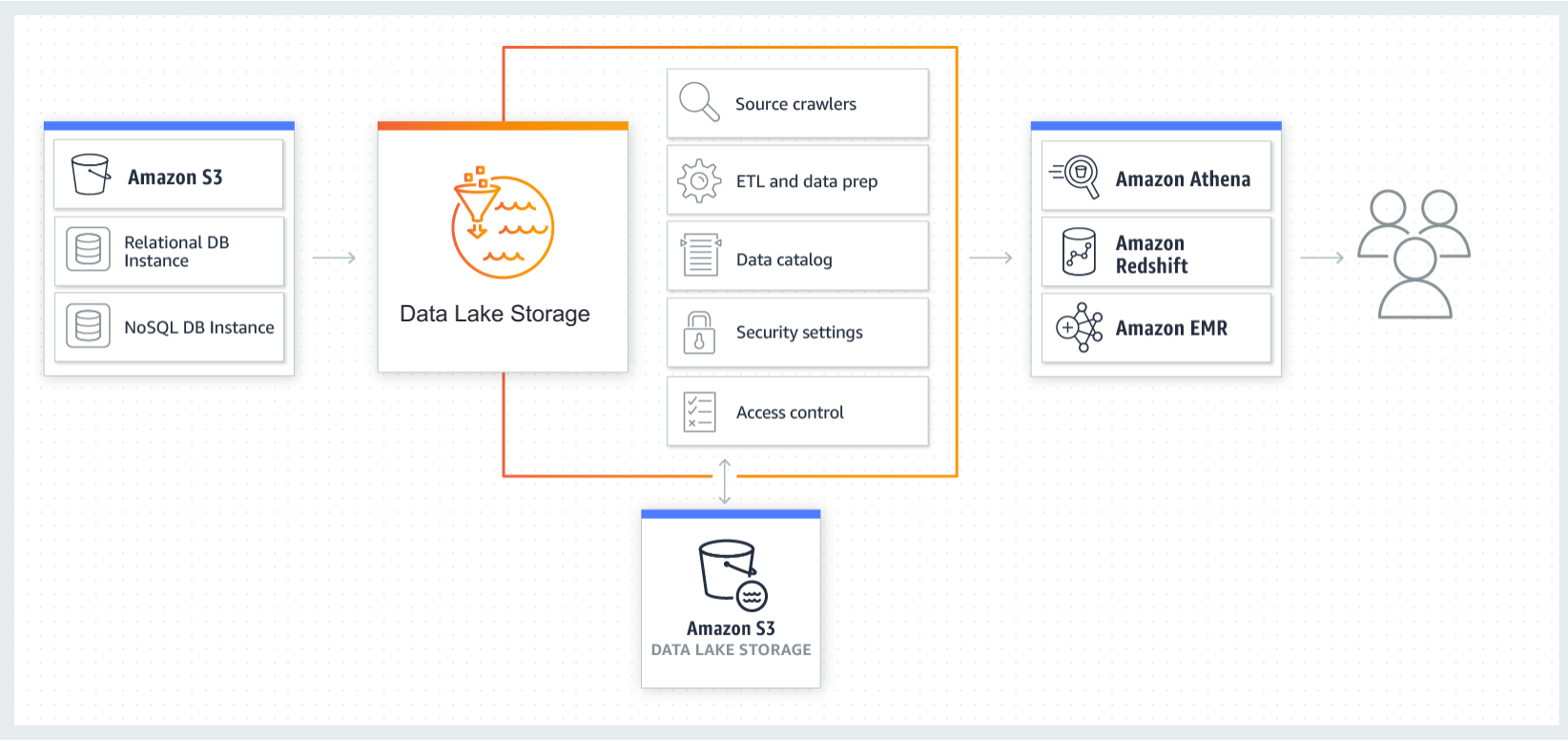
Data lakes are central to the modern data architecture

- Ingest all types of data in any format
- Create Refined, Standardized, Trusted datasets for various use cases
- Store data for longer periods of time to enable historical analysis
- Query and access data using a variety of methods
- Manage streaming and batch data in a converged platform
- Provide shorter time-to-insight with proper management and governance

Data lake on AWS



AWS Data Lake Architecture



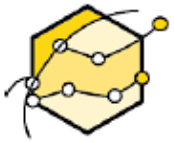
Why Amazon S3 for a Data Lake?

Durable

Designed for 11 9s of durability

Available

Designed for **99.99%** availability



High performance

- Multiple upload
- Range GET
- Scalable throughput

Easy to use

- Simple REST API
- AWS SDKs
- Simple management tools
- Event notification
- Lifecycle policies

Scalable

- Store as much as you need
- Scale storage and compute independently
- No minimum usage commitments

Integrated

- Amazon EMR
- Amazon Redshift Spectrum
- Amazon DynamoDB
- Amazon Athena
- AWS Glue

Benefits of an AWS Amazon S3 Data Lake

Fixed cluster data lake

- Limited to only the single tool contained on the cluster (for example, Hadoop or data warehouse or Cassandra). Use cases and ecosystem tools change rapidly.
- Expensive to add nodes to add storage capacity
- Expensive to replicate data against node loss
- Complexity in scaling local storage capacity
- Long refresh cycles to add additional storage equipment

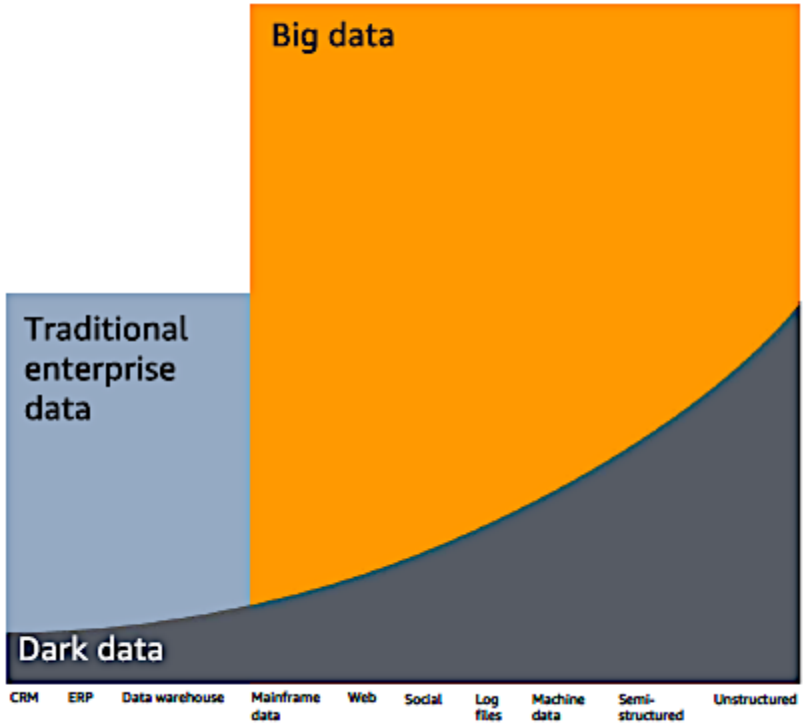
AWS Amazon S3 data lake

- Decouple storage and compute by making S3 object based storage, not a fixed tool to manage the data lake
- Flexibility to use any and all tools in the ecosystem. ***The right tool for the job.***
- Catalog, transform, and query in place
- Future-proof your architecture. As new use cases and tools emerge you can plug and play current best of breed.

Storing is Not Enough…
Data Needs to Be Discoverable

"Dark data are the information assets organizations collect, process, and store during regular business activities, but generally fail to use for other purposes (for example, analytics, business relationships and direct monetizing)."

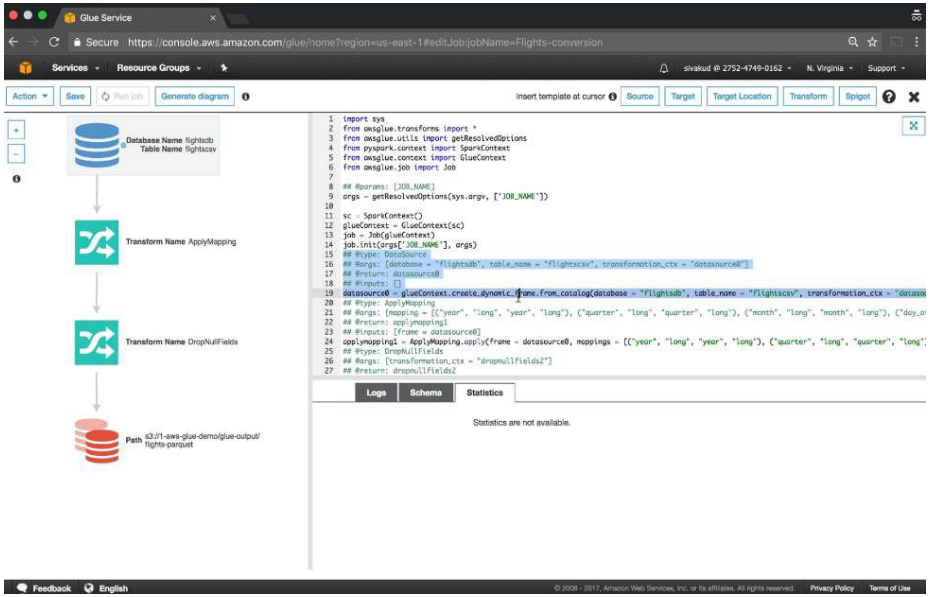
Gartner IT Glossary, 2018
https://www.gartner.com/it-glossary/dark-data



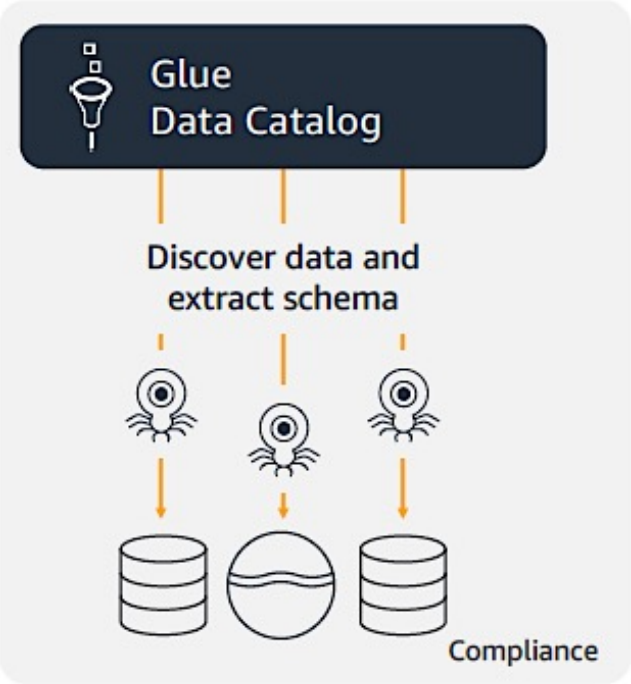
AWS Glue—ETL Service

Make data transformation scripting easy

- Automatically generates Spark (ETL) code
- Scala or Python shell script
- Code is customizable
- Endpoints provided to edit, debug, test code
- Jobs are scheduled or event-based



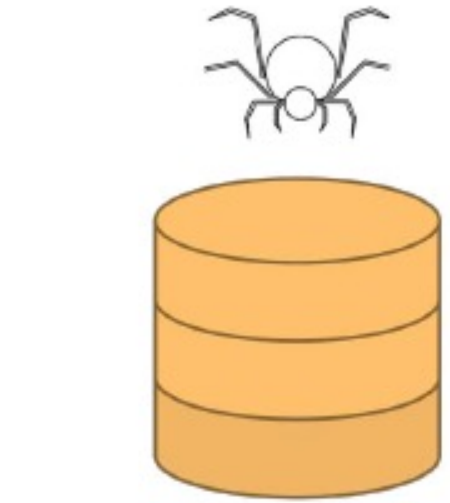
AWS Glue—Data Catalog
Make data discoverable



- Automatically discovers data and stores schema
- Catalog makes data searchable, and available for ETL
- Catalog contains table and job definitions
- Computes statistics to make queries efficient

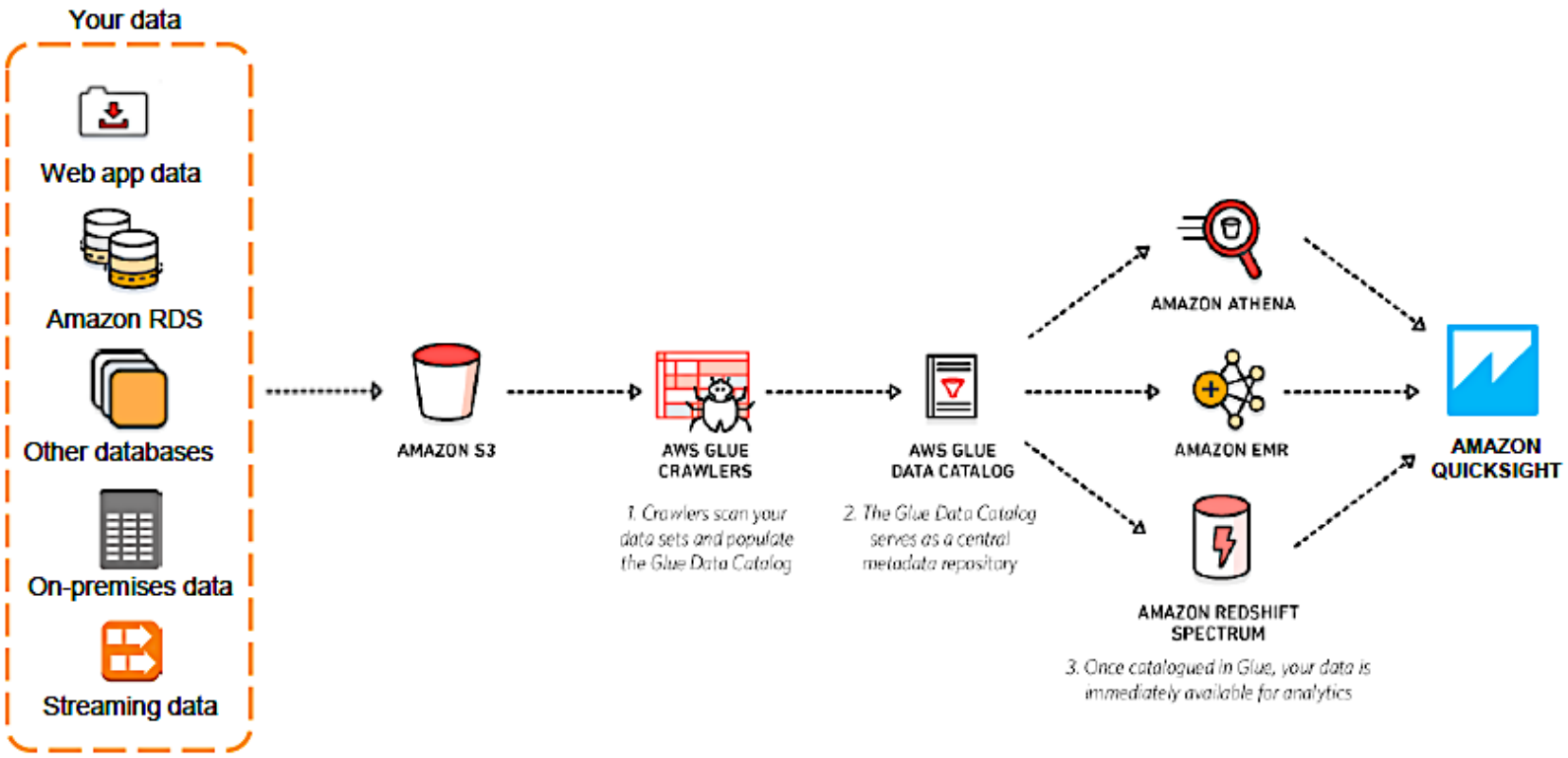
AWS Glue Crawlers

- Crawlers automatically build your Data Catalog and keep it in sync.
- Automatically discover new data, extracts schema definitions
- Detect schema changes and version tables
- Detect Hive style partitions on Amazon S3
- Built-in classifiers for popular format types
- Run ad hoc or on a schedule

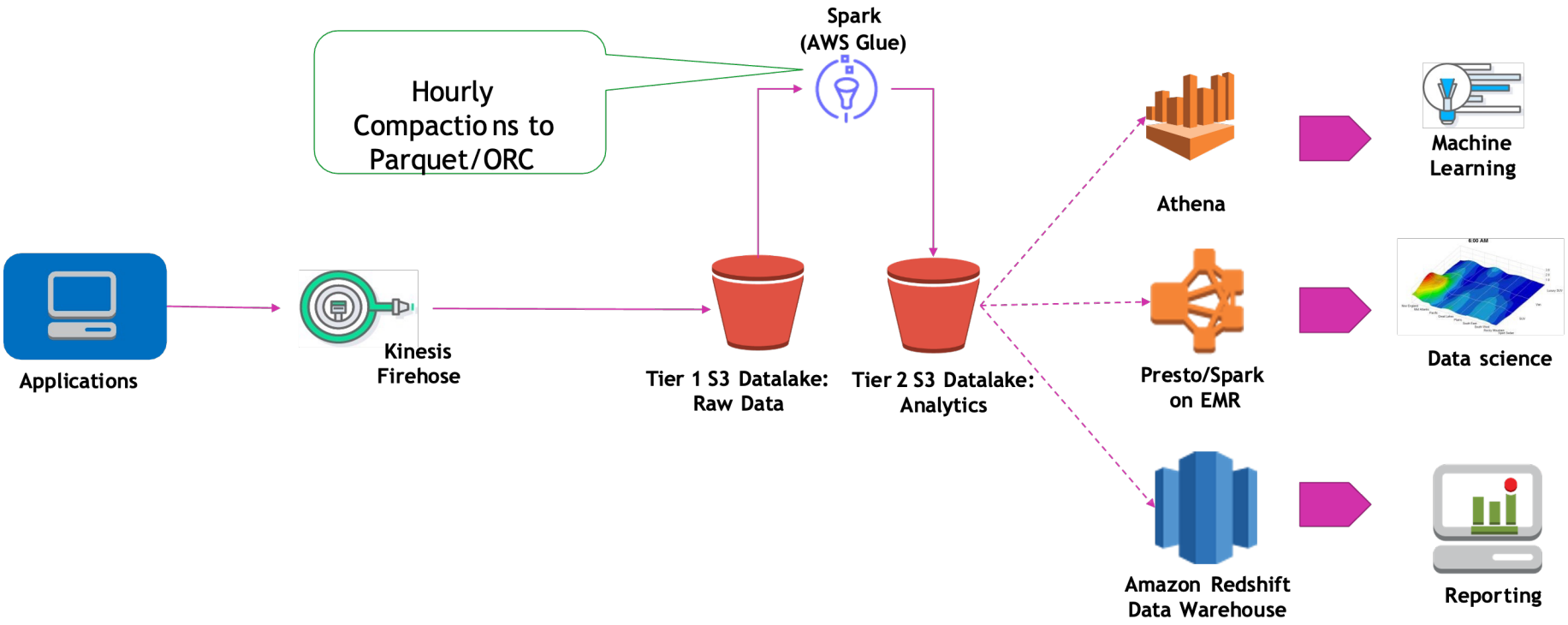


Crawlers
Automatically catalog your data

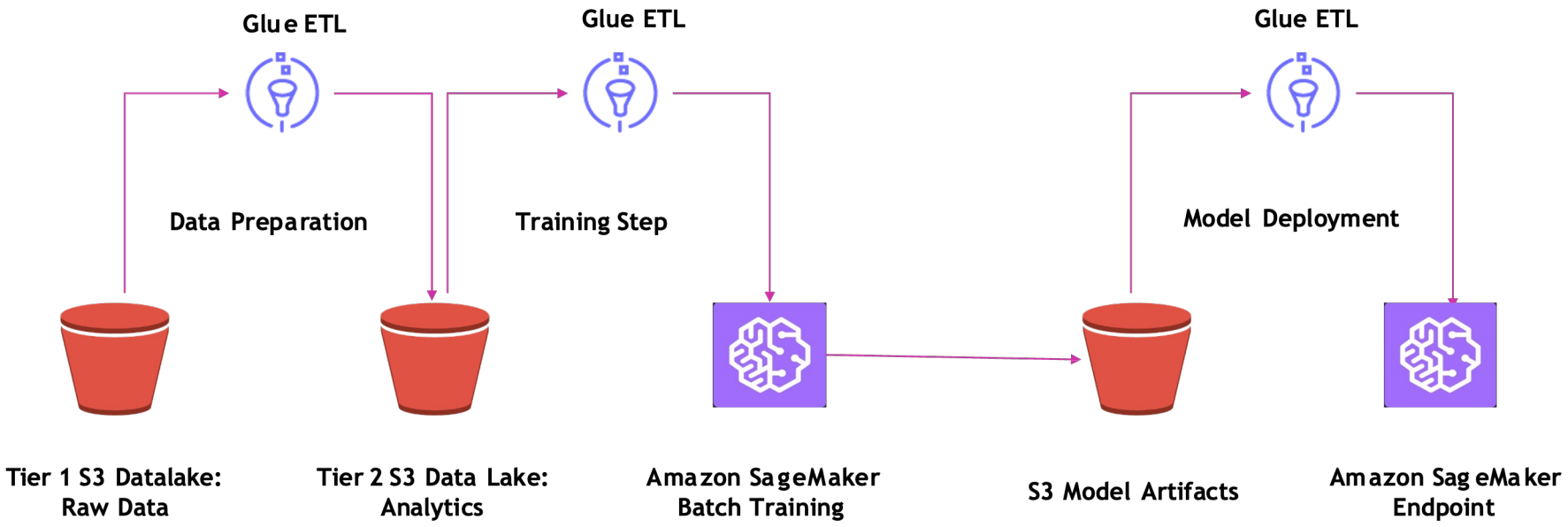
Data Lake on Amazon S3 with AWS Glue



Log analytics, ClickStream analytics, IoT sensor data



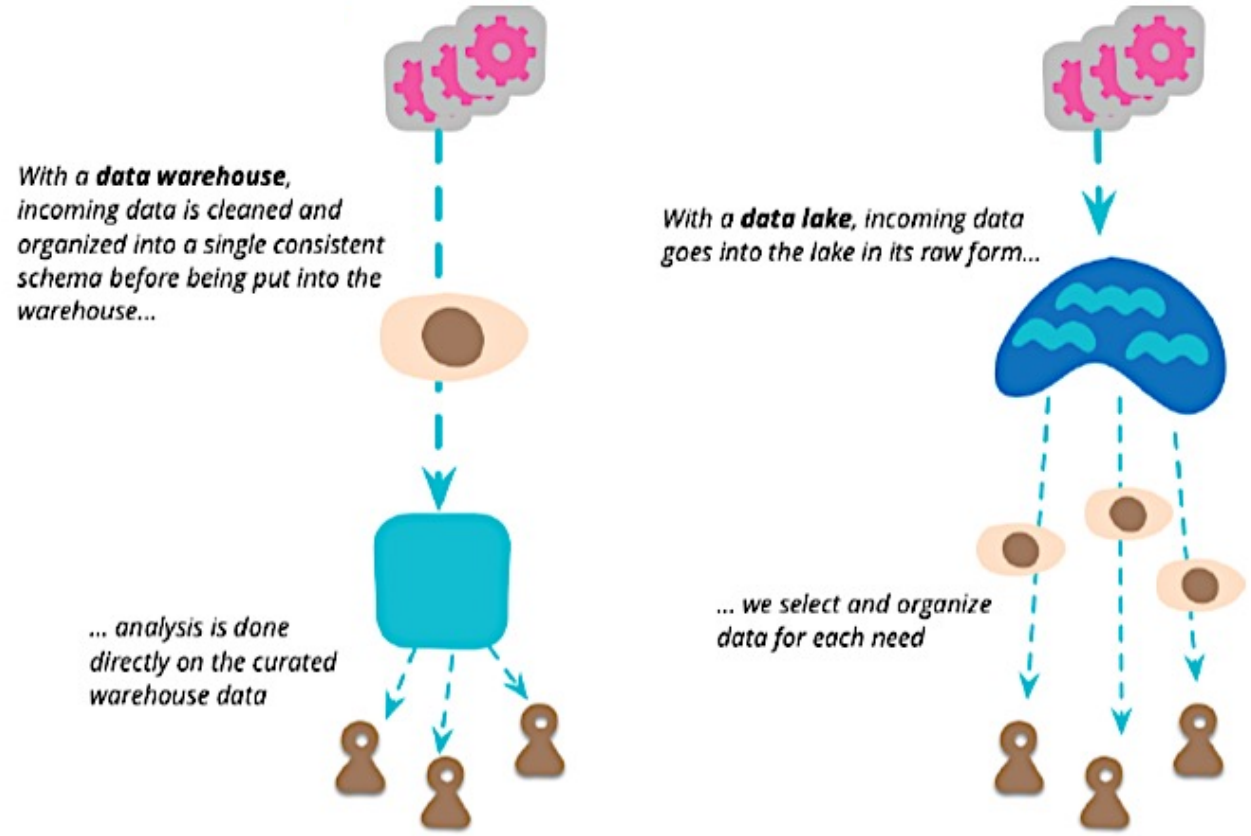
Machine Learning—Batch training pipeline



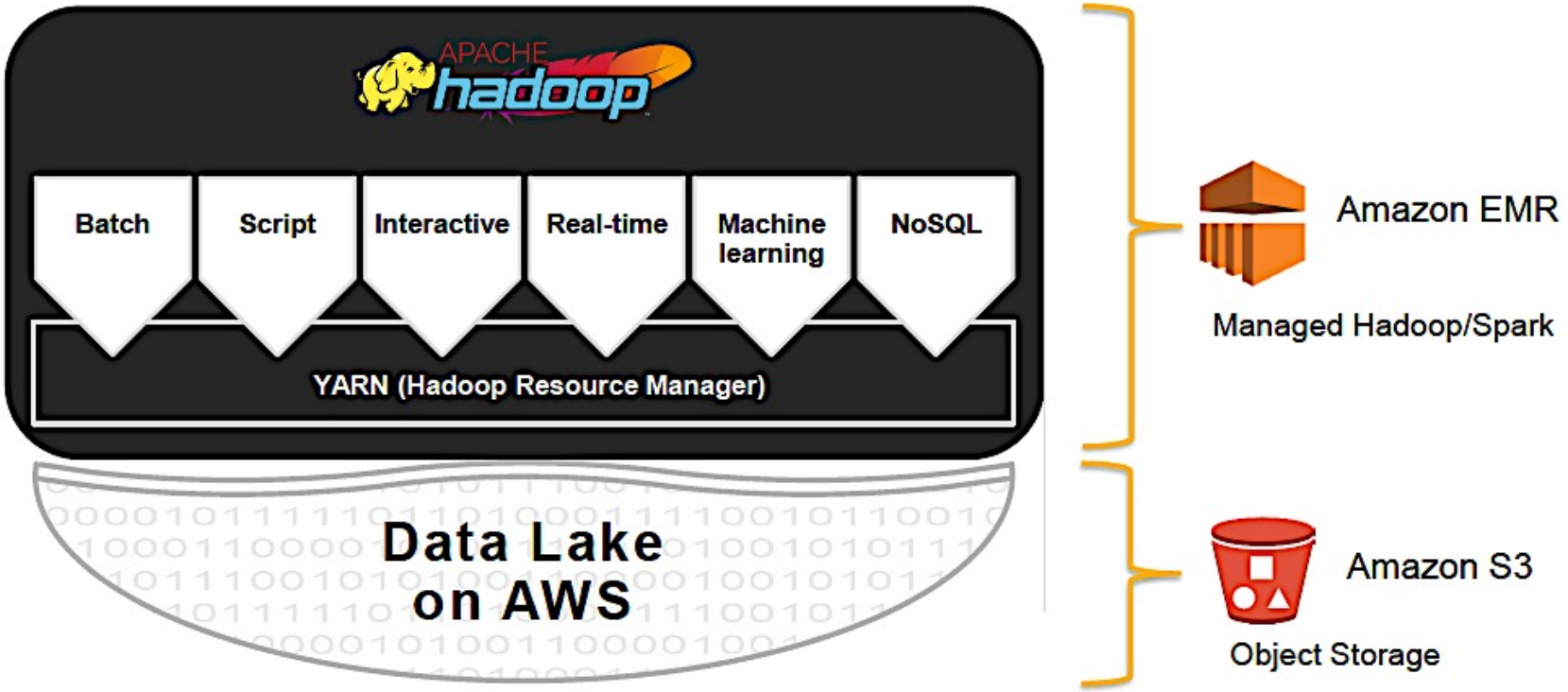
The Hadoop Data Lake Concept

- Important concept: “You don’t have to know what you’re going to do with your data before you store it”
- Data of many types, sizes, shapes and structures can all be thrown into the Hadoop Distributed File System, and other Hadoop data storage systems
- While some metadata needs to be stored, to know what’s in there, no need yet to know how it will be structured!
- Therefore, the data is stored in its original granular form, with nothing thrown away
- In fact, no structural information is defined at all when the data is stored
- So “schema on read” implies that the schema is defined at the time the data is read and used, not at the time that it is written and stored
- When someone is ready to use that data, then, at that time, they define what pieces are essential to their purpose, where to find those pieces of information that matter for that purpose, and which pieces of the data set to ignore
- The Hadoop data lake concept can be summed up as, “Store it all in HDFS, figure out what to do with it later”

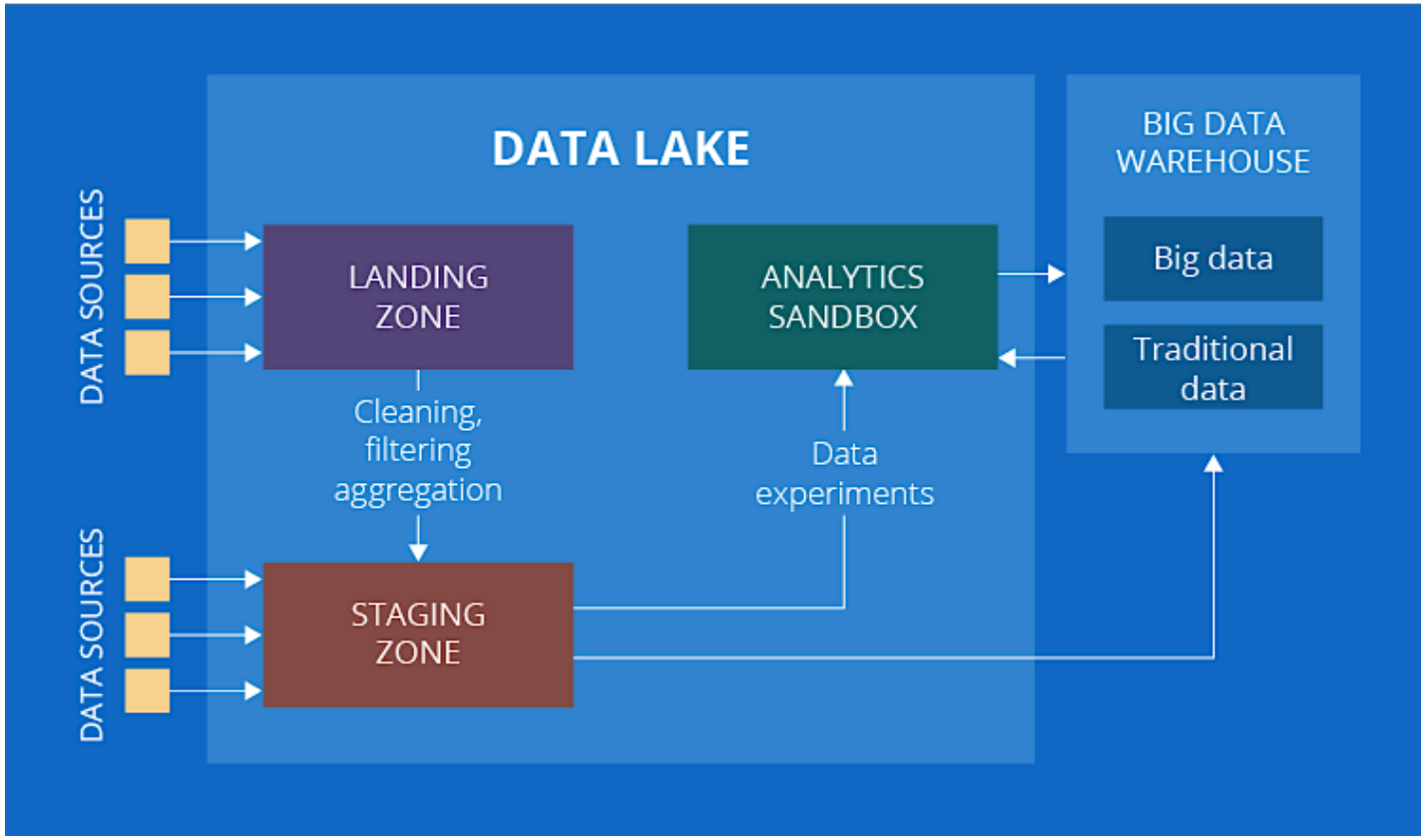
Hadoop Data Lake vs. Enterprise Data Warehouse



Data Lake with AWS EMR (or AWS Glue)



Data Lake Zones

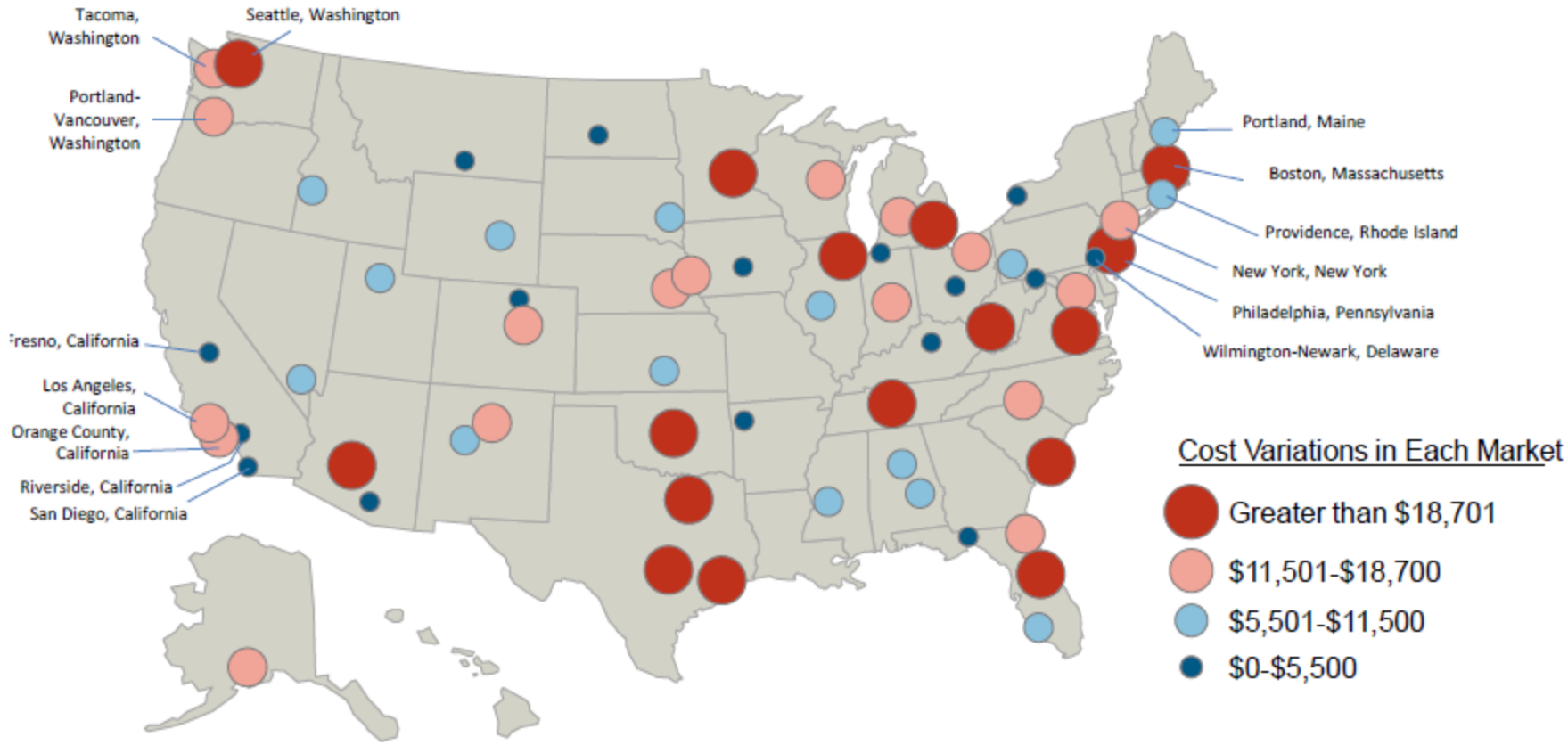


The Lambda Architecture Pattern

Big Data and Healthcare

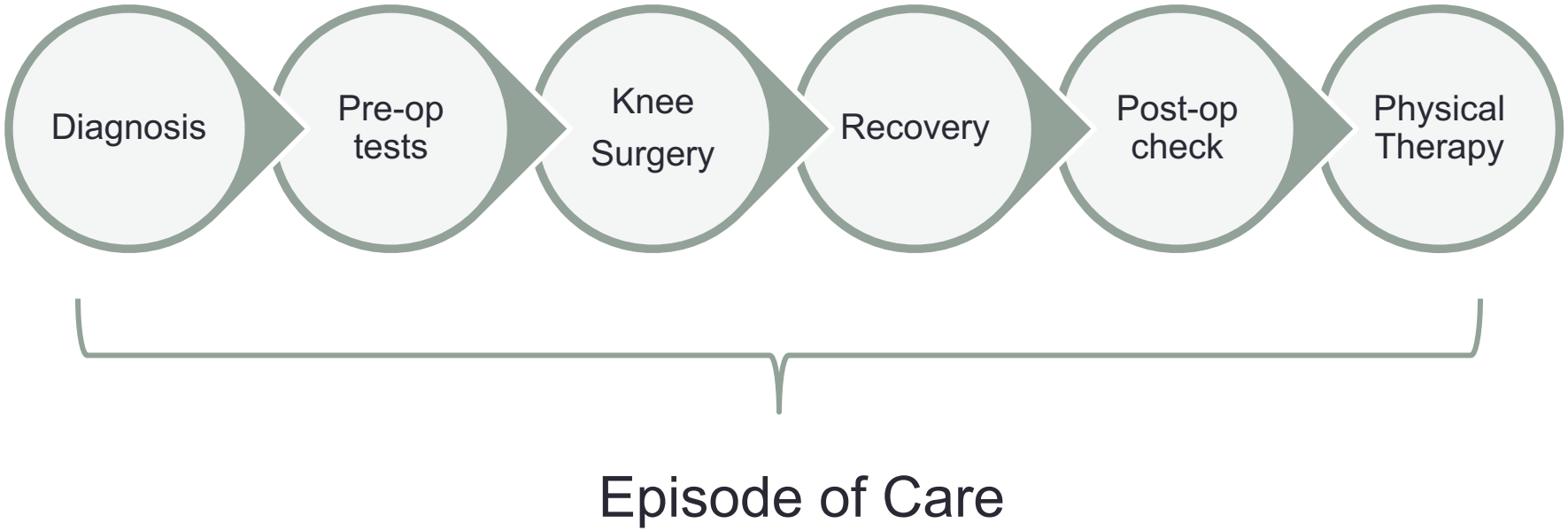
The Health of America Report is a collaboration between the Blue Cross Blue Shield Association and Blue Health Intelligence that uses advanced analytics to report on key trends in healthcare to support improved quality and affordability for all Americans

Cost Variation for Knee Replacement Procedures Across the Country



http://www.bcbs.com/healthofamerica/BCBS_BHI_Report-Jan-_21_Final.pdf

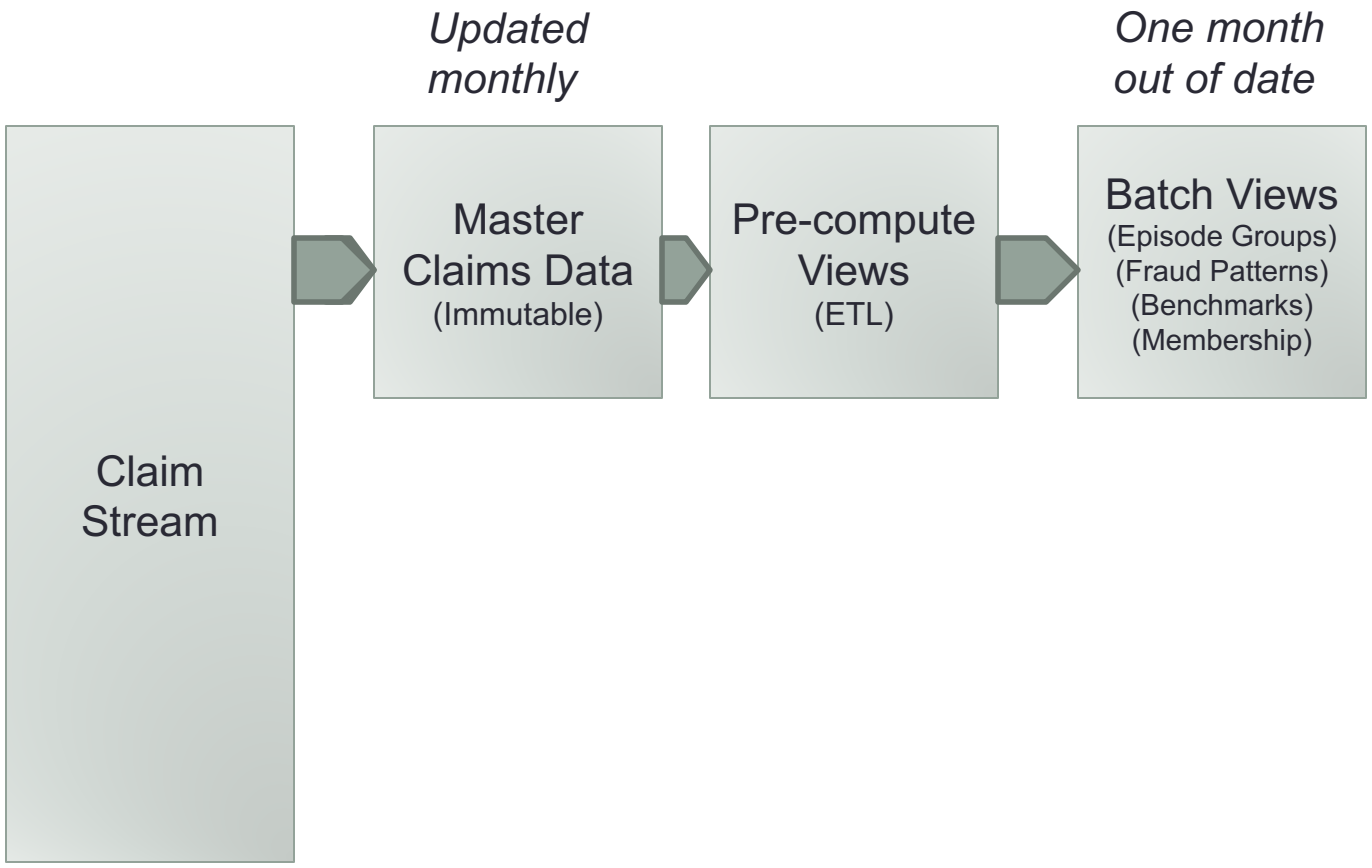
But What is a Knee Replacement?



Calculate Knee Replacement Episode of Care

- Collect a time ordered sequence of medical and pharmacy claims from across the country
  - Hundreds of millions of claims with billions of claim lines
- Use a heuristic algorithm to group episodes
  - Filter claims to look for knee surgery charges
  - Look earlier and later to capture other aspects of the episode
  - Ignore irrelevant claim costs such as vaccinations, comorbidities (diabetes, heart disease) and so on
- Apply complex and evolving rules to better determine the scope of an episode and relevant charges
  - Improved rules are released each year by vendors
- Episode of care determination is a batch operation over a large volume of historical claims
  - Executing each year's new episode grouping software on historical data may improve episode identification

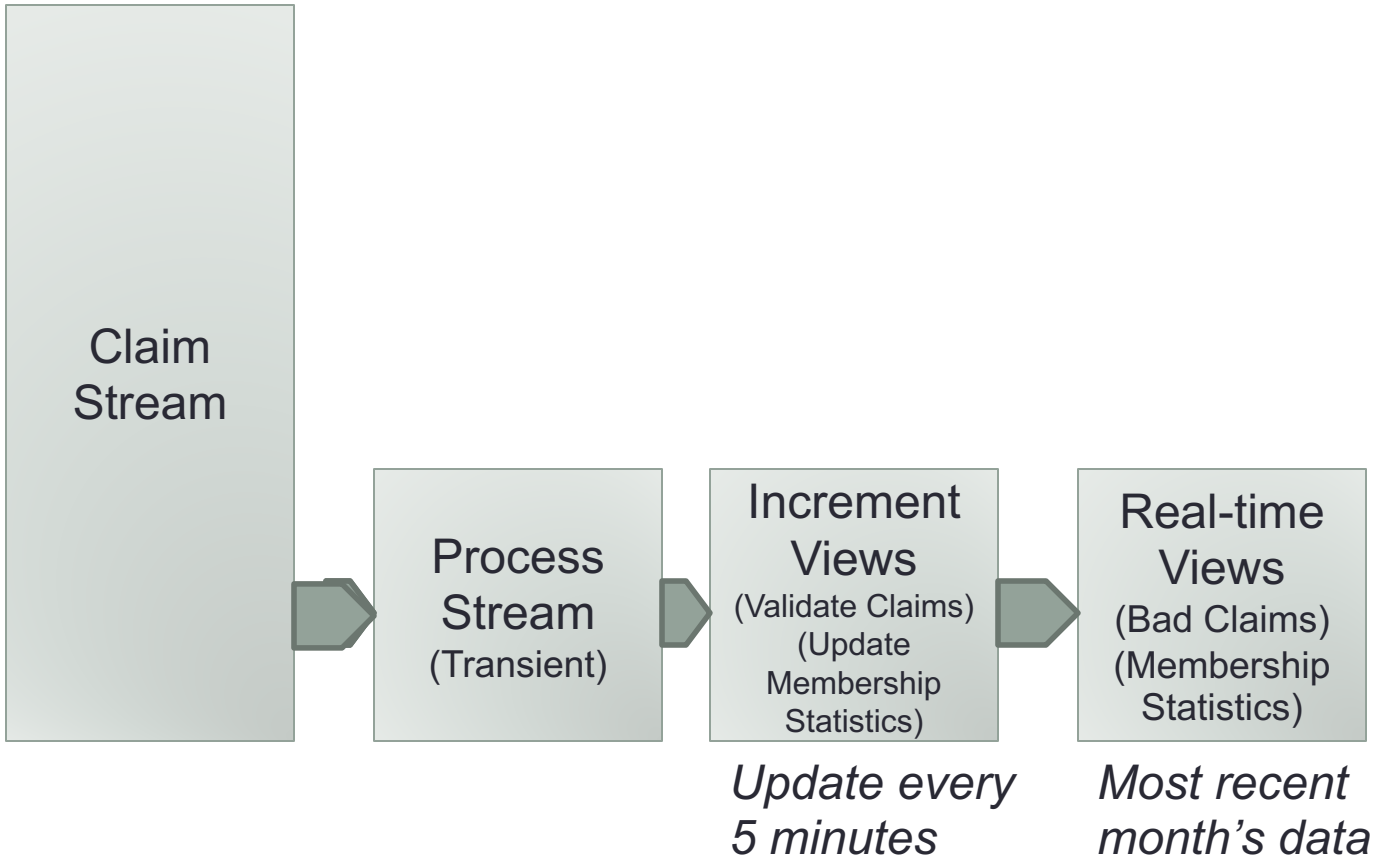
Grouping Episodes



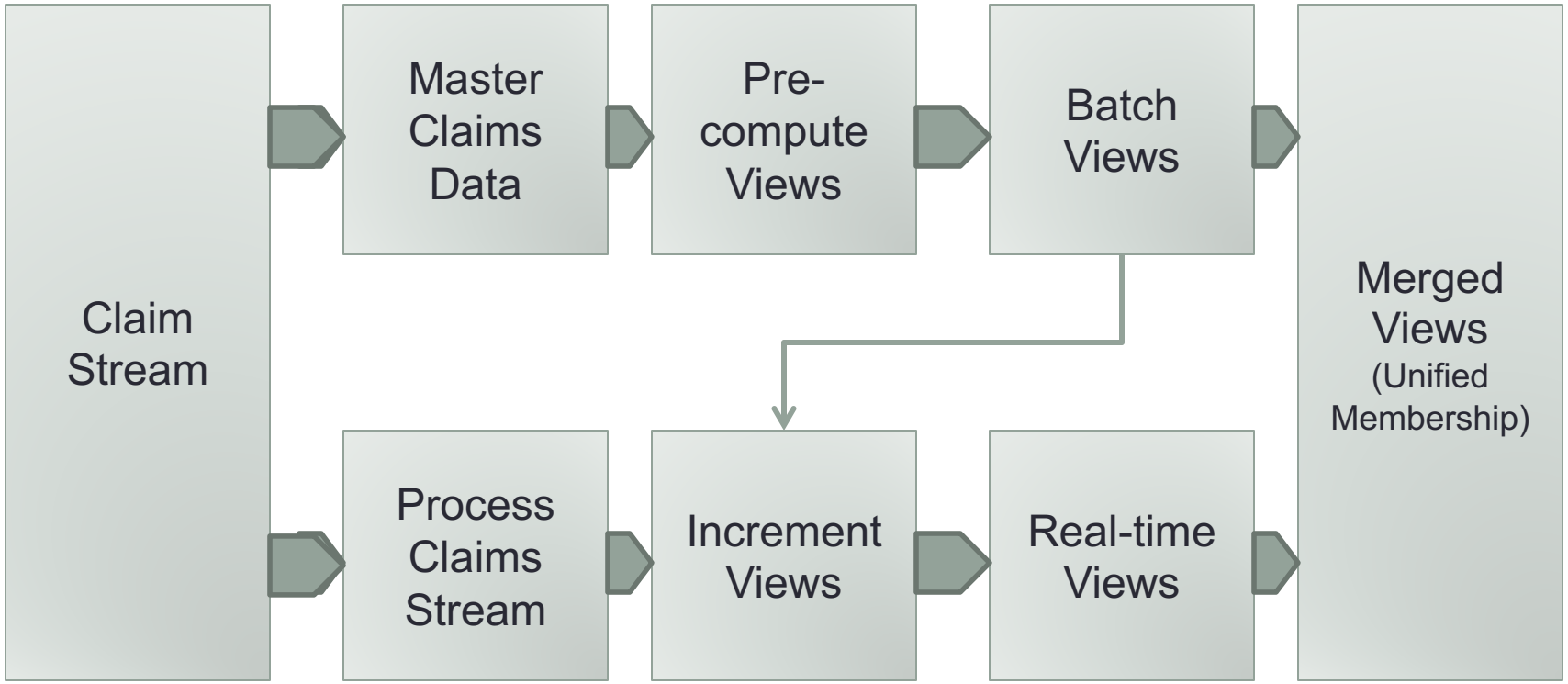
Fraud and Error Detection

- Some claims are invalid due to entry errors
  - Claim total charge does not equal sum of claim line charges
  - A required information field is empty or out of range
- Some claims are fraudulent attempts to receive payment for services
  - Claims are evaluated for internal consistency
  - Claims are checked against historical patterns of fraud
  - We want to pay claims promptly but not pay bad actors
- Fraud and error detection is a near real-time operation over individual claims
  - Requires evaluating millions of arriving claims per day
  - Sometimes also using summarized historical information

Fraud and Error Detection



Unified Claims Handling Batch and Real-Time

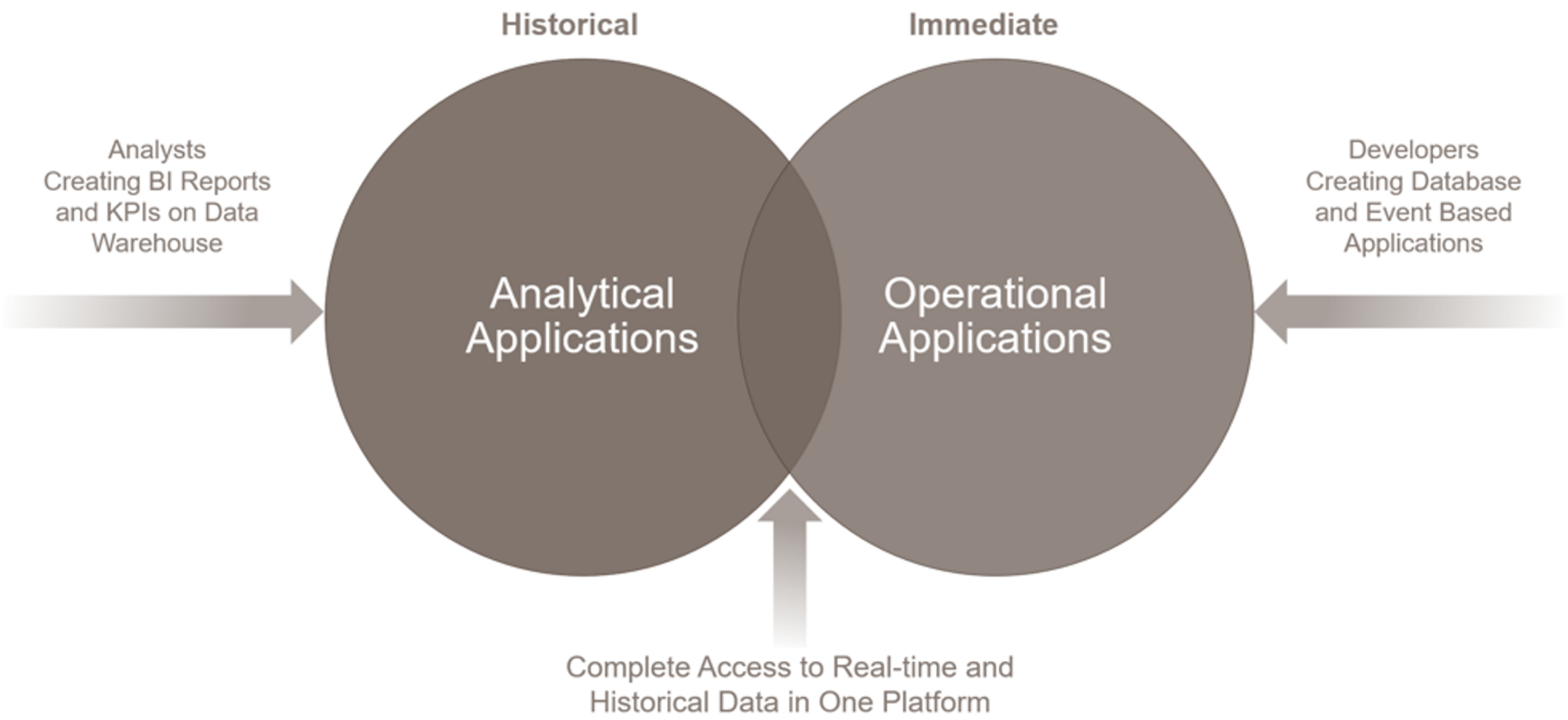


Lambda Architecture

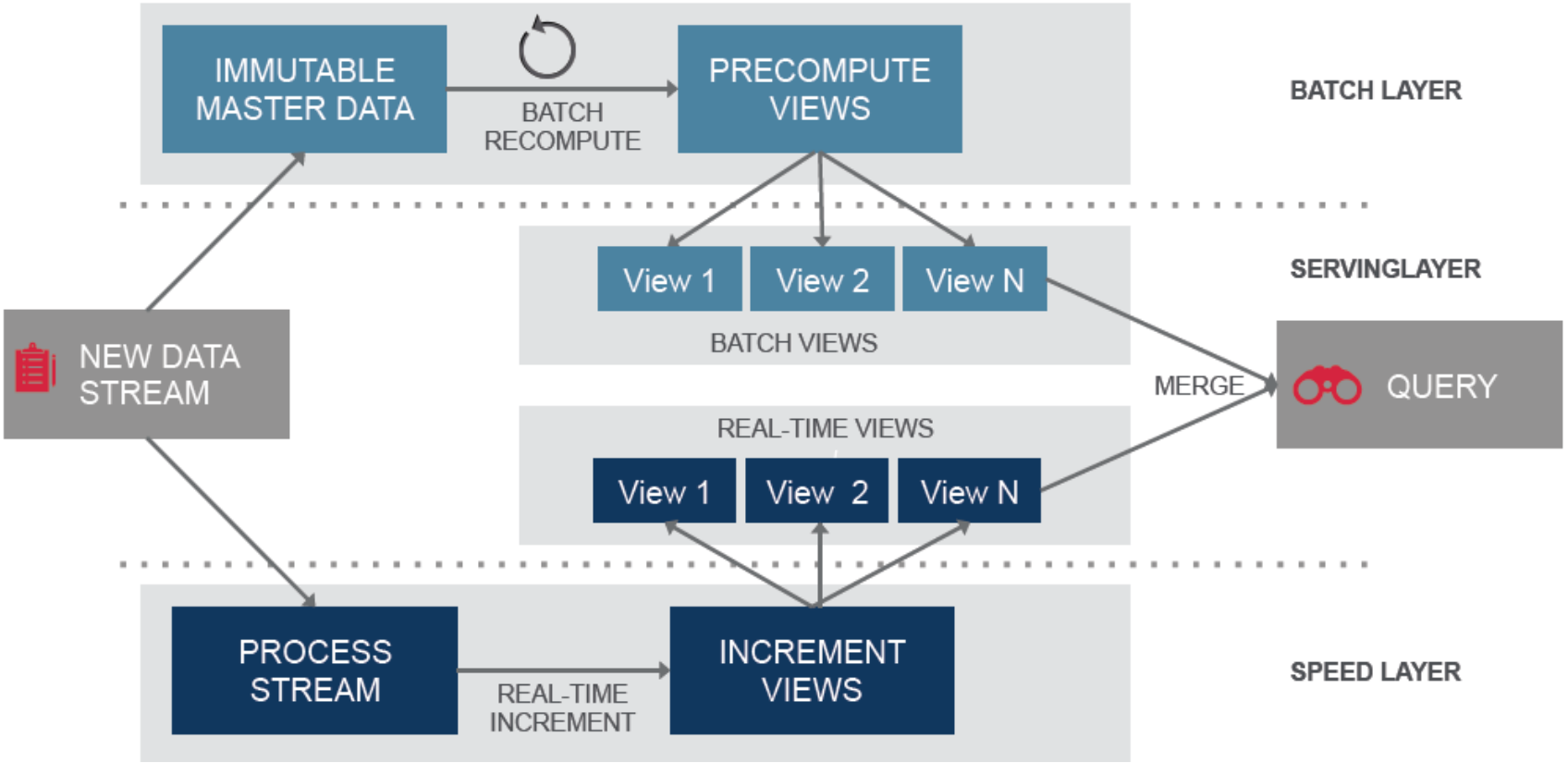
A data-processing architecture designed to handle massive quantities of data by taking advantage of both batch and stream (real-time) subsystems

AKA Converged Architecture

Converging Analytical and Operational Processing

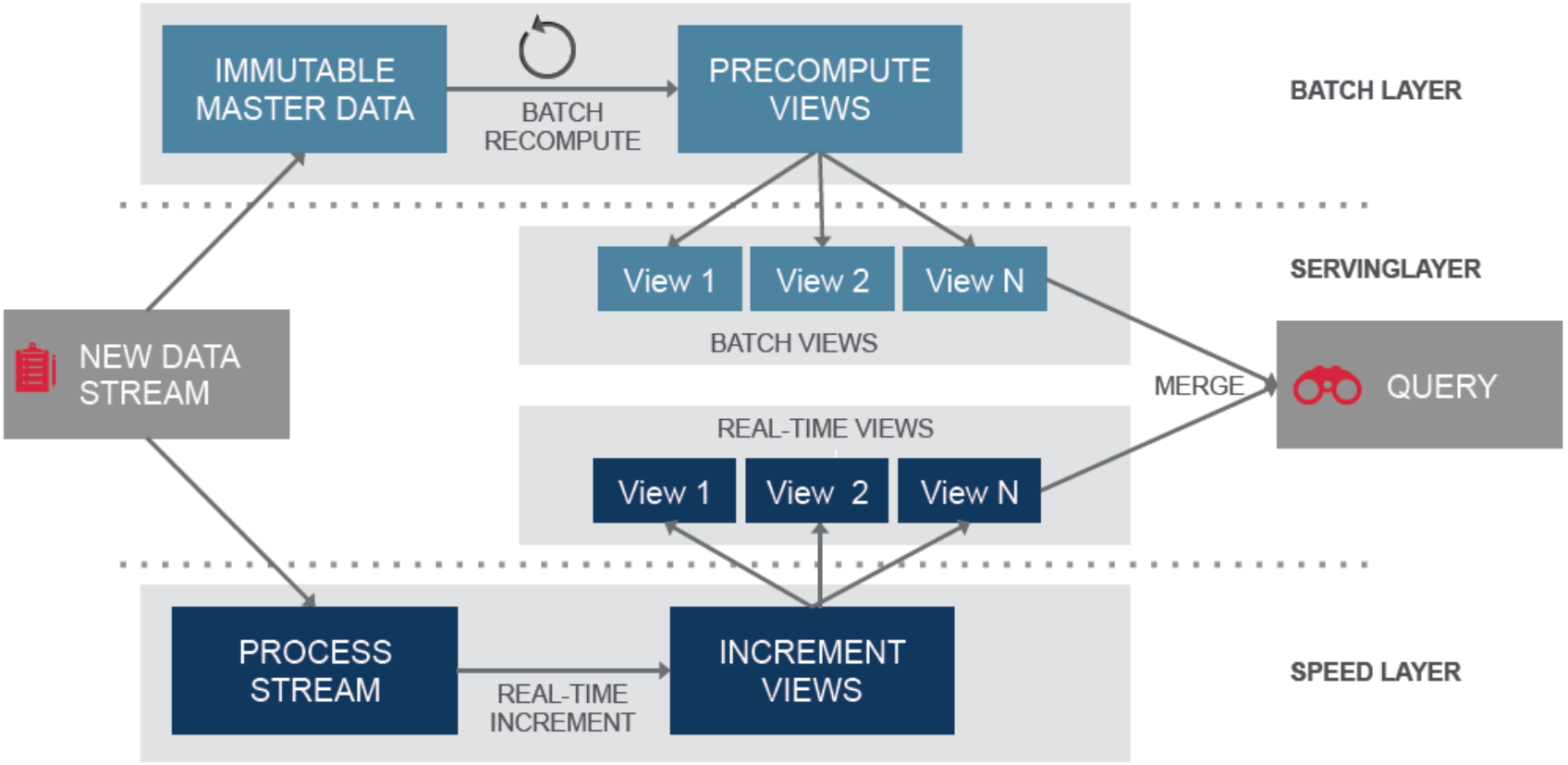


Big Data Architecture Pattern!



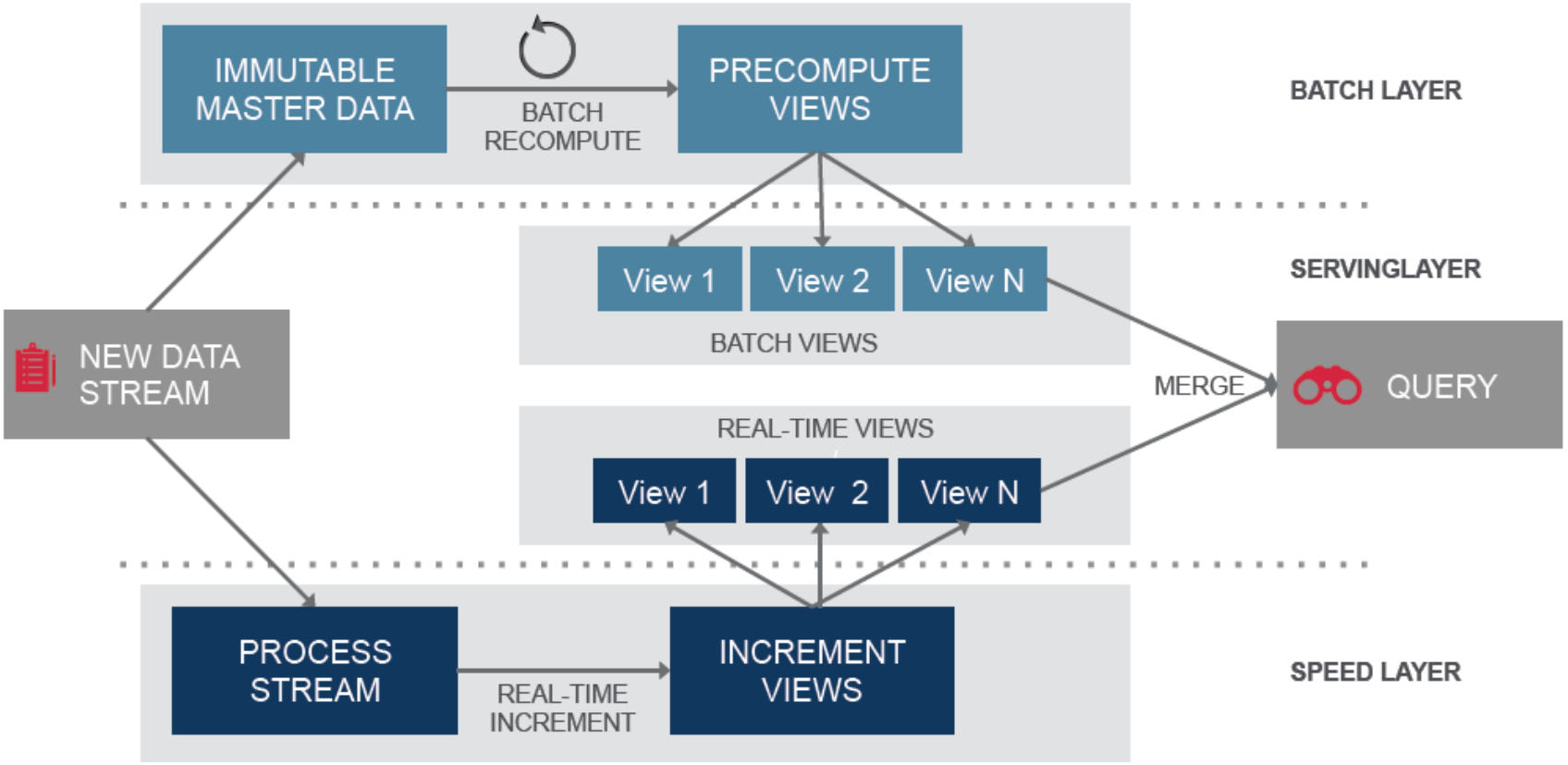
Batch Layer			
Stores an immutable constantly growing dataset	Computes arbitrary views from this dataset	View creation can take hours	Views can always be recreated from master data

Big Data Architecture Pattern!



Speed Layer		
Computes \views from the constant stream of data it receives	Needed to compensate for the high latency of the batch layer	Incremental model and views are transient

Big Data Architecture Pattern!

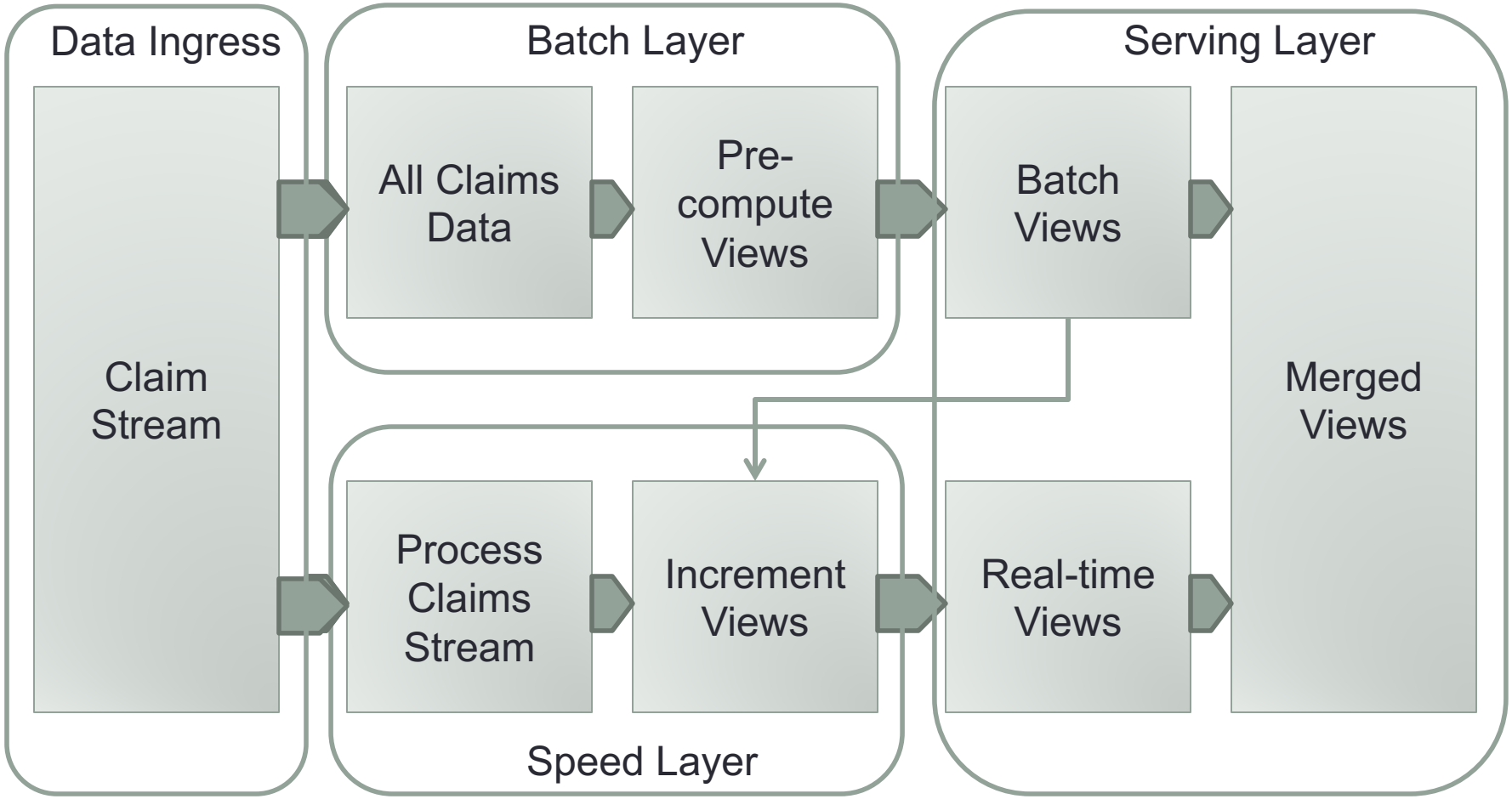


Serving Layer		
Responsible for exposing batch views so that they can be queried	Exposes the incremented real-time views for operational support	Merges batch and the real-time views into a consistent result

Benefits

- Higher value, immediate responses to events as they happen
- Competitive advantage delivering new insights about historical data
- No added overhead of managing separate, unnecessary data silos
- Reduced administrative overhead, lower risk of security gaps on data access
- Use the right technology for the job, less coding and less complexity
- Avoid discarding valuable historical context

Unified Claims Handling Batch and Real-Time



Lambda Architecture

Software Components