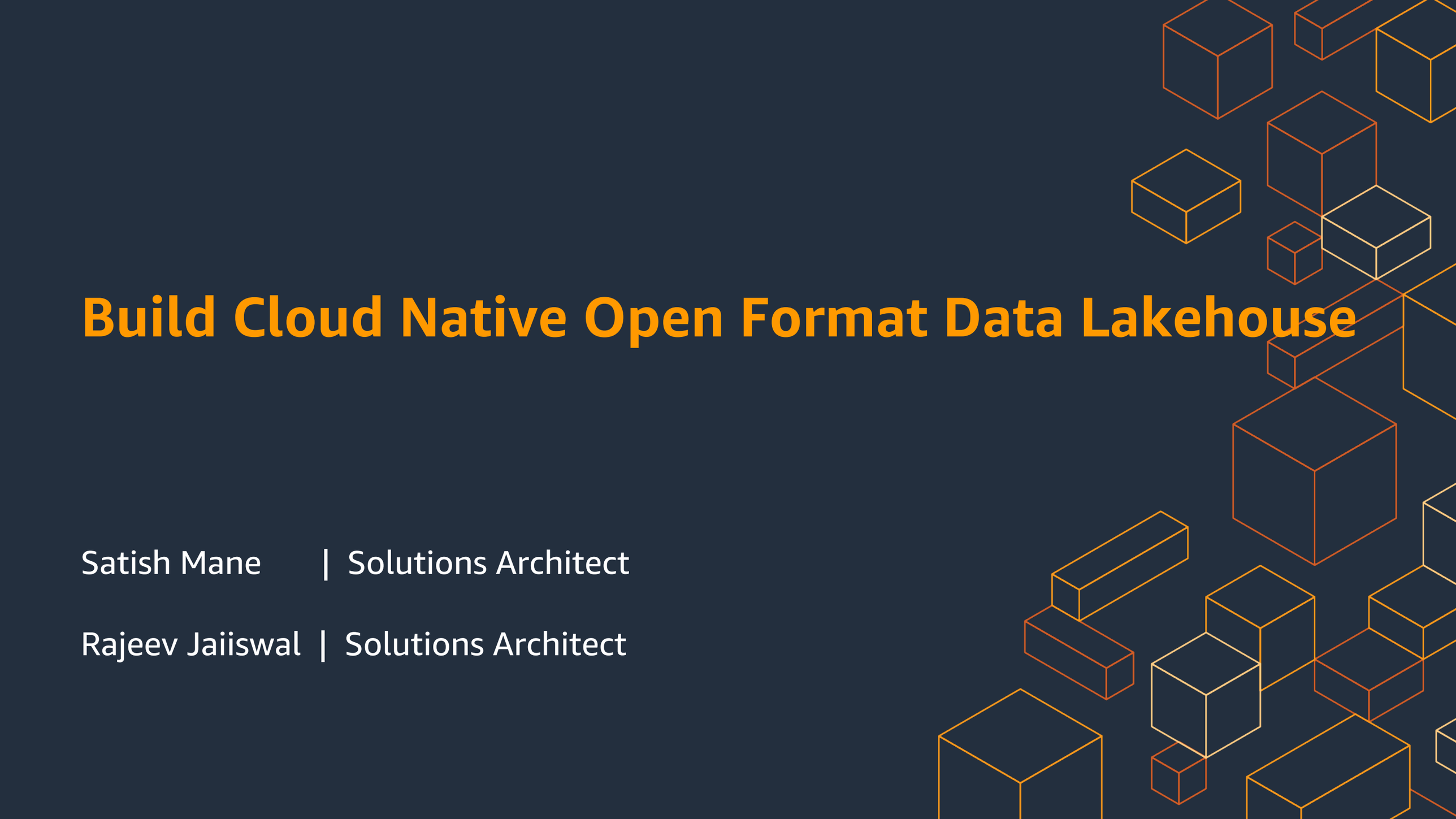
Build Cloud Native Open Format Data Lakehouse

Satish Mane | Solutions Architect

Rajeev Jaiiswal | Solutions Architect



Data Driven Digital Era – an imperative for the future

Anomaly and fraud detection

Tailoring customer experience in real time

Empowering IoT analytics

Nourishing marketing campaigns

Real-time personalization

Supporting healthcare and emergency services

Customers want more value from their data

Growing Exponentially

From new sources

Increasingly diverse

Used by many people

Analyzed by many applications

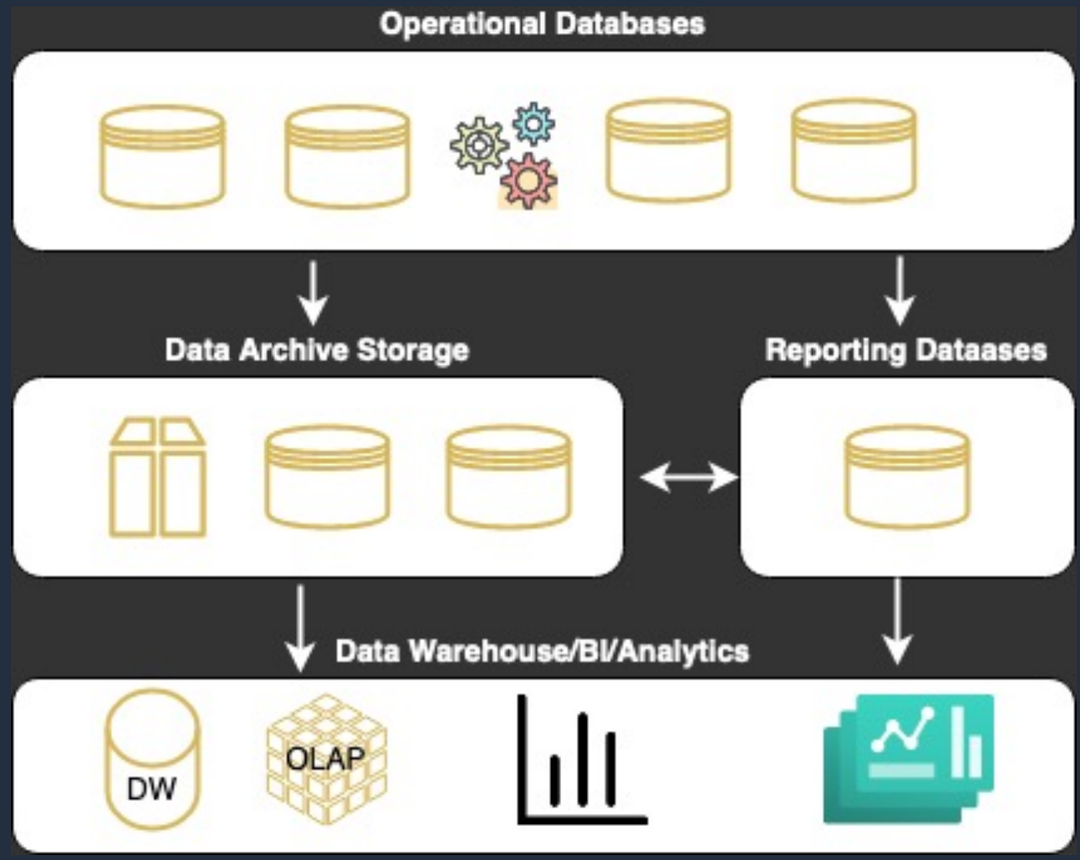
Traditional data analytics and its drawbacks

Data Storage

Retention Vs Cost

Data Collaboration

Compute at scale



What is scalable Data Lake

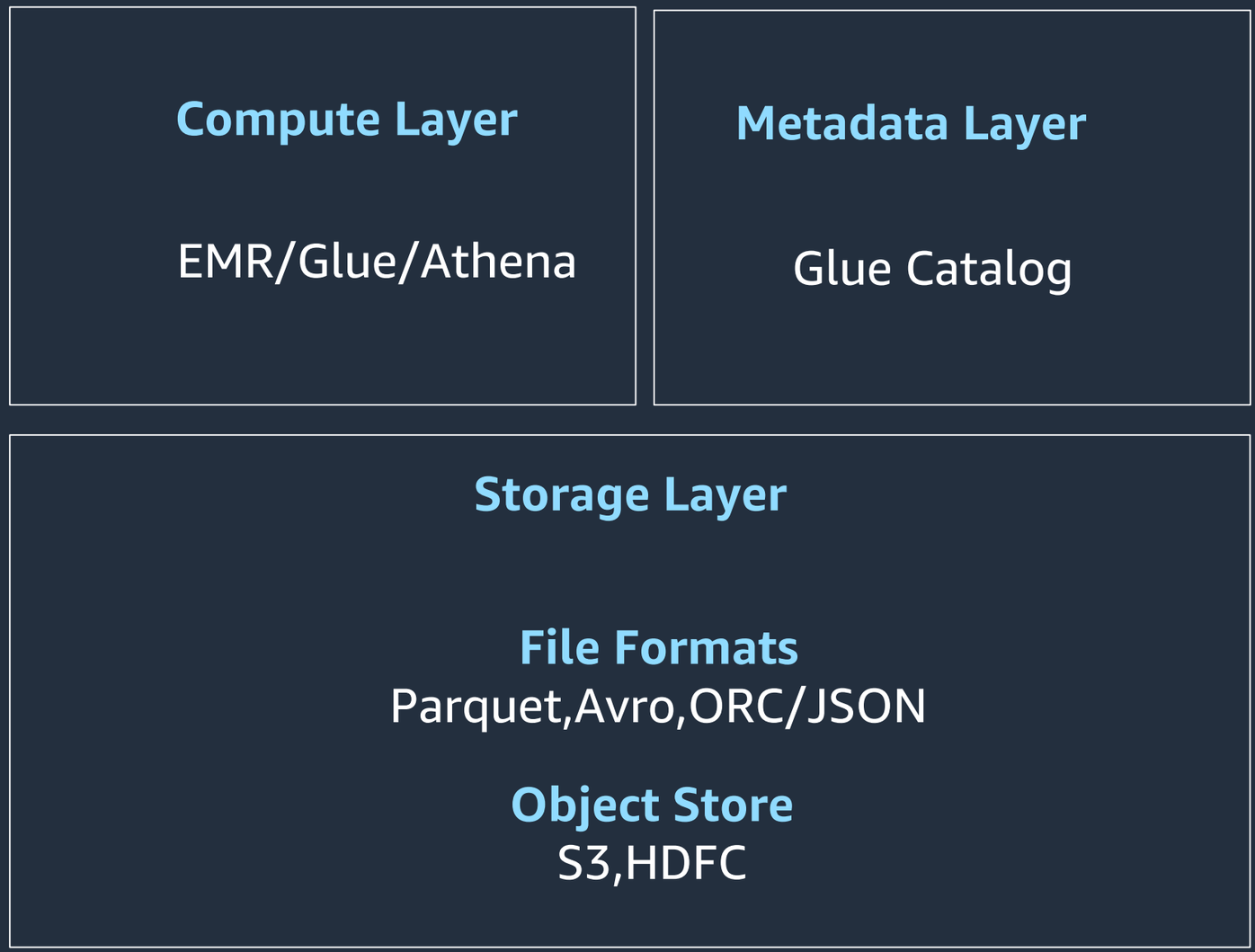
Variety of sources
and data types

Data volume and velocity

Increasing and
unpredictable cost

A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale

Components of Data lake on cloud

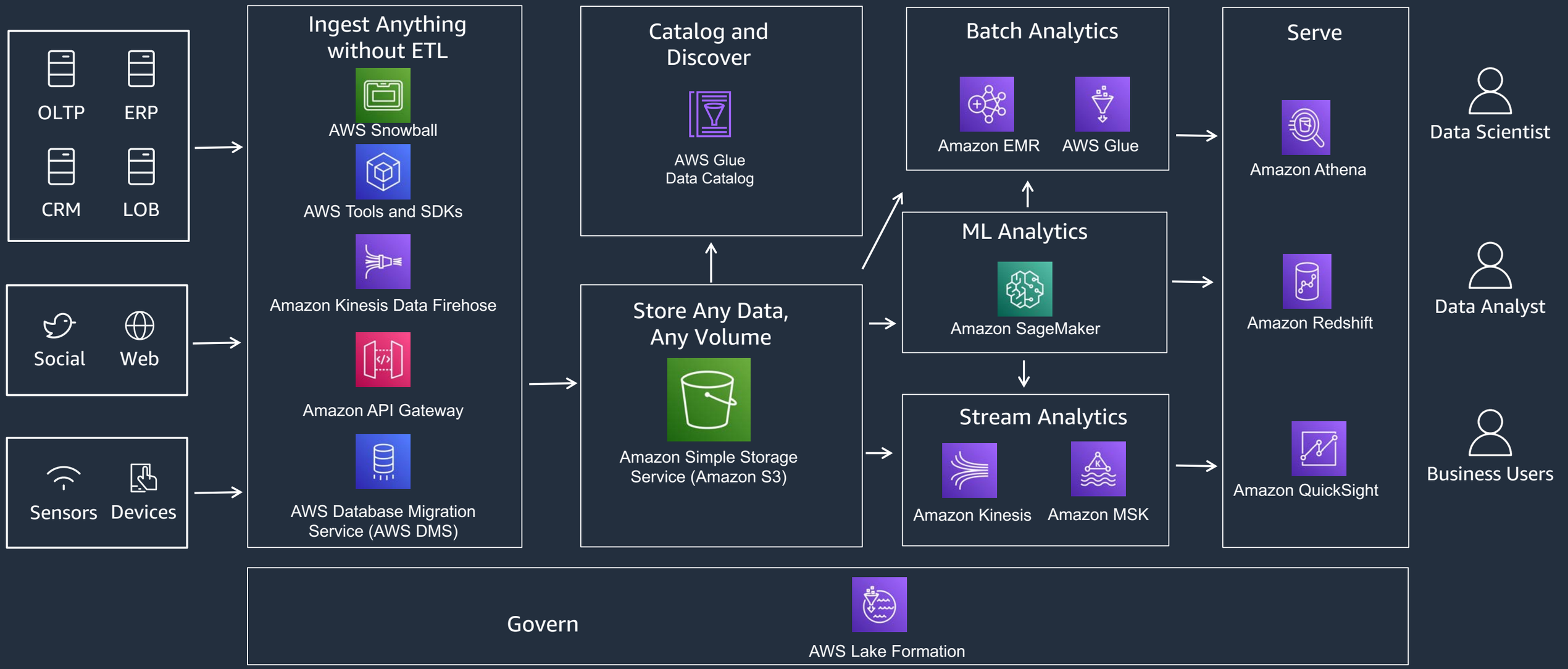


Building a an Open Data Lake
Store ANY data in ANY format

File Format	Properties	Use Cases
Orc	Columnar, schema stored in footer.	Read heavy analytical workloads, e.g. Hive Tables.
Parquet	Columnar, schema stored in footer.	Read heavy analytical workloads, e.g. Spark processing.
Avro	Row-major, schema and data separate.	Write heavy workloads, e.g. Apache Kafka.
CSV	Human readable, fixed schema.	Small volumes, consumer is an analyst.
JSON	Human readable, flexible schema.	Small volumes, consumer is an application.

Figure 3: Data storage properties and use cases

Data Lake Architecture if you decide to build one on AWS



A note on Serverless.
Why Serverless Data Lake?

Lakehouse Architecture Dive Deep

What are the challenges with regular data lake

Data Re-processing issues

Change Data Capture pipeline

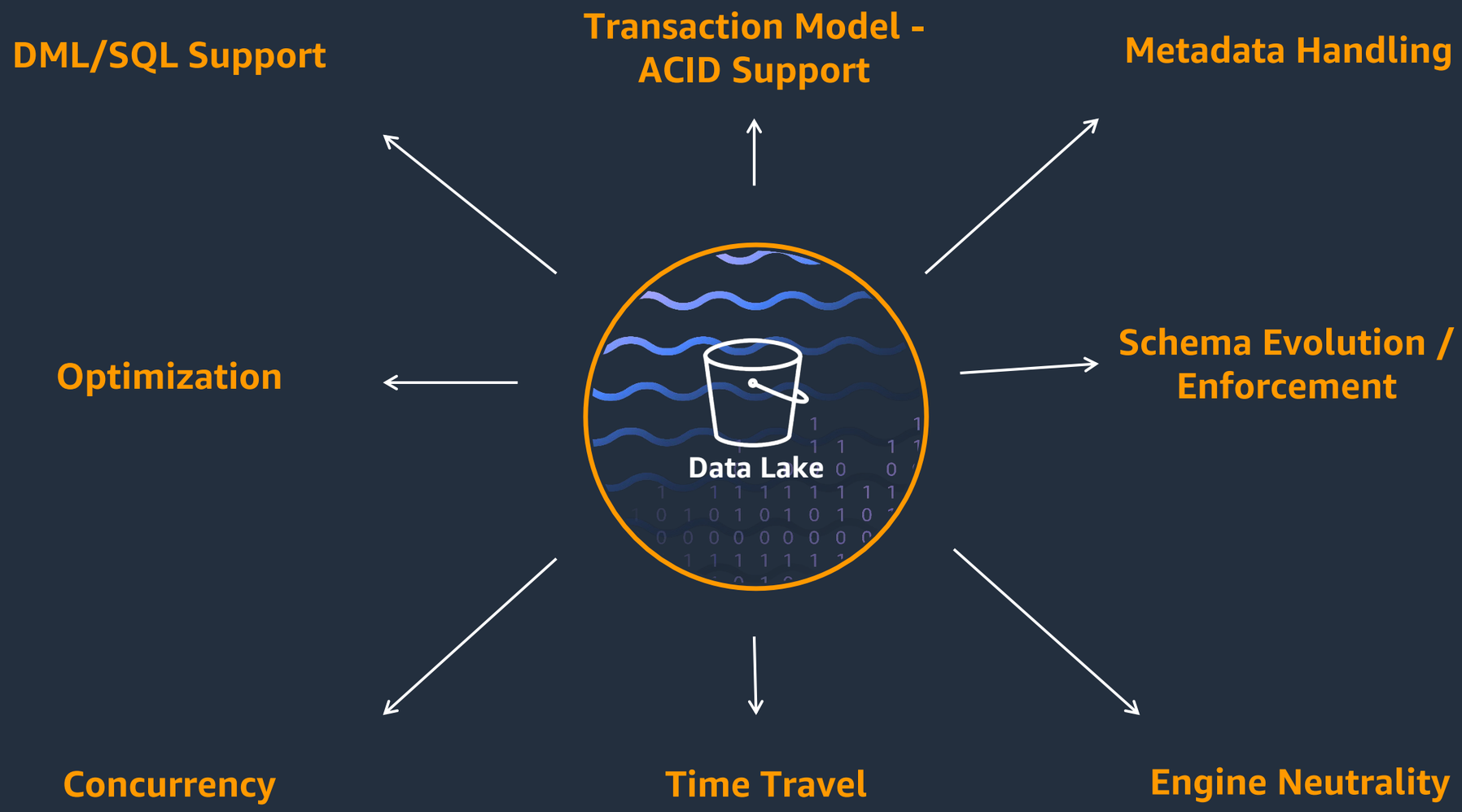
Lack of data quality enforcements

Performance optimization

Transaction across partitions

Must know table structure

How open table format solves those challenges



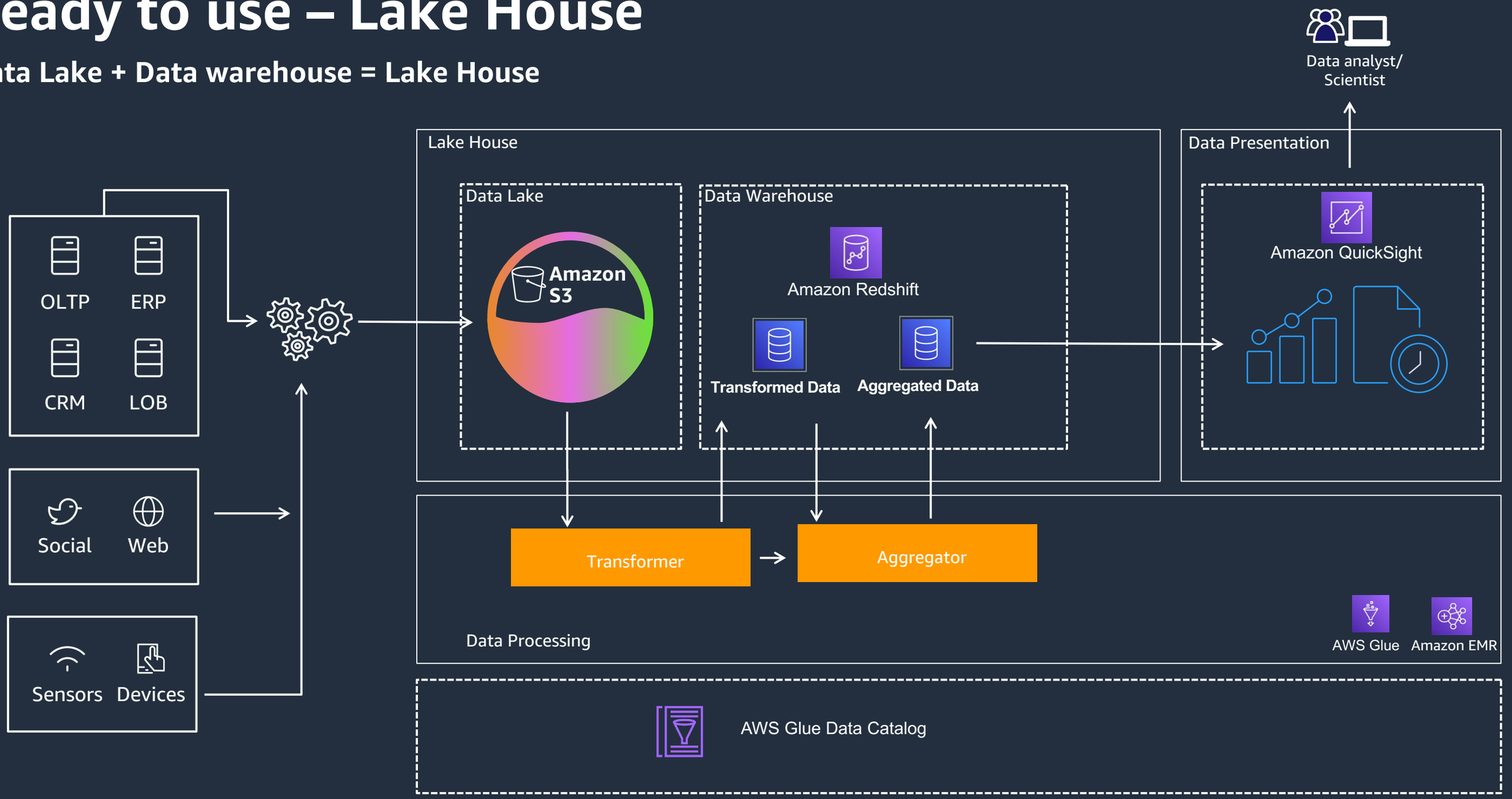
What are the options to build data lakehouse

Data Lake vs Data Lakehouse vs Data Warehouse

	Data Lake	Data Warehouse	Data Lakehouse
Nature of data	All types (structured, semi-structured, raw)	Structured only	structured, semi-structured, raw
Table format	Hive	Proprietary format	Hudi/Delta Lake/Iceberg
Schema	On Read	On write	On Write
File Format	Parquet/Avro/ORC/JSON	Parquet/Avro/ORC/JSON (Ingestion - Redshift)	Parquet/Avro/ORC/JSON
Cost	$	$$$	$
Scalability	Any amount of data	Scale up, but more expensive	Any amount of data
How easy to operate	Difficult	Simple	Simple
Transactional features	Not supported	ACID/DML/Concurrency	ACID/DML/Concurrency
Optimization	Storage level	Storage/Write/Read	Storage/Write/Read
Performance	Low	High	High
Users	Data scientist	Data Analyst	Data Scientists, Data Analyst

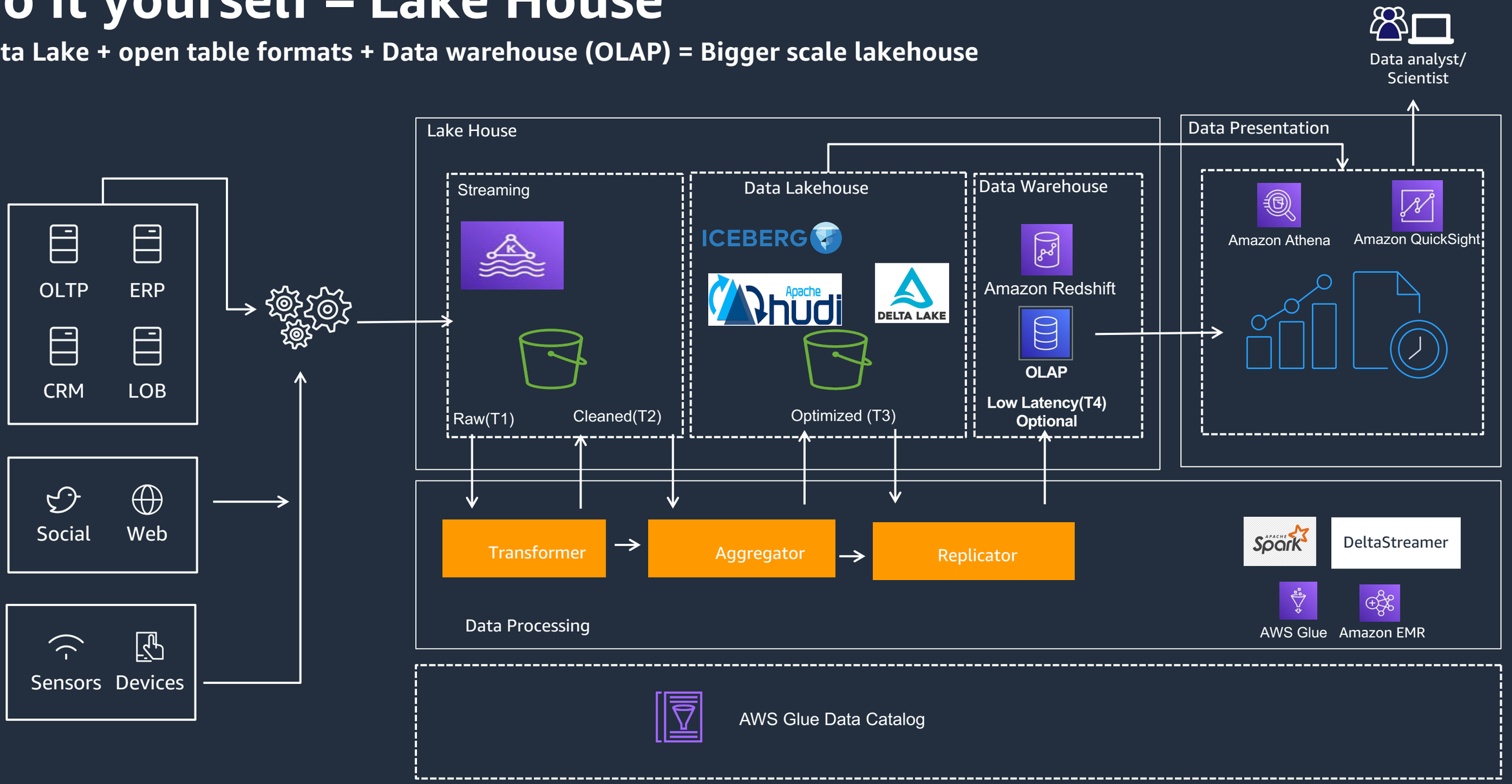
Ready to use – Lake House

Data Lake + Data warehouse = Lake House



Do it yourself – Lake House

Data Lake + open table formats + Data warehouse (OLAP) = Bigger scale lakehouse



Apache Hudi

Transaction Model
Timeline

Concurrency Control
MVCC/OCC

Time Travel
Hudi commit time

Schema Evolution
Add/Delete/Change

Storage Optimization
Auto Compaction/File Sizing

Index
Multi Modal Indexes

	COW	MOR
Write Cost	Higher	Lower
Data Latency	Slower	Faster
Query Speed	Faster	Slower before compaction
		Same after compaction
Overall Cost	Aggressive rewrites with every update	Can amortize compaction with other services

Apache Iceberg

Transaction Model
Snapshot

Concurrency Control
OCC

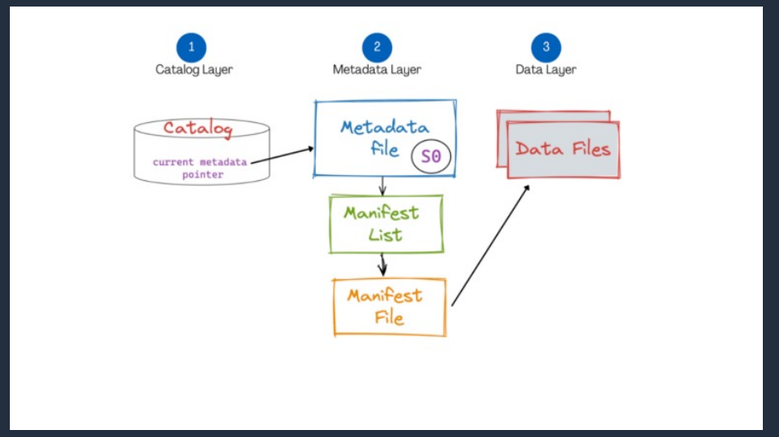
Time Travel
Snapshot id and timestamp

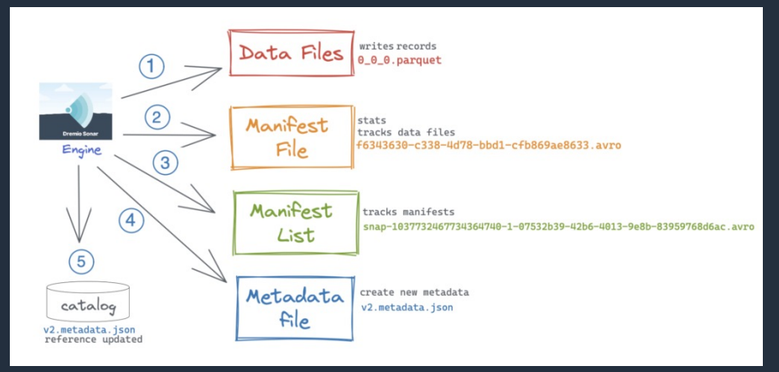
Schema Evolution
Add/Delete/Change

Partition Evolution
Partition column change

Storage Optimization
Manual Compaction/File Sizing

Index
Column min/max index





Linux Foundation Delta Lake

Transaction Model
Transaction log

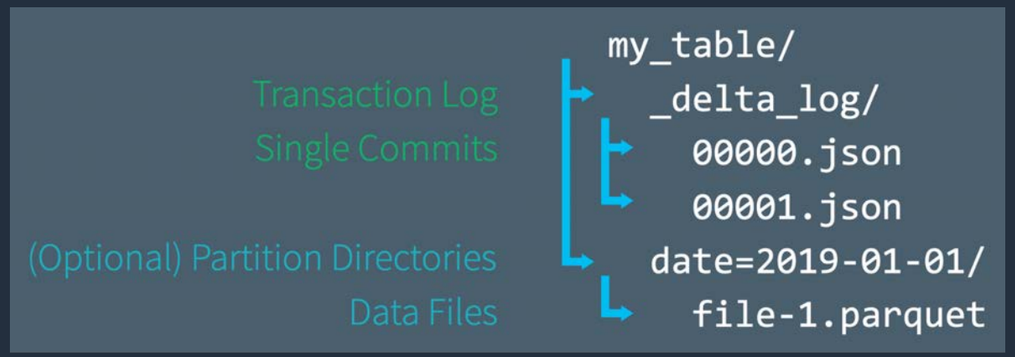
Concurrency Control
OCC

Time Travel
Timestamp or version number

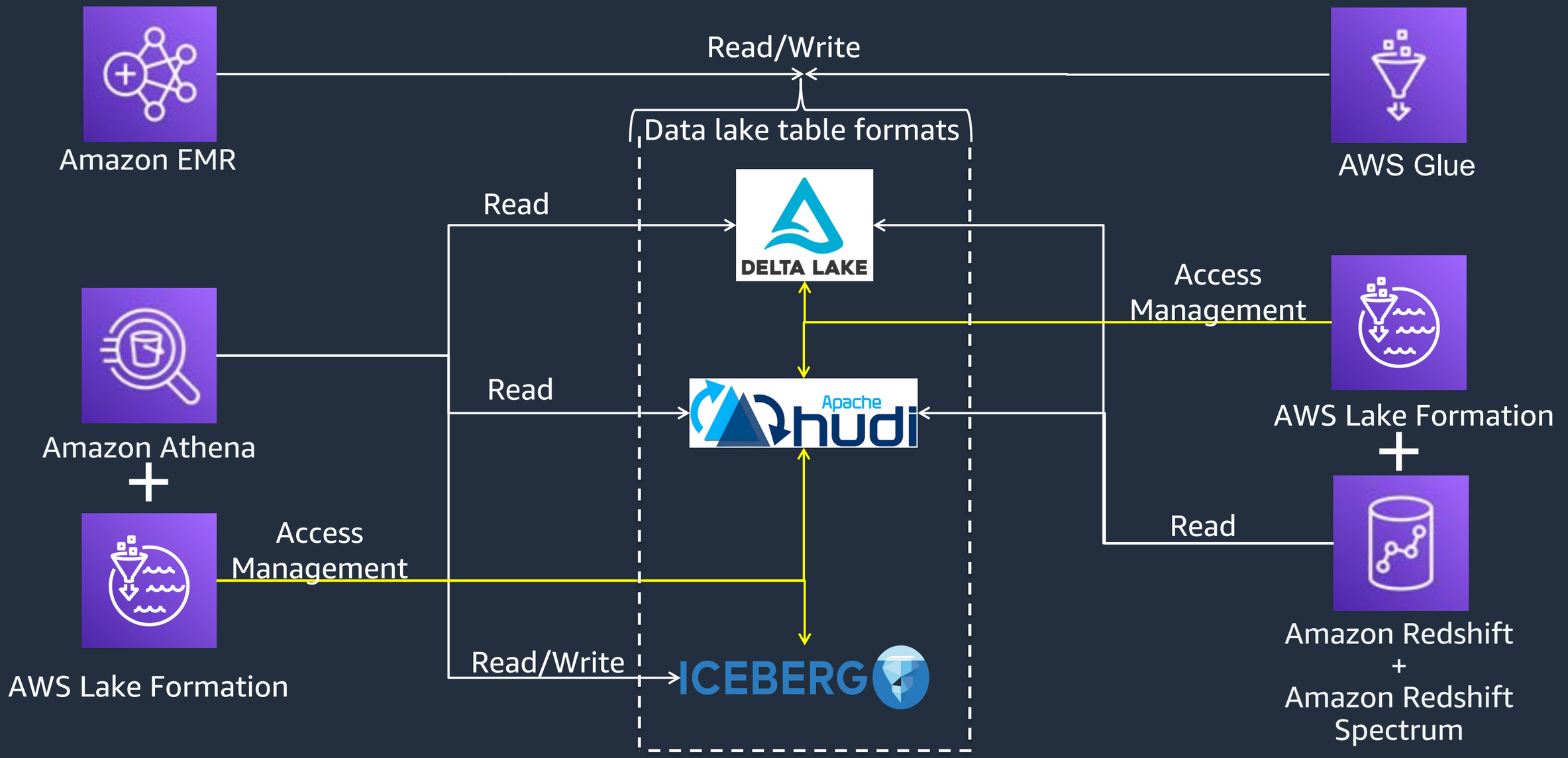
Schema Evolution
Add/Delete/Change

Storage Optimization
Manual File Sizing

Index
Column min/max + Z order index



Integration with AWS Analytics Services



Hudi vs Iceberg vs Delta Lake - Comparison

	Apache Hudi	Iceberg	Delta Lake
Transaction Model	Timeline	Snapshot	Transaction Log
Upsert	CoW + MoR	CoW + MoR	CoW
Data Model	Primary Key + Partition Key	Partition Key	Partition Key
Indexing / Data Skipping	Multi modal Index : • Partition index for file listing • Min, max col index for range pruning of files • Bloom Filter Index for record level pruning	• Col stats (min,max) in manifest file pruning. • Partition values to filter out manifest files	• Min/max col stats to skip the files. • Bloom filter based record level filtering
Schema Evolution	Add/Delete/Modify Columns. Parition columns can not be changed	Add/Delete/Modify Columns. Parition columns can changed	Add/Delete/Modify Columns. Partition columns can not be changed
Data File Format	Parquet	Parquet,ORC,Avro	Parquet
Log File Format	Versioned Avro	Versioned Avro	Incremental Json files plus checkpoint parquet
Meta Data Table	HFile based Internal Hudi Table	Avro manifest file based lookup	Tx logs + checkpoints lookup
File Sizing	Auto / Manual	Manual using Optimize cmd	Manual using spark action
Compaction	Managed sync / async	Manual Spark job action	No compaction as it follows CoW model
Cleaning	Managed cleaning utility	Spark job to expire and clean snapshots. Metadata files auto cleaned	Run Vacuum CMD manually to delete expired data files. Transaction logs are auto cleaned after checkpoint
Managed Data Ingestion	Delta Streamer utility	None	Auto Loader proprietary (Not part of Delta Lake)

Thank you!

Satish Mane, Solutions Architect
sbmane@amazon.com

Rajeev Jaiiswal, Solutions Architect
rjjaiisw@amazon.com