Deploying a Data Lake in AWS

Siva Raghupathy, Senior Manager
Big Data Solutions Architecture, AWS

March 21, 2017

amazon web services | Webinars

Agenda

Data Lake Concepts

Simplify Data Lake

What technologies should you use?

- Why?
- How?

Reference architecture

Design patterns

What is a Data Lake?

It is an architecture that allows you to collect, store, process, analyze and consume all data that flows into your organization.

Why Data Lake?

- Leverage **all data** that flows into your organization
  - Customer centricity
  - Business agility
  - Better predictions via Machine Learning
  - Competitive advantage

Data Lake Enablers

- Big Data technology evolution
- Cloud services evolution/economics
- Big Data + Cloud architecture convergence

Big Data Evolution

Batch processing

Stream processing

Artificial Intelligence

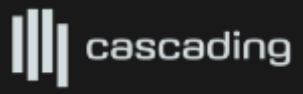
Cloud Services Evolution

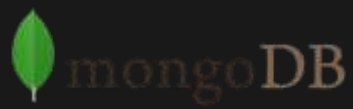
Virtual machines

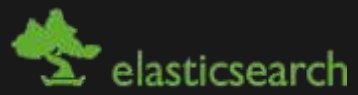
Managed services

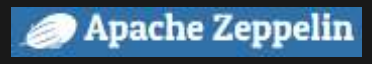
Serverless

Plethora of Tools









EMR

S3

DynamoDB

SQS

Amazon Redshift

Amazon Glacier

RDS

ElastiCache

Amazon Kinesis

Amazon QuickSight

Data Pipeline

Amazon Elasticsearch Service

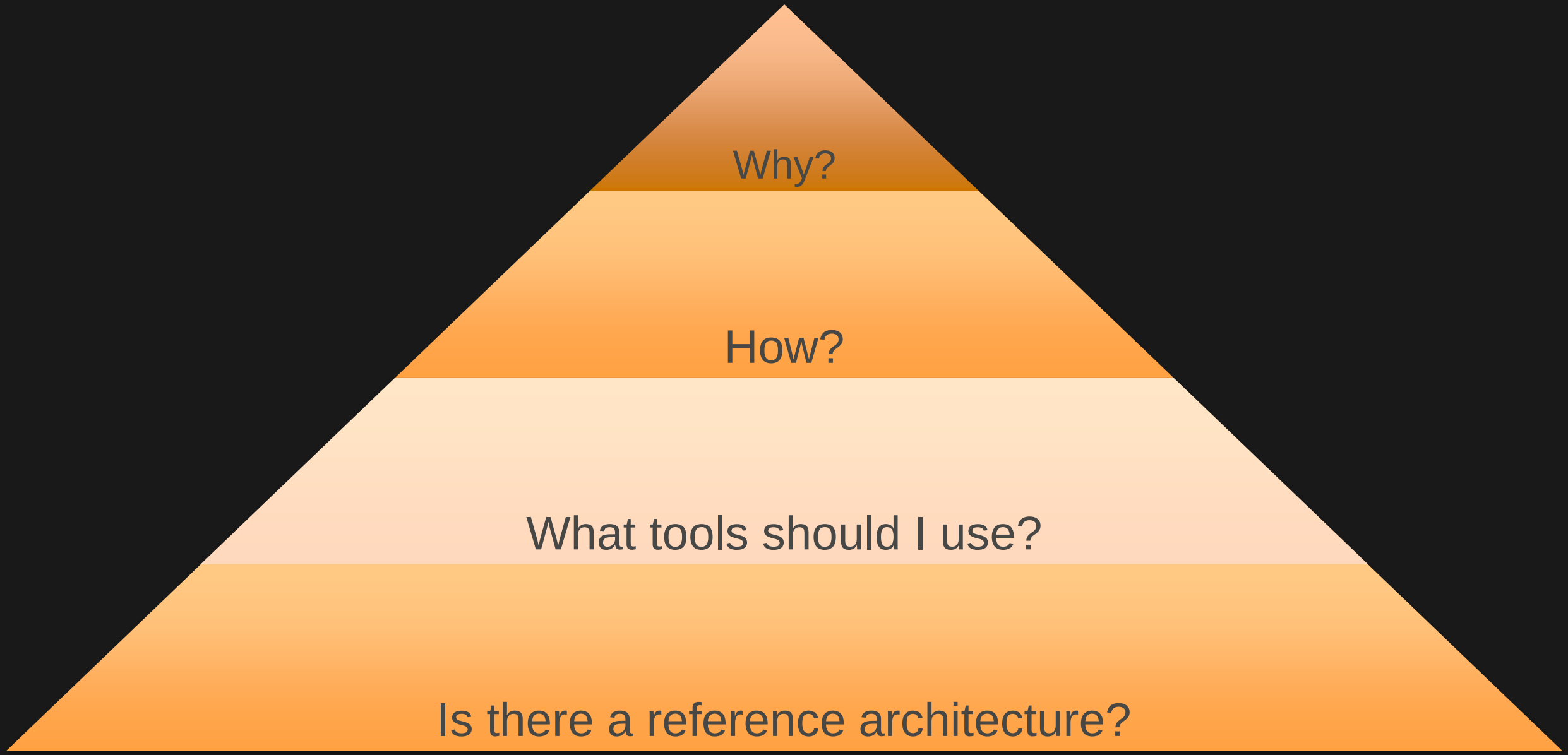
Lambda

Amazon ML

DynamoDB Streams

Amazon Kinesis Analytics

Data Lake Challenges



Architectural Principles

Build decoupled systems

- Data → Store → Process → Store → Analyze → Answers

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Leverage AWS managed services

- Scalable/elastic, available, reliable, secure, no/low admin

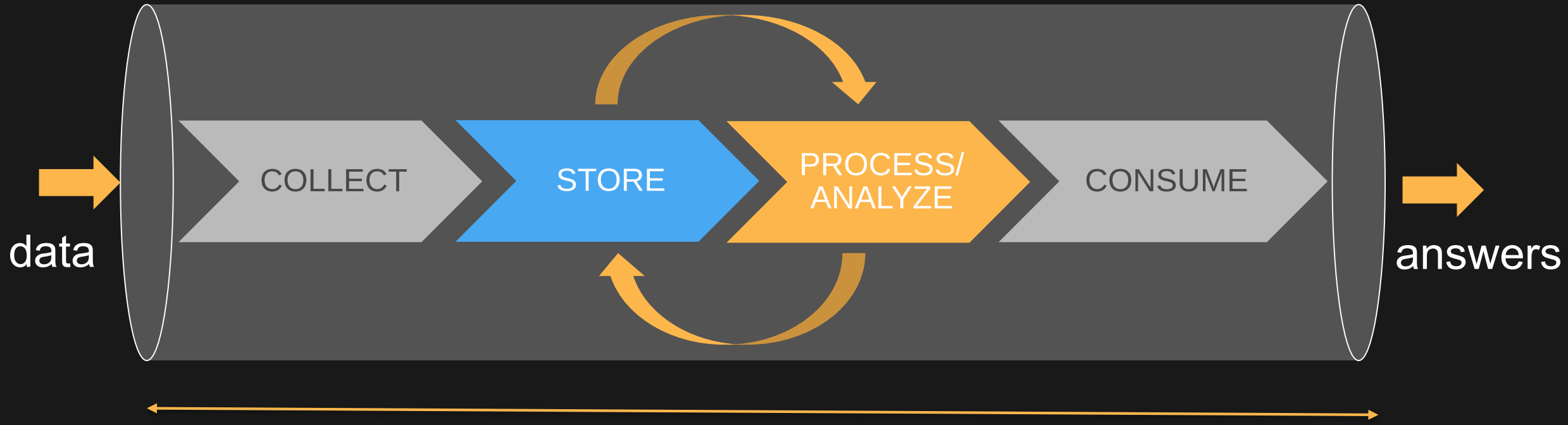
Use log-centric design patterns

- Immutable logs, materialized views

Be cost-conscious

- Big data ≠ big cost

Simplify Data Lake

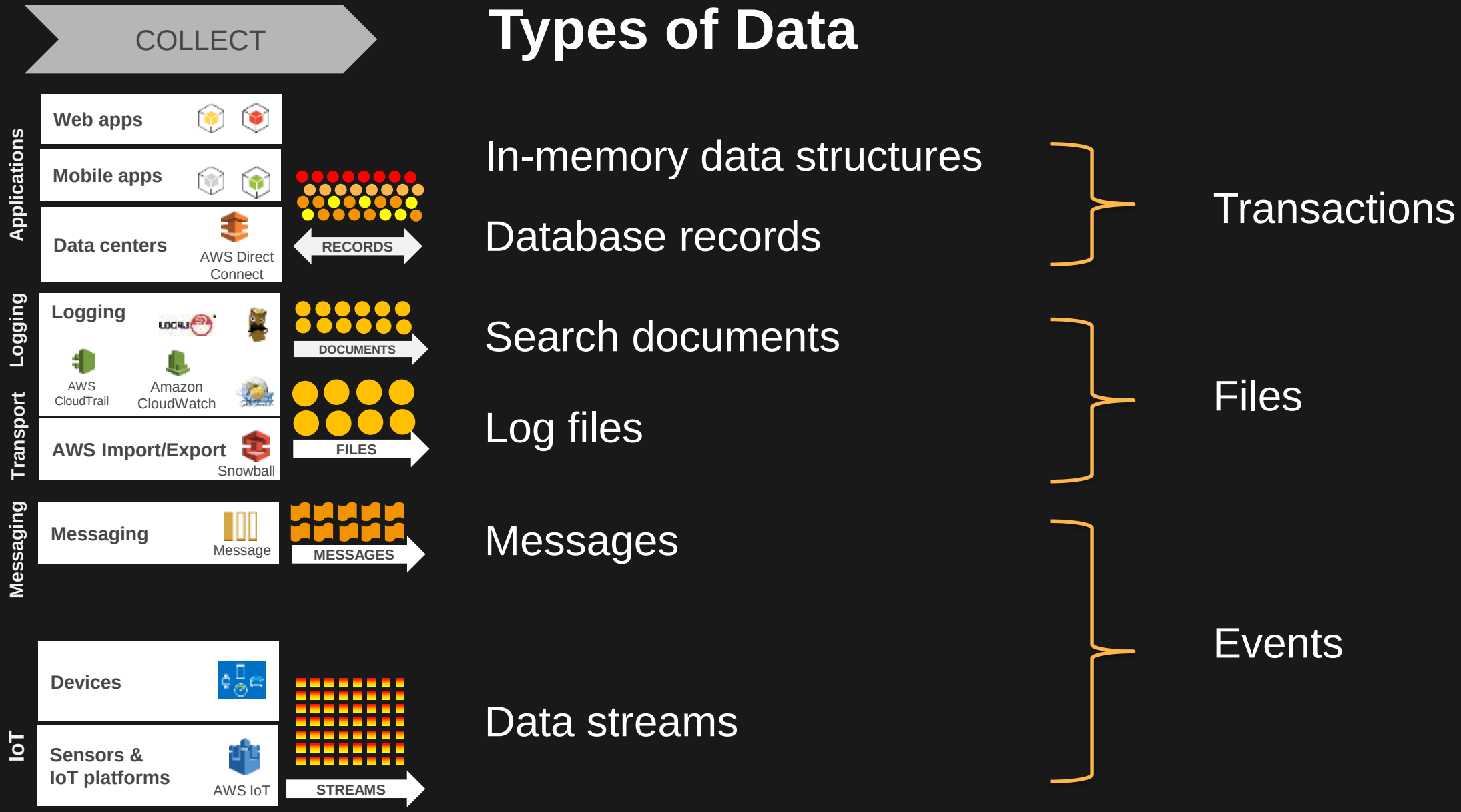


Time to answer (Latency)

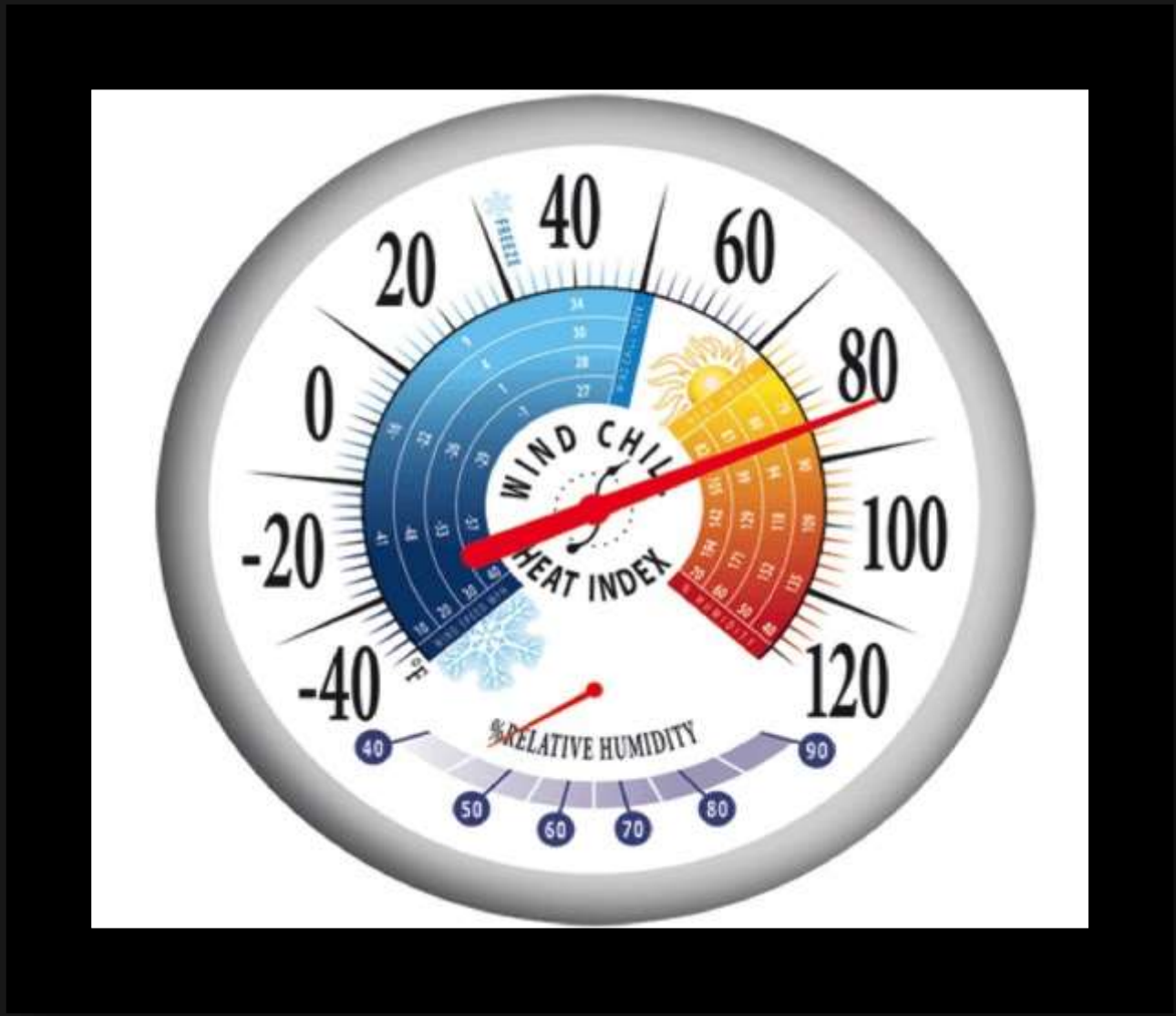
Throughput

Cost

Types of Data



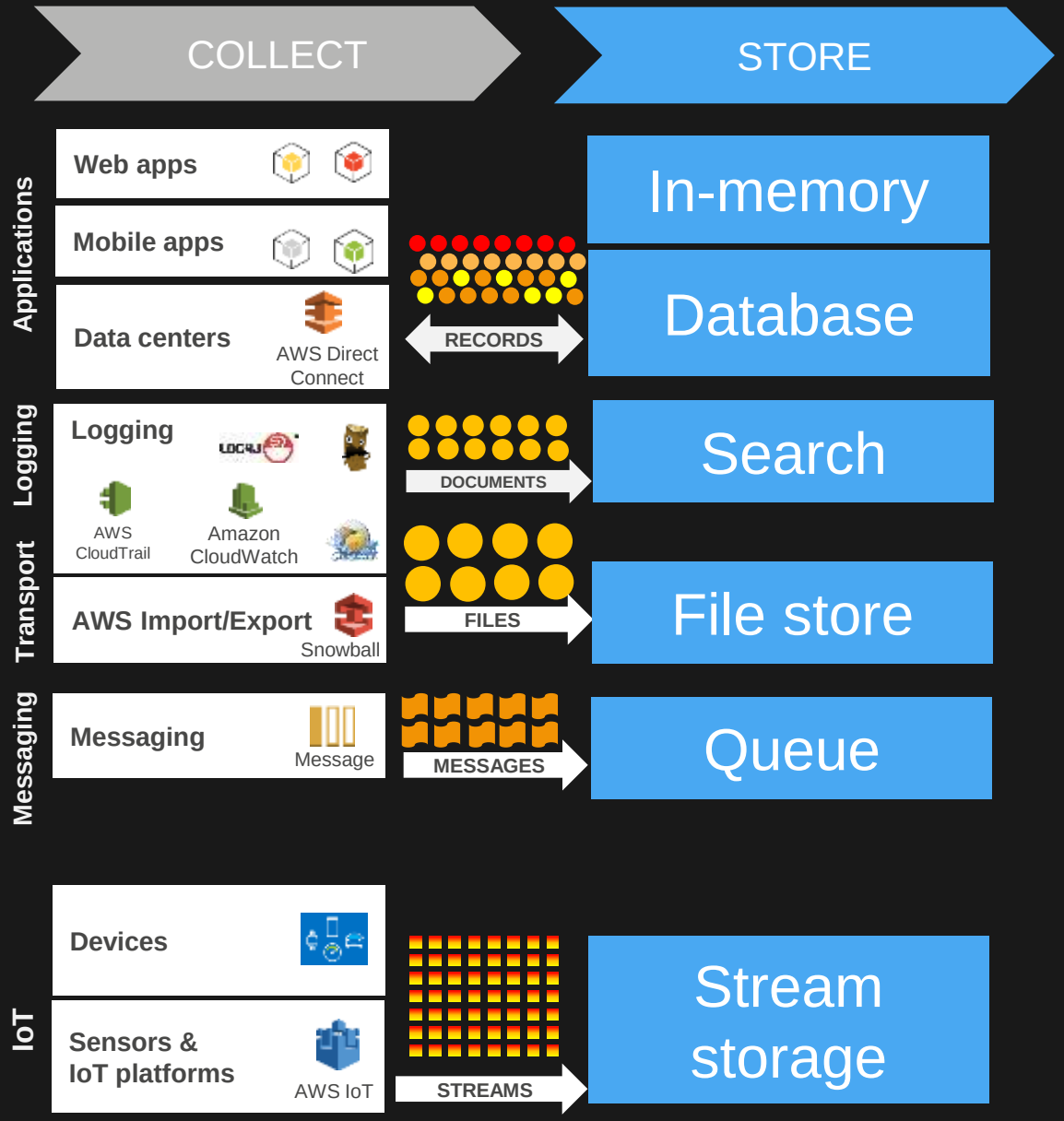
What Is the Temperature of Your Data ?



Data Characteristics: Hot, Warm, Cold

	Hot	Warm	Cold
Volume	MB–GB	GB–TB	PB–EB
Item size	B–KB	KB–MB	KB–TB
Latency	ms	ms, sec	min, hrs
Durability	Low–high	High	Very high
Request rate	Very high	High	Low
Cost/GB	$$-$	$-¢¢	¢


Store



Types of Data Stores

Caches, data structure servers

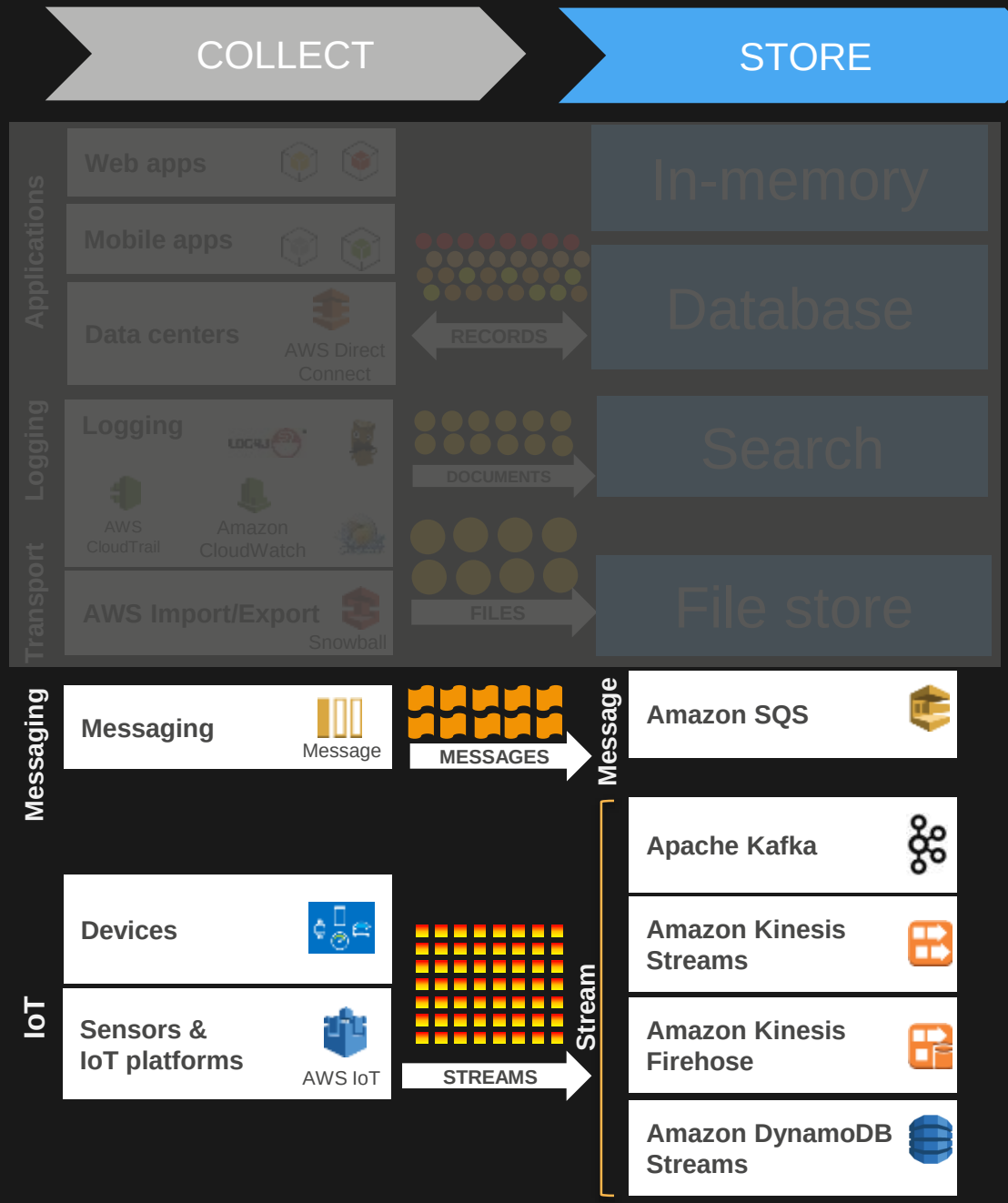
SQL & NoSQL databases

Search engines

File systems

Message queues

Pub/sub message queues



Message & Stream Storage

Amazon SQS

- Managed message queue service

Apache Kafka

- High throughput distributed streaming pla

Amazon Kinesis Streams

- Managed stream storage + processing

Amazon Kinesis Firehose

- Managed data delivery

Amazon DynamoDB

- Managed NoSQL database
- Tables can be stream-enabled

Why Stream Storage?

Decouple producers & consumers

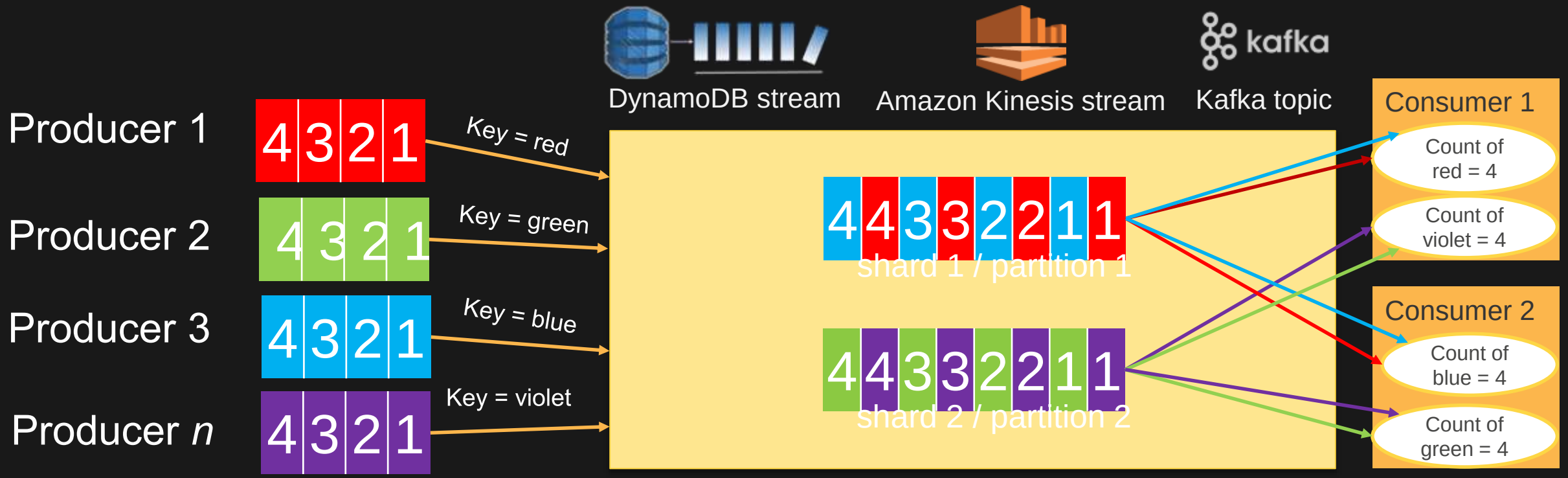
Persistent buffer

Collect multiple streams

Preserve client ordering

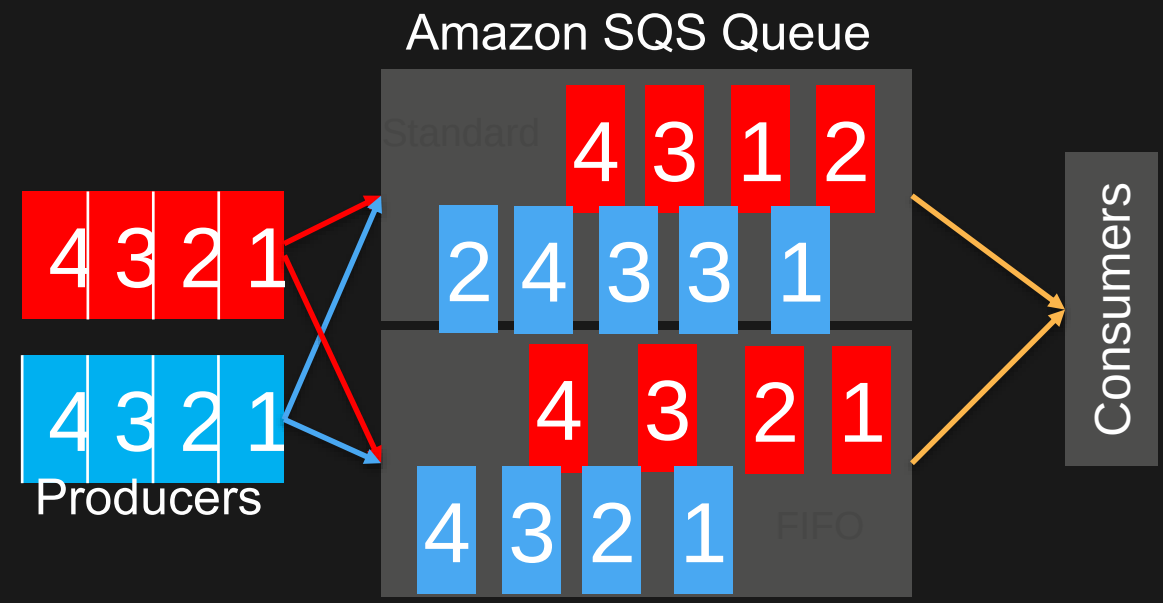
Parallel consumption

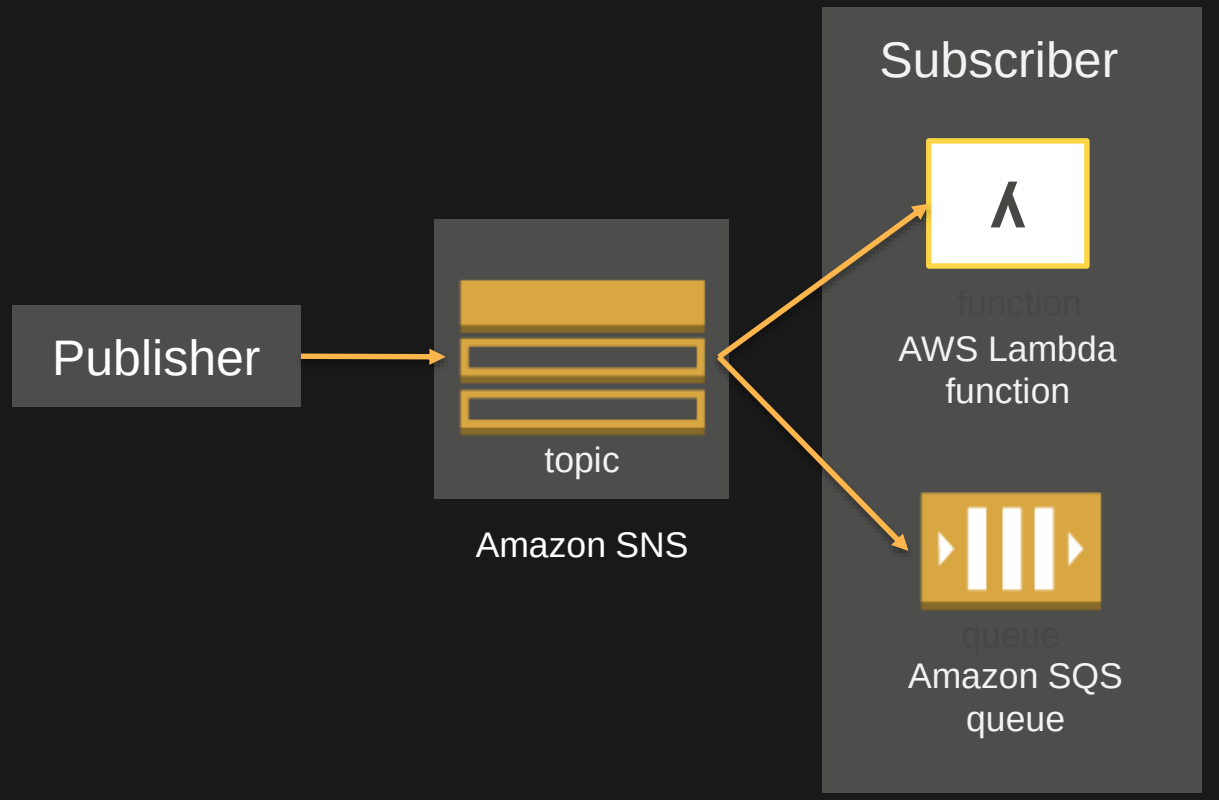
Streaming MapReduce



What About Amazon SQS?

- Decouple producers & consumers
- Persistent buffer
- Collect multiple streams
- **No** client ordering (Standard)
  - FIFO queue preserves client ordering
- **No** streaming MapReduce
- **No** parallel consumption
  - Amazon SNS can publish to multiple SNS subscribers (queues or λ functions)



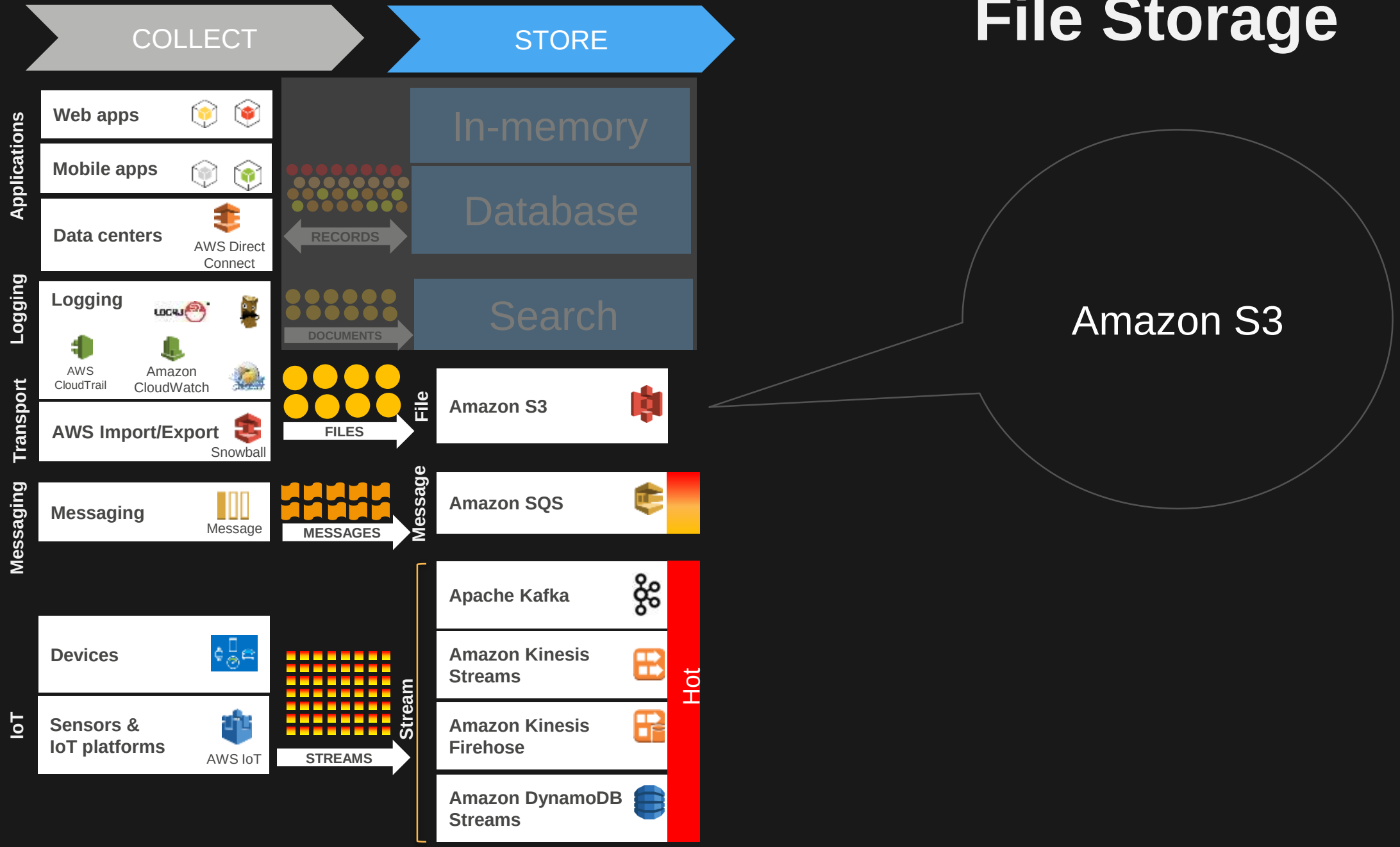


Which Stream/Message Storage Should I Use?

	Amazon DynamoDB Streams	Amazon Kinesis Streams	Amazon Kinesis Firehose	Apache Kafka	Amazon SQS (Standard)	Amazon SQS (FIFO) New
AWS managed	Yes	Yes	Yes	No	Yes	Yes
Guaranteed ordering	Yes	Yes	No	Yes	No	Yes
Delivery (deduping)	Exactly-once	At-least-once	At-least-once	At-least-once	At-least-once	Exactly-once
Data retention period	24 hours	7 days	N/A	Configurable	14 days	14 days
Availability	3 AZ	3 AZ	3 AZ	Configurable	3 AZ	3 AZ
Scale / throughput	No limit / ~ table IOPS	No limit / ~ shards	No limit / automatic	No limit / ~ nodes	No limits / automatic	300 TPS / queue
Parallel consumption	Yes	Yes	No	Yes	No	No
Stream MapReduce	Yes	Yes	N/A	Yes	N/A	N/A
Row/object size	400 KB	1 MB	Destination row/object size	Configurable	256 KB	256 KB
Cost	Higher (table cost)	Low	Low	Low (+admin)	Low-medium	Low-medium

Hot — Warm

File Storage



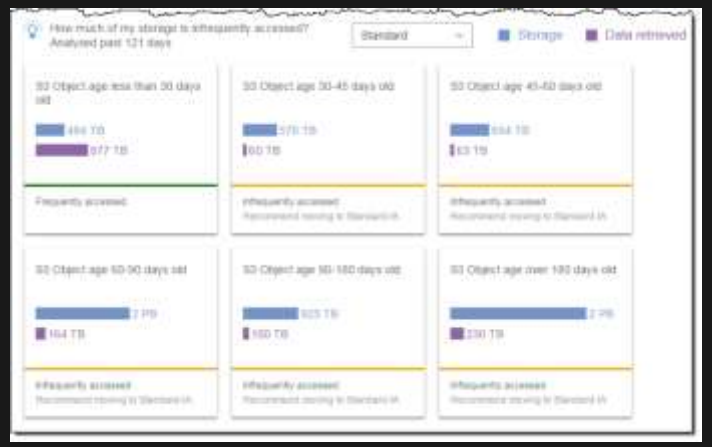
Why Is Amazon S3 the Fabric of Data Lake?

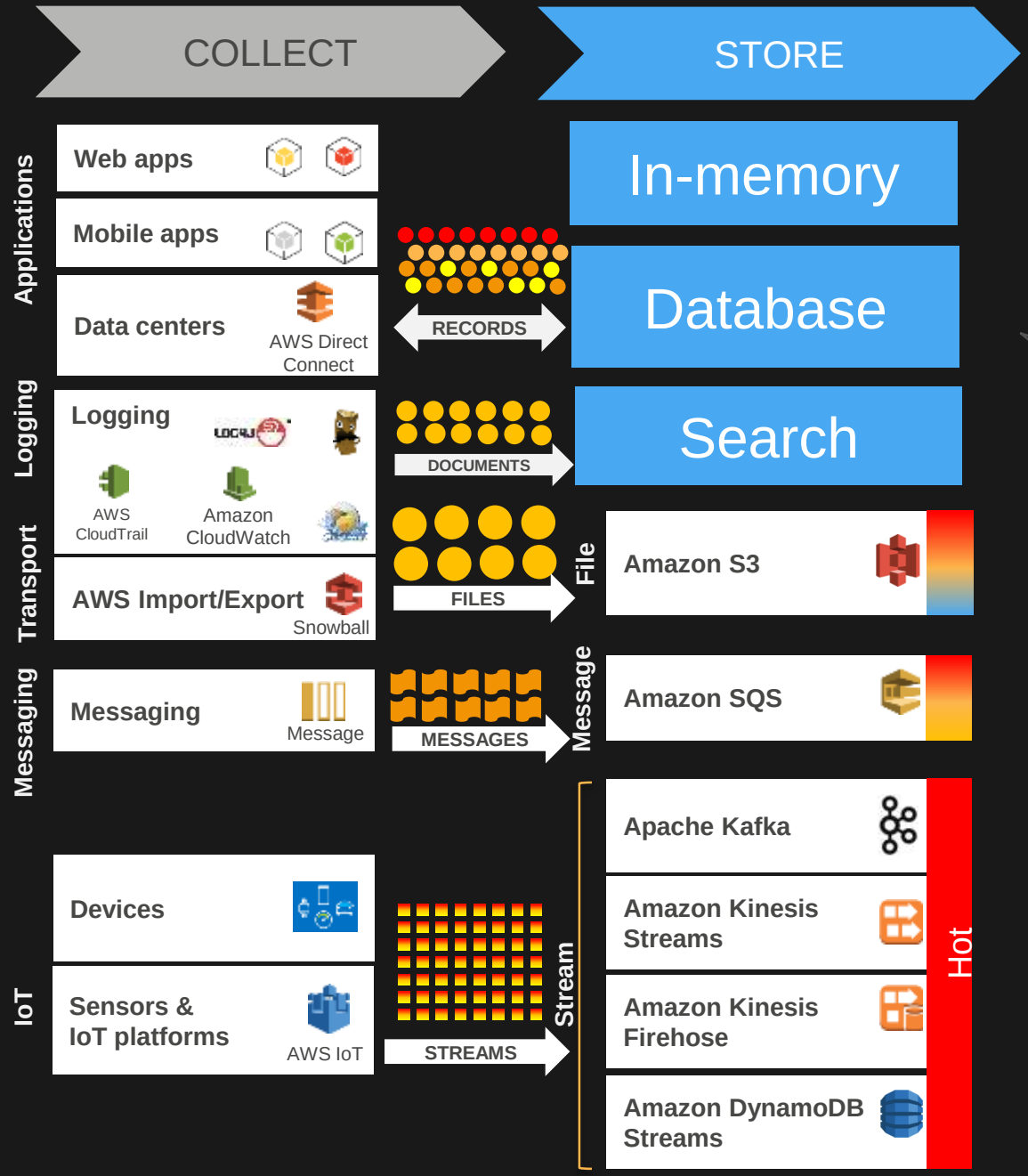
- Natively supported by big data frameworks (Spark, Hive, Presto, etc.)
- Decouple storage and compute
  - No need to run compute clusters for storage (unlike HDFS)
  - Can run transient Hadoop clusters & Amazon EC2 Spot Instances
  - Multiple & heterogeneous analysis clusters can use the same data
- Unlimited number of objects and volume of data
- Very high bandwidth – no aggregate throughput limit
- Designed for 99.99% availability – can tolerate zone failure
- Designed for 99.999999999% durability
- No need to pay for data replication
- Native support for versioning
- Tiered-storage (Standard, IA, Amazon Glacier) via life-cycle policies
- Secure – SSL, client/server-side encryption at rest
- Low cost

What About HDFS & Data Tiering?

- Use HDFS for very frequently accessed (hot) data
- Use Amazon S3 Standard for frequently accessed data
- Use Amazon S3 Standard – IA for less frequently accessed data
- Use Amazon Glacier for archiving cold data
- Use Amazon S3 Analytics for storage class analysis

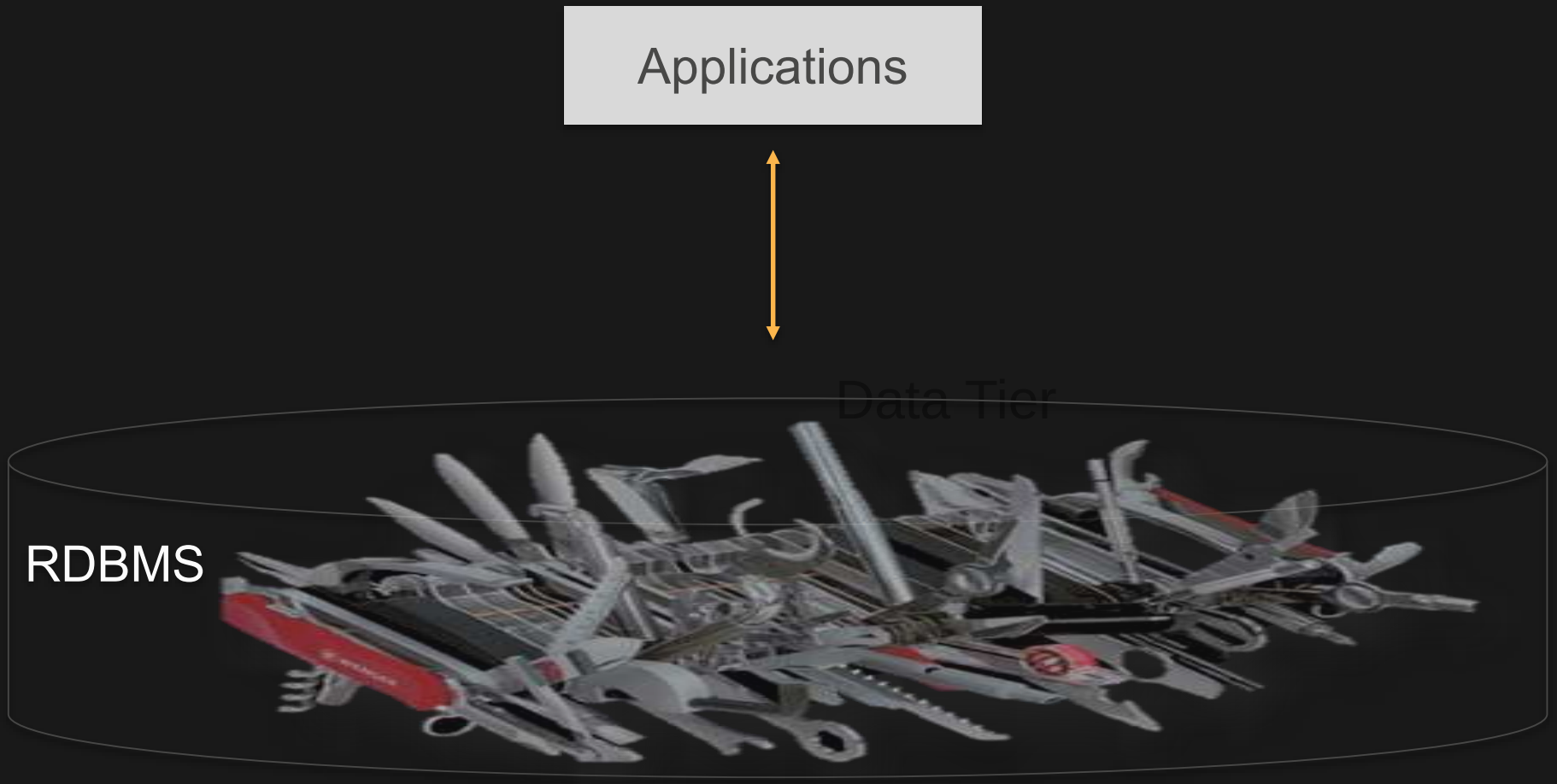
New



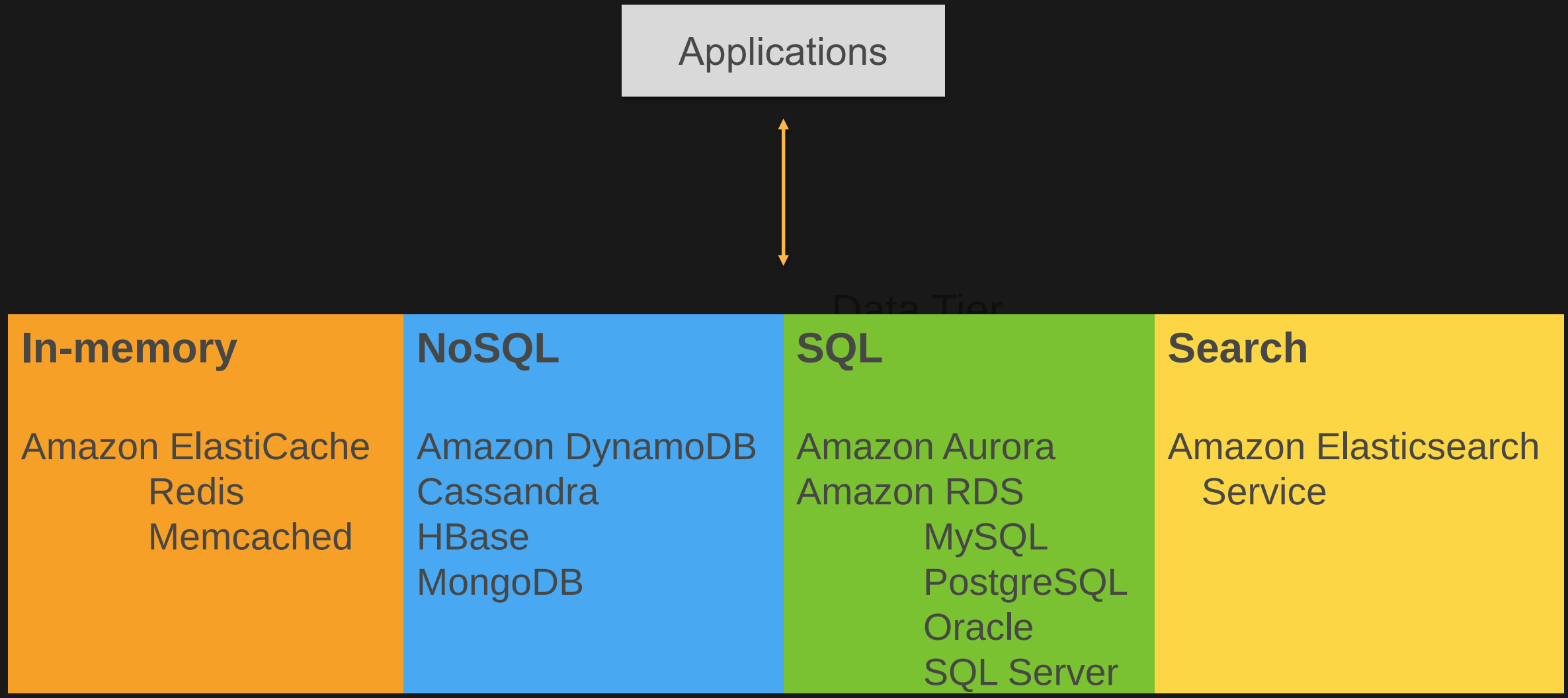


In-memory, Database, Search

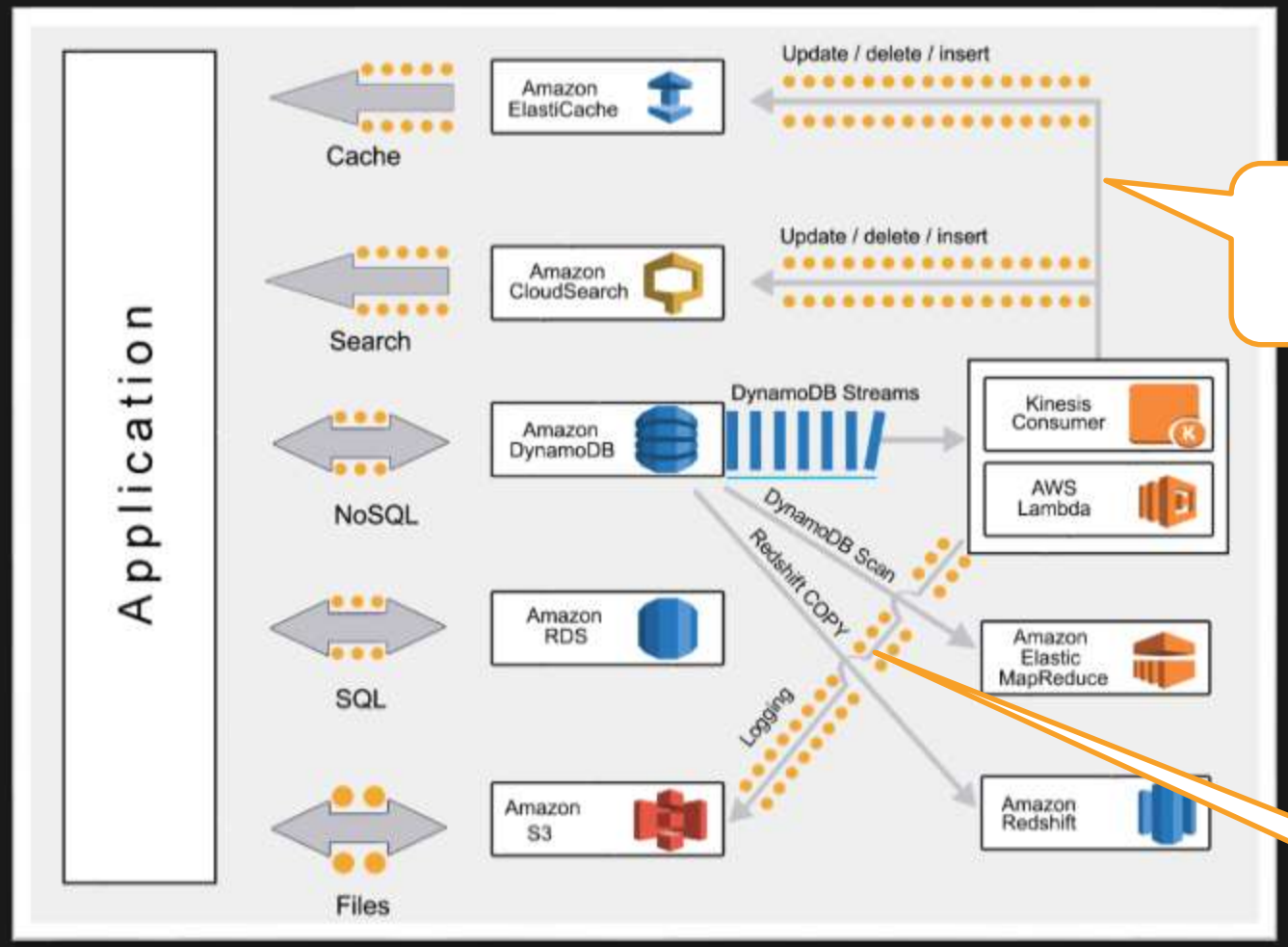
Anti-Pattern

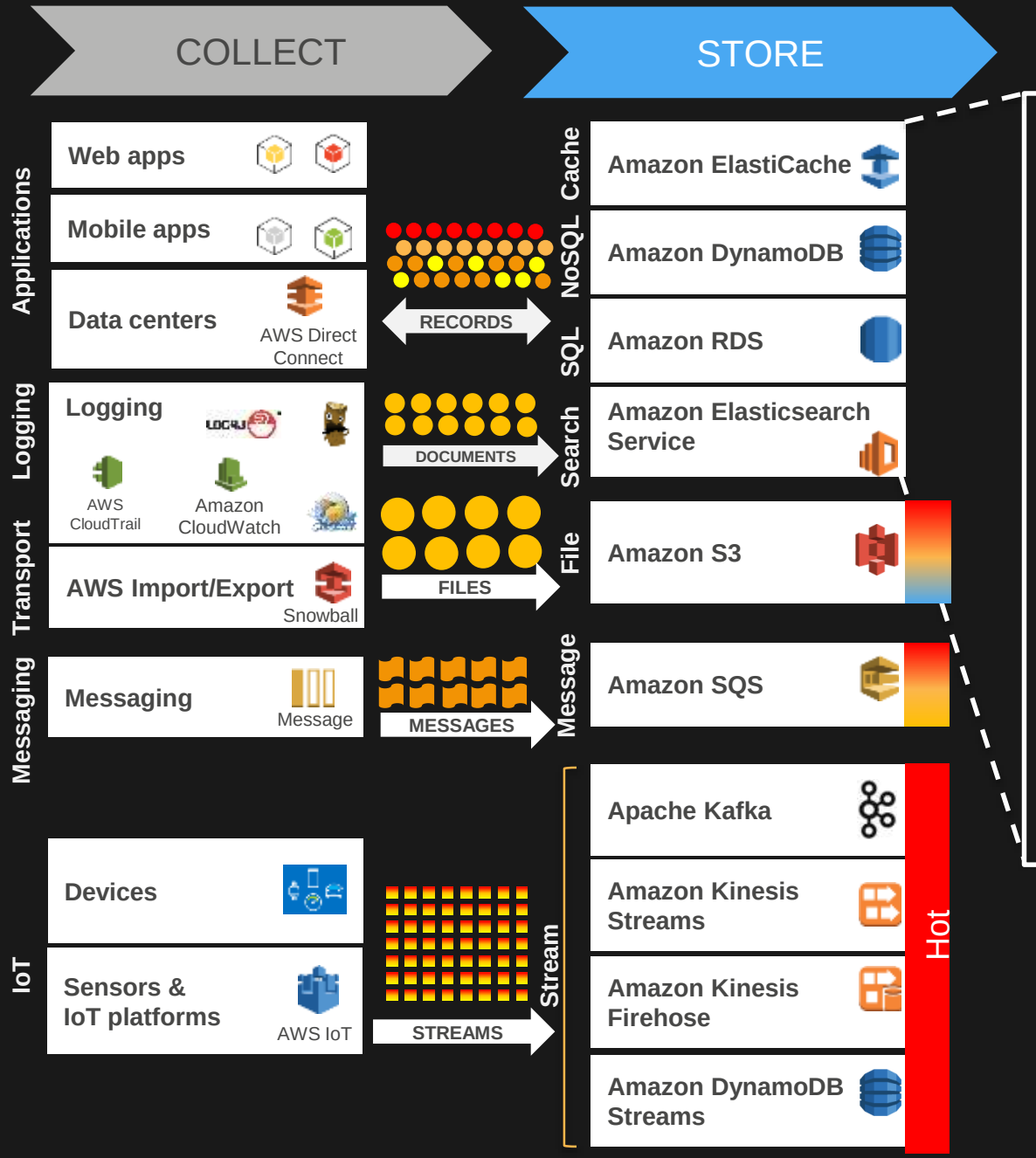


Best Practice: Use the Right Tool for the Job



Materialized Views & Immutable Log





Amazon ElastiCache

- Managed Memcached or Redis service

Amazon DynamoDB

- Managed NoSQL database service

Amazon RDS

- Managed relational database service

Amazon Elasticsearch Service

- Managed Elasticsearch service

Which Data Store Should I Use?

Data structure → Fixed schema, JSON, key-value

Access patterns → Store data in the format you will access it

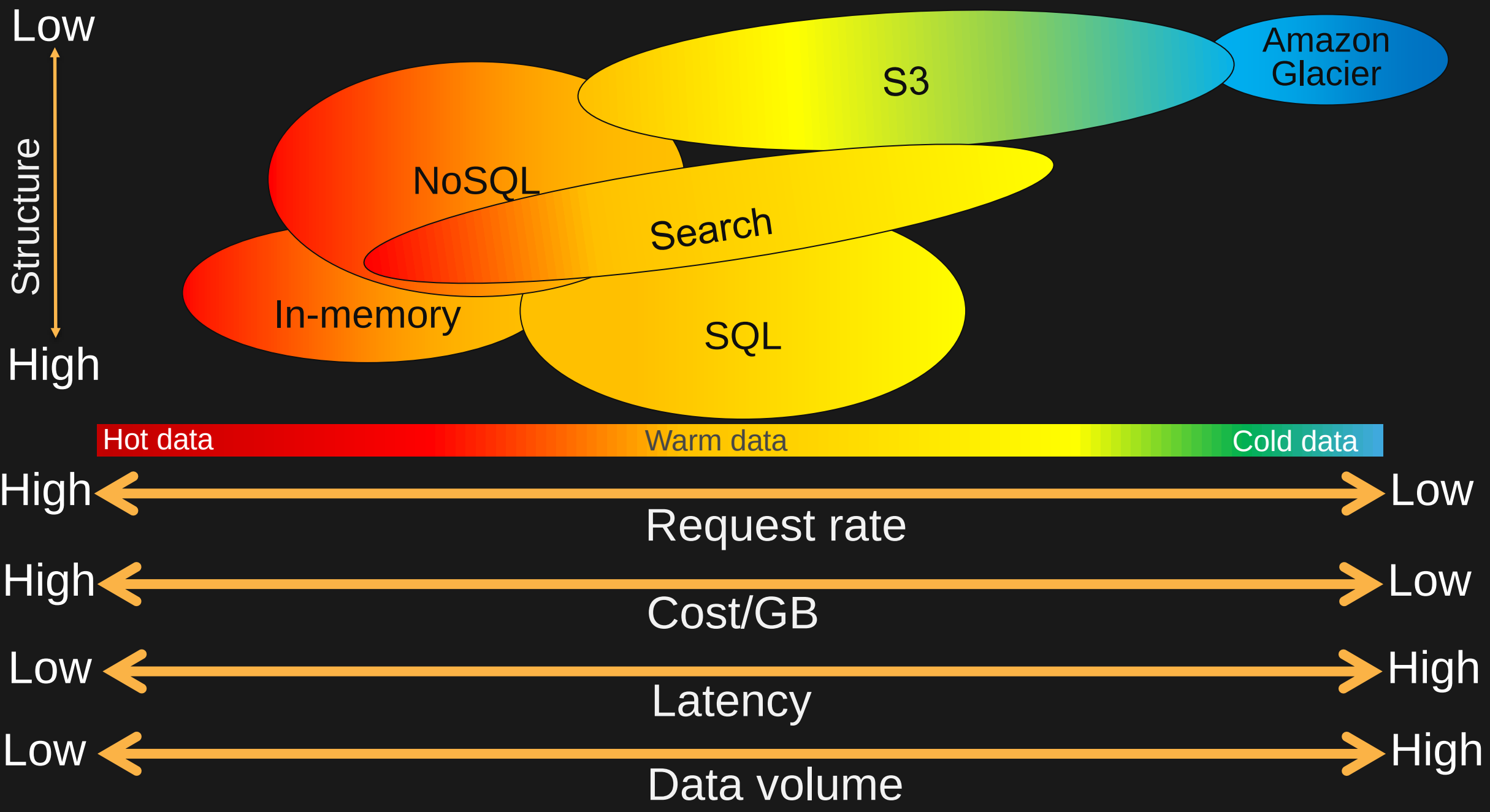
Data characteristics → Hot, warm, cold

Cost → Right cost

Data Structure and Access Patterns

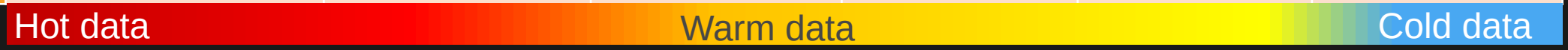
Access Patterns	What to use?
Put/Get (key, value)	In-memory, NoSQL
Simple relationships → 1:N, M:N	NoSQL
Multi-table joins, transaction, SQL	SQL
Faceting, search	Search

Data Structure	What to use?
Fixed schema	SQL, NoSQL
Schema-free (JSON)	NoSQL, Search
(Key, value)	In-memory, NoSQL


Low
Structure
High
S3
Amazon Glacier
NoSQL
Search
In-memory
SQL
Hot data
Warm data
Cold data
High
Low
Request rate
High
Low
Cost/GB
Low
High
Latency
Low
High
Data volume

Which Data Store Should I Use?

	Amazon ElastiCache	Amazon DynamoDB	Amazon RDS/Aurora	Amazon ES	Amazon S3	Amazon Glacier
Average latency	ms	ms	ms, sec	ms,sec	ms,sec,min (~ size)	hrs
Typical data stored	GB	GB–TBs (no limit)	GB–TB (64 TB max)	GB–TB	MB–PB (no limit)	GB–PB (no limit)
Typical item size	B-KB	KB (400 KB max)	KB (64 KB max)	B-KB (2 GB max)	KB-TB (5 TB max)	GB (40 TB max)
Request Rate	High – very high	Very high (no limit)	High	High	Low – high (no limit)	Very low
Storage cost GB/month	$$	¢¢	¢¢	¢¢	¢	¢4/10
Durability	Low - moderate	Very high	Very high	High	Very high	Very high
Availability	High 2 AZ	Very high 3 AZ	Very high 3 AZ	High 2 AZ	Very high 3 AZ	Very high 3 AZ



Cost-Conscious Design

Example: Should I use Amazon S3 or Amazon DynamoDB?

"I'm currently scoping out a project. The design calls for **many small files**, perhaps up to a **billion during peak**. The **total size** would be on the order of **1.5 TB per month**…"

Request rate (Writes/sec)	Object size (Bytes)	Total size (GB/month)	Objects per month
300	2048	1483	777,600,000

Cost-Conscious Design

Example: Should I use Amazon S3 or Amazon DynamoDB?

Simple Monthly Calculator

https://calculator.s3.amazonaws.com/index.html

Amazon S3 or DynamoDB?

Request rate (Writes/sec)	Object size (Bytes)	Total size (GB/month)	Objects per month
300	2,048	1,483	777,600,000

Amazon DynamoDB is a high performance non-relational database service that is easy to set up, operate, and scale. It is designed to address the core problems of database management, performance, scalability, and reliability. It also provides predictable high performance and low latency at scale.

Indexed Data Storage:

Dataset Size: 1483 GB

Provisioned Throughput Capacity *:

Item Size (All attributes): 2 KB

Number of items read per second: 0 Reads/Second

Read Consistency: Strongly Consistent / Eventually Cons cheaper)

Number of items written per second: 300 Writes/Second

Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	

Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Storage:

Storage: 1483 GB

Reduced Redundancy Storage: 0 GB

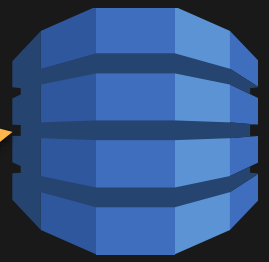
Requests:

PUT/COPY/POST/LIST Requests: 77760000 Requests

GET and Other Requests: 0 Requests

Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	

amazon web services SIMPLE MONTHLY CALCULATOR



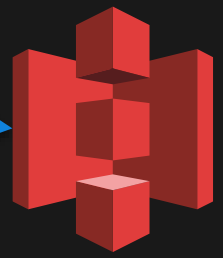
Amazon DynamoDB

use

Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	
DynamoDB Streams:	$ 0.00	

	Request rate (Writes/sec)	Object size (Bytes)	Total size (GB/month)	Objects per month
Scenario 1	300	2,048	1,483	777,600,000
Scenario 2	300	32,768	23,730	777,600,000

use



Amazon S3

Amazon S3 Service (US-East)		$ 4588.55
Storage:	$ 700.55	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 10131.40
Provisioned Throughput Capacity:	$ 4187.04	
Indexed Data Storage:	$ 5944.36	
DynamoDB Streams:	$ 0.00	

PROCESS / ANALYZE

Analytics Types & Frameworks

Batch

Takes minutes to hours

Example: Daily/weekly/monthly reports

Amazon EMR (MapReduce, Hive, Pig, Spark)

Interactive

Takes seconds

Example: Self-service dashboards

Amazon Redshift, Amazon Athena, Amazon EMR (Presto, Spark)

Message

Takes milliseconds to seconds

Example: Message processing

Amazon SQS applications on Amazon EC2

Stream

Takes milliseconds to seconds

Example: Fraud alerts, 1 minute metrics

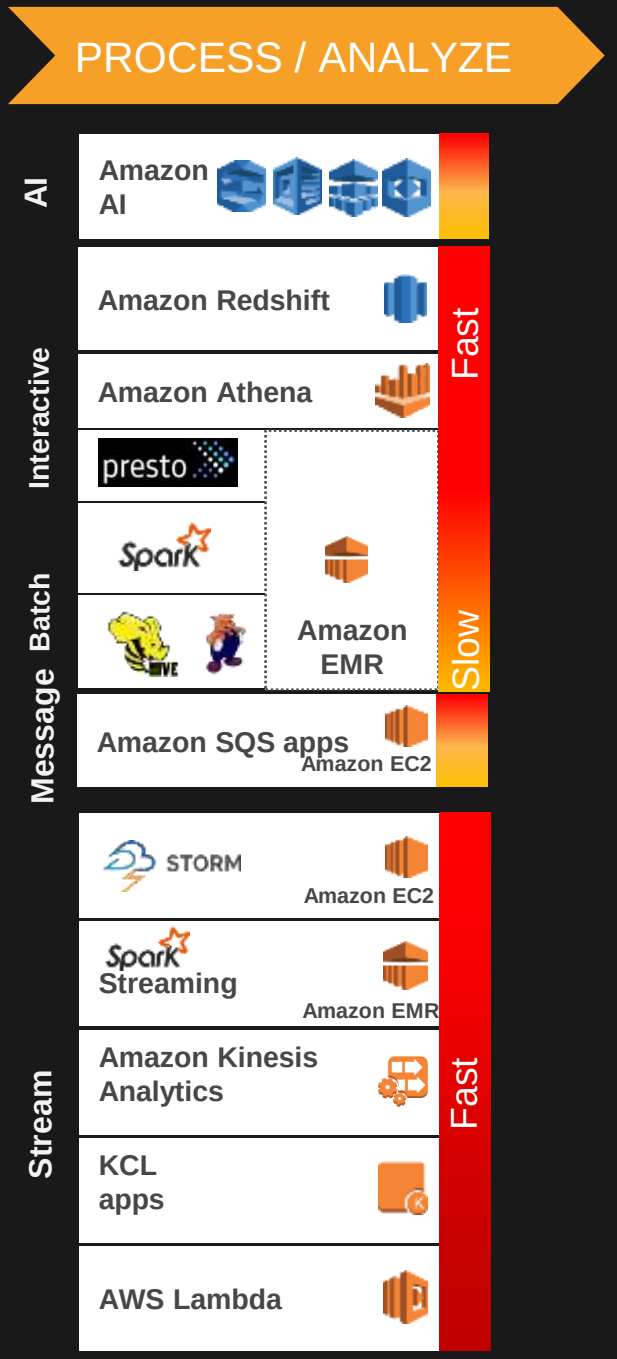
Amazon EMR (Spark Streaming), Amazon Kinesis Analytics, KCL, Storm, AWS Lambda

Artificial Intelligence

Takes milliseconds to minutes

Example: Fraud detection, forecast demand, text to speech

Amazon AI (Lex, Polly, ML, Rekognition), Amazon EMR (Spark ML), Deep Learning AMI (MXNet, TensorFlow, Theano, Torch, CNTK and Caffe)



Which Stream & Message Processing Technology Should I Use?

	Amazon EMR (Spark Streaming)	Apache Storm	KCL Application	Amazon Kinesis Analytics	AWS Lambda	Amazon SQS Application
AWS managed	Yes (Amazon EMR)	No (Do it yourself)	No (EC2 + Auto Scaling)	Yes	Yes	No (EC2 + Auto Scaling)
Serverless	No	No	No	Yes	Yes	No
Scale / throughput	No limits / ~ nodes	No limits / ~ nodes	No limits / ~ nodes	Up to 8 KPU / automatic	No limits / automatic	No limits / ~ nodes
Availability	Single AZ	Configurable	Multi-AZ	Multi-AZ	Multi-AZ	Multi-AZ
Programming languages	Java, Python, Scala	Almost any language via Thrift	Java, others via MultiLangDaemon	ANSI SQL with extensions	Node.js, Java, Python	AWS SDK languages (Java, .NET, Python, …)
Uses	Multistage processing	Multistage processing	Single stage processing	Multistage processing	Simple event-based triggers	Simple event based triggers
Reliability	KCL and Spark checkpoints	Framework managed	Managed by KCL	Managed by Amazon Kinesis Analytics	Managed by AWS Lambda	Managed by SQS Visibility Timeout

Fast

Which Analysis Tool Should I Use?

	Amazon Redshift	Amazon Athena	Amazon EMR		
			Presto	Spark	Hive
Use case	Optimized for data warehousing	Ad-hoc Interactive Queries	Interactive Query	General purpose (iterative ML, RT, ..)	Batch
Scale/throughput	~Nodes	Automatic / No limits	~ Nodes		
AWS Managed Service	Yes	Yes, Serverless	Yes		
Storage	Local storage	Amazon S3	Amazon S3, HDFS		
Optimization	Columnar storage, data compression, and zone maps	CSV, TSV, JSON, Parquet, ORC, Apache Web log	Framework dependent		
Metadata	Amazon Redshift managed	Athena Catalog Manager	Hive Meta-store		
BI tools supports	Yes (JDBC/ODBC)	Yes (JDBC)	Yes (JDBC/ODBC & Custom)		
Access controls	Users, groups, and access controls	AWS IAM	Integration with LDAP		
UDF support	Yes (Scalar)	No	Yes		

Fast Slow

What About ETL?

Data Integration Partners

Reduce the effort to move, cleanse, synchronize, manage, and automatize data related processes.

alteryx | ATTUNITY | informatica | ironSource .atom

MATILLION | snapLogic | bryte | talend

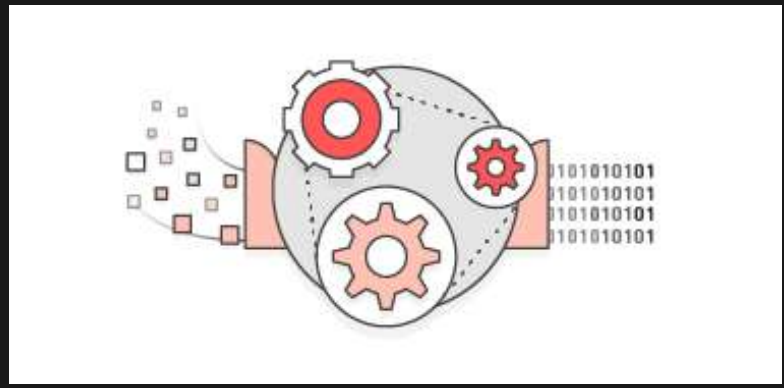
alooma

• • • • •

https://aws.amazon.com/big-data/partner-solutions/

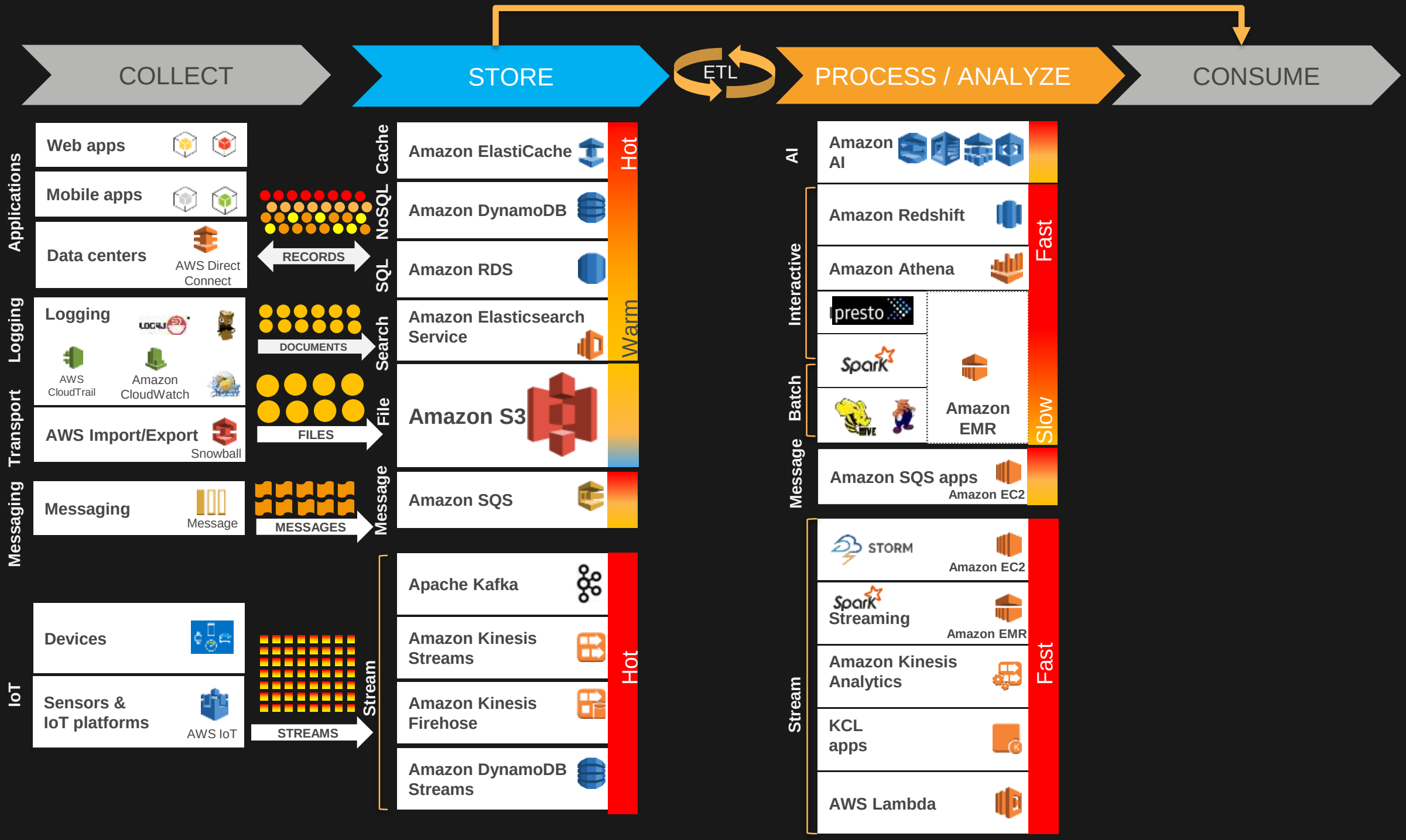
New

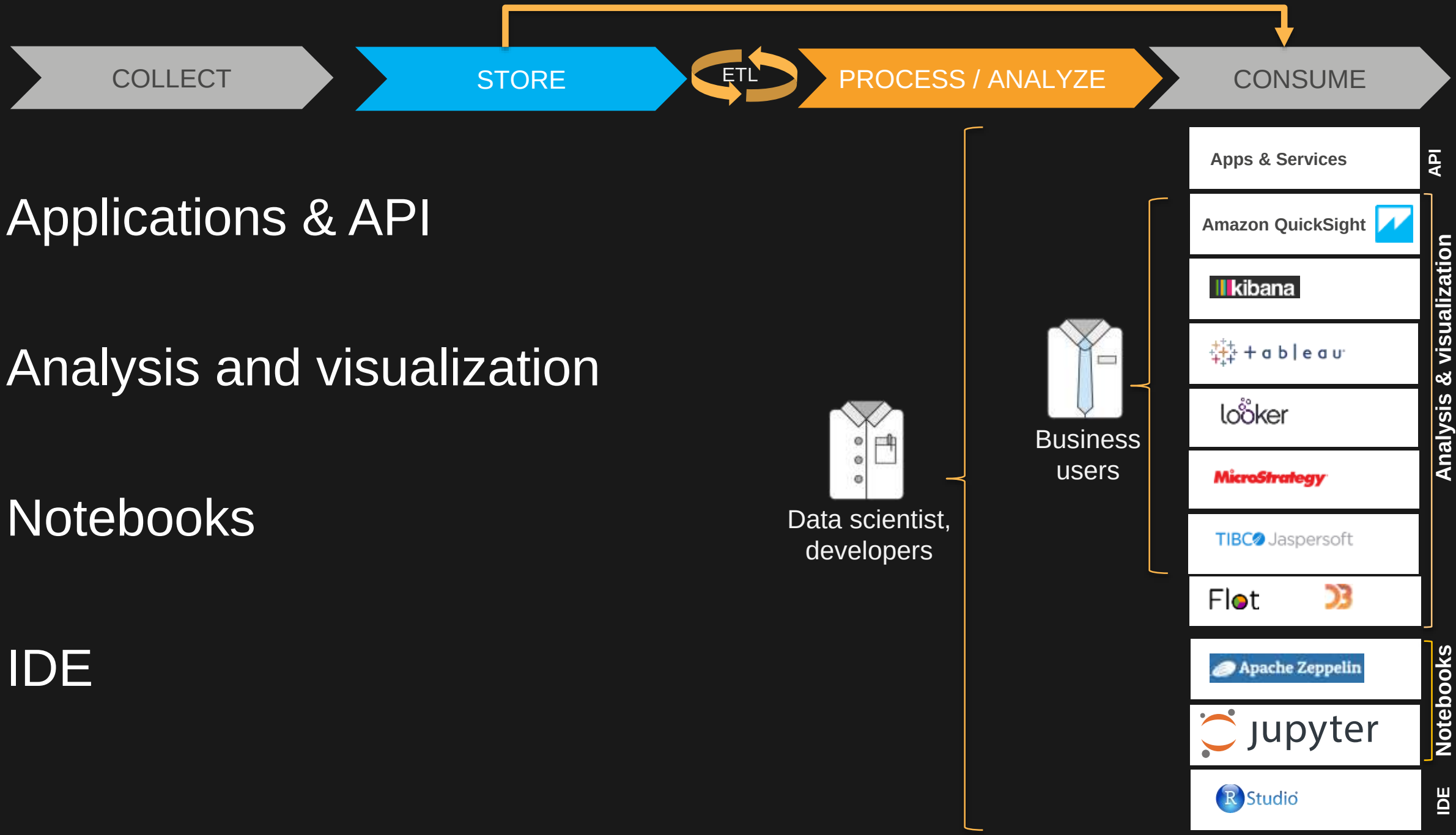
AWS Glue



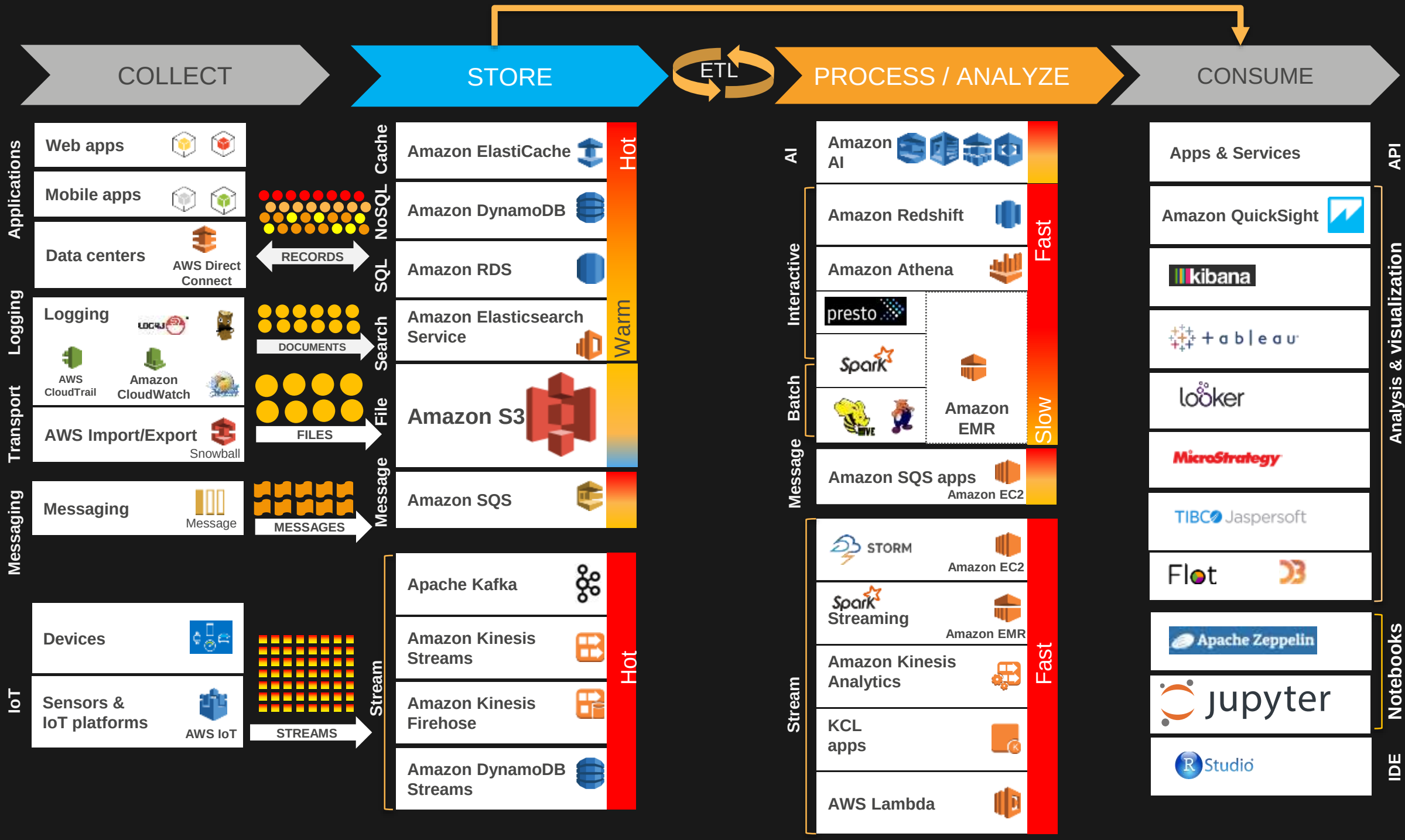
AWS Glue is a fully managed ETL service that makes it easy to understand your data sources, prepare the data, and move it reliably between data stores

CONSUME

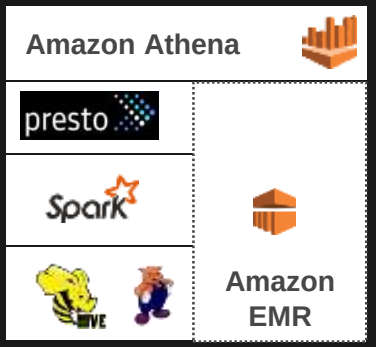

COLLECT
STORE
ETL
PROCESS / ANALYZE
CONSUME
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
Transport
AWS Import/Export
Snowball
Messaging
Messaging
Message
IoT
Devices
Sensors & IoT platforms
AWS IoT
RECORDS
DOCUMENTS
FILES
MESSAGES
STREAMS
Cache
NoSQL
SQL
Search
File
Message
Stream
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon Elasticsearch Service
Amazon S3
Amazon SQS
Apache Kafka
Amazon Kinesis Streams
Amazon Kinesis Firehose
Amazon DynamoDB Streams
Hot
Warm
Hot
AI
Amazon AI
Interactive
Amazon Redshift
Amazon Athena
presto
Spark
Batch
Amazon EMR
Message
Amazon SQS apps
Amazon EC2
Stream
STORM
Amazon EC2
Spark Streaming
Amazon EMR
Amazon Kinesis Analytics
KCL apps
AWS Lambda
Fast
Slow
Fast


COLLECT
STORE
ETL
PROCESS / ANALYZE
CONSUME
Applications & API
Analysis and visualization
Notebooks
IDE
Data scientist, developers
Business users
Apps & Services
Amazon QuickSight
kibana
tableau
looker
MicroStrategy
TIBCO Jaspersoft
Flot
Apache Zeppelin
jupyter
RStudio
API
Analysis & visualization
Notebooks
IDE

Putting It All Together


COLLECT
STORE
ETL
PROCESS / ANALYZE
CONSUME
Applications
Web apps
Mobile apps
Data centers
AWS Direct Connect
Logging
Logging
AWS CloudTrail
Amazon CloudWatch
Transport
AWS Import/Export
Snowball
Messaging
Messaging
Message
IoT
Devices
Sensors & IoT platforms
AWS IoT
RECORDS
DOCUMENTS
FILES
MESSAGES
STREAMS
Cache
NoSQL
SQL
Search
File
Message
Stream
Hot
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon Elasticsearch Service
Warm
Amazon S3
Amazon SQS
Apache Kafka
Amazon Kinesis Streams
Amazon Kinesis Firehose
Amazon DynamoDB Streams
AI
Amazon AI
Interactive
Batch
Message
Stream
Amazon Redshift
Amazon Athena
presto
Spark
Amazon EMR
Fast
Slow
Amazon SQS apps
Amazon EC2
STORM
Amazon EC2
Spark Streaming
Amazon EMR
Amazon Kinesis Analytics
KCL apps
AWS Lambda
Apps & Services
API
Amazon QuickSight
kibana
tableau
looker
MicroStrategy
TIBCO Jaspersoft
Flot
Analysis & visualization
Apache Zeppelin
jupyter
Notebooks
RStudio
IDE

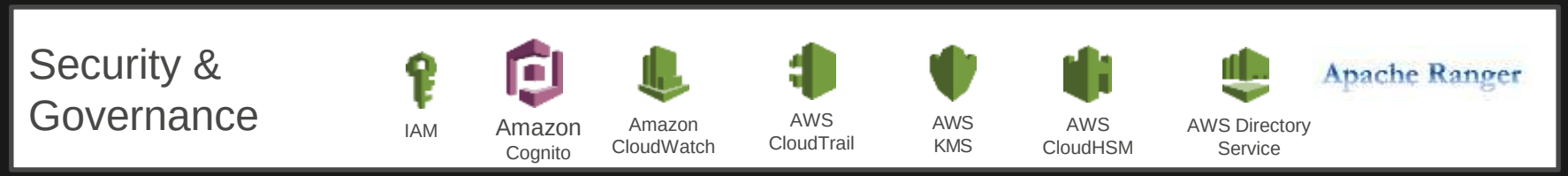
What about Metadata?



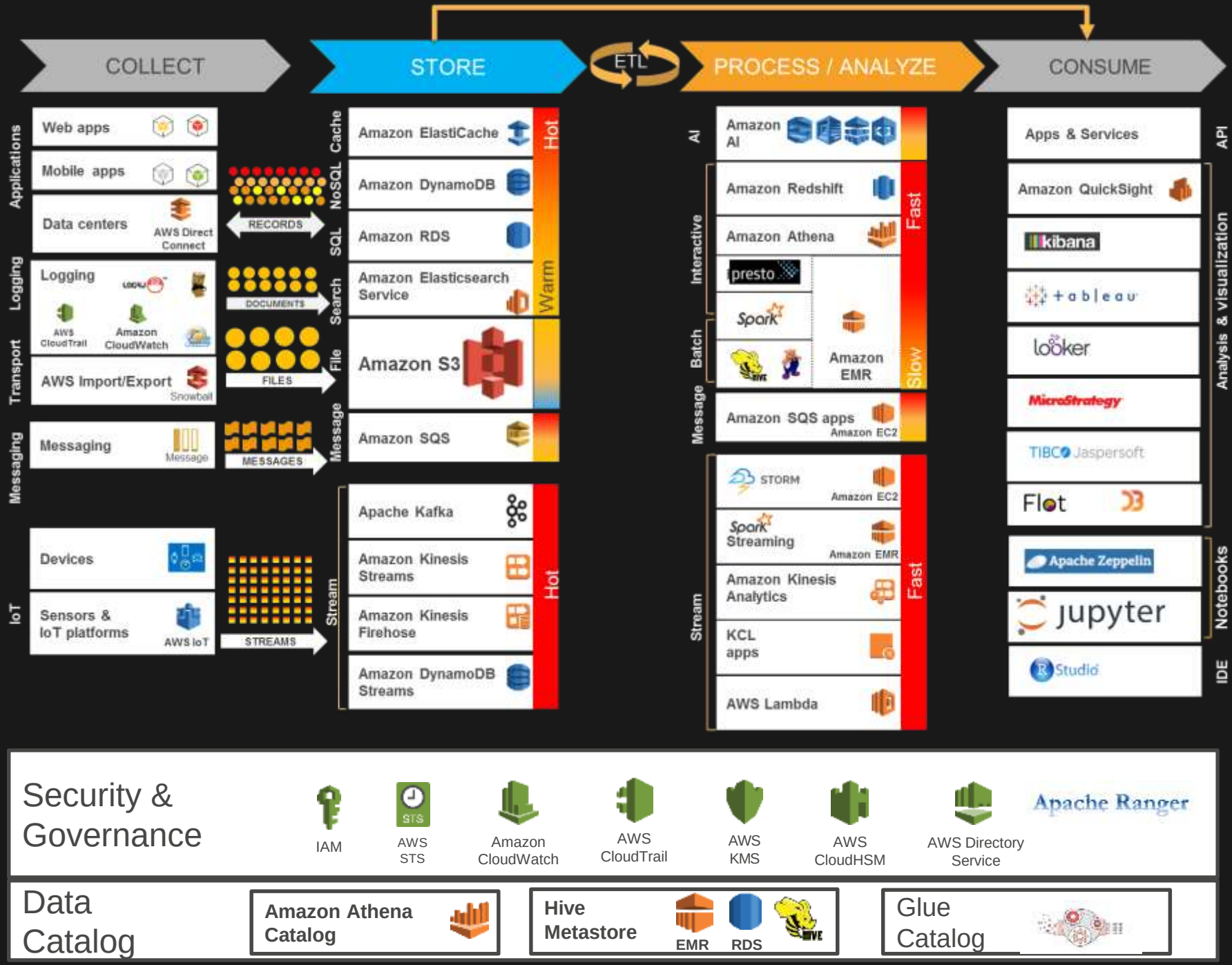
- Amazon Athena Catalog
  - An internal data catalog for tables/schemas on S3
- Glue Catalog
  - Hive Metastore compliant
  - Crawlers - Detect new data, schema, partitions
  - Search - Metadata discovery
- EMR Hive Metastore (Presto, Spark, Hive, Pig)
  - Can be hosted on Amazon RDS

Security & Governance

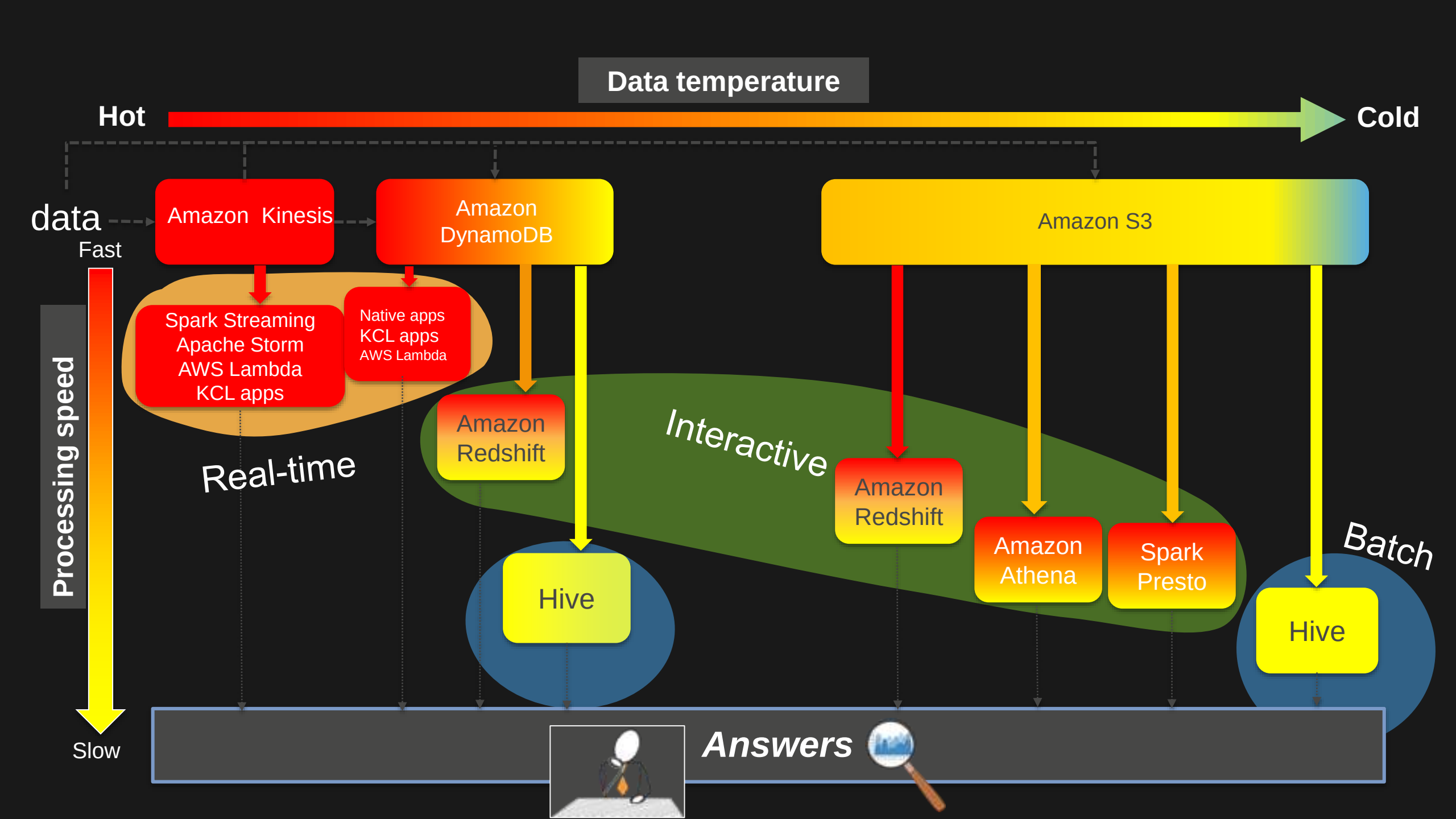
- AWS Identity and Access Management (IAM)
- Amazon Cognito
- Amazon CloudWatch & AWS CloudTrail
- Amazon KMS
- AWS Directory Service
- Apache Ranger



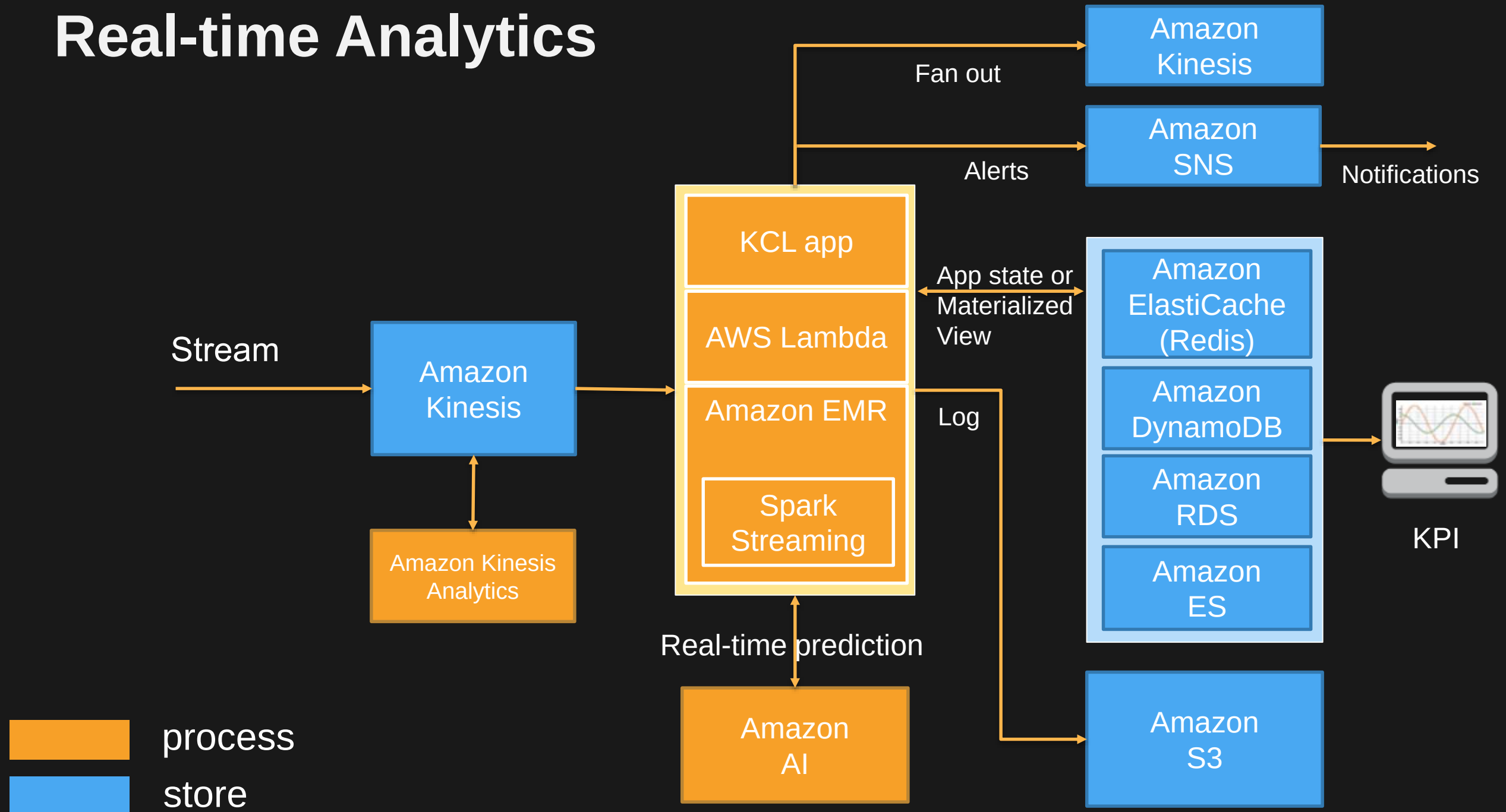
Data Lake Reference Architecture



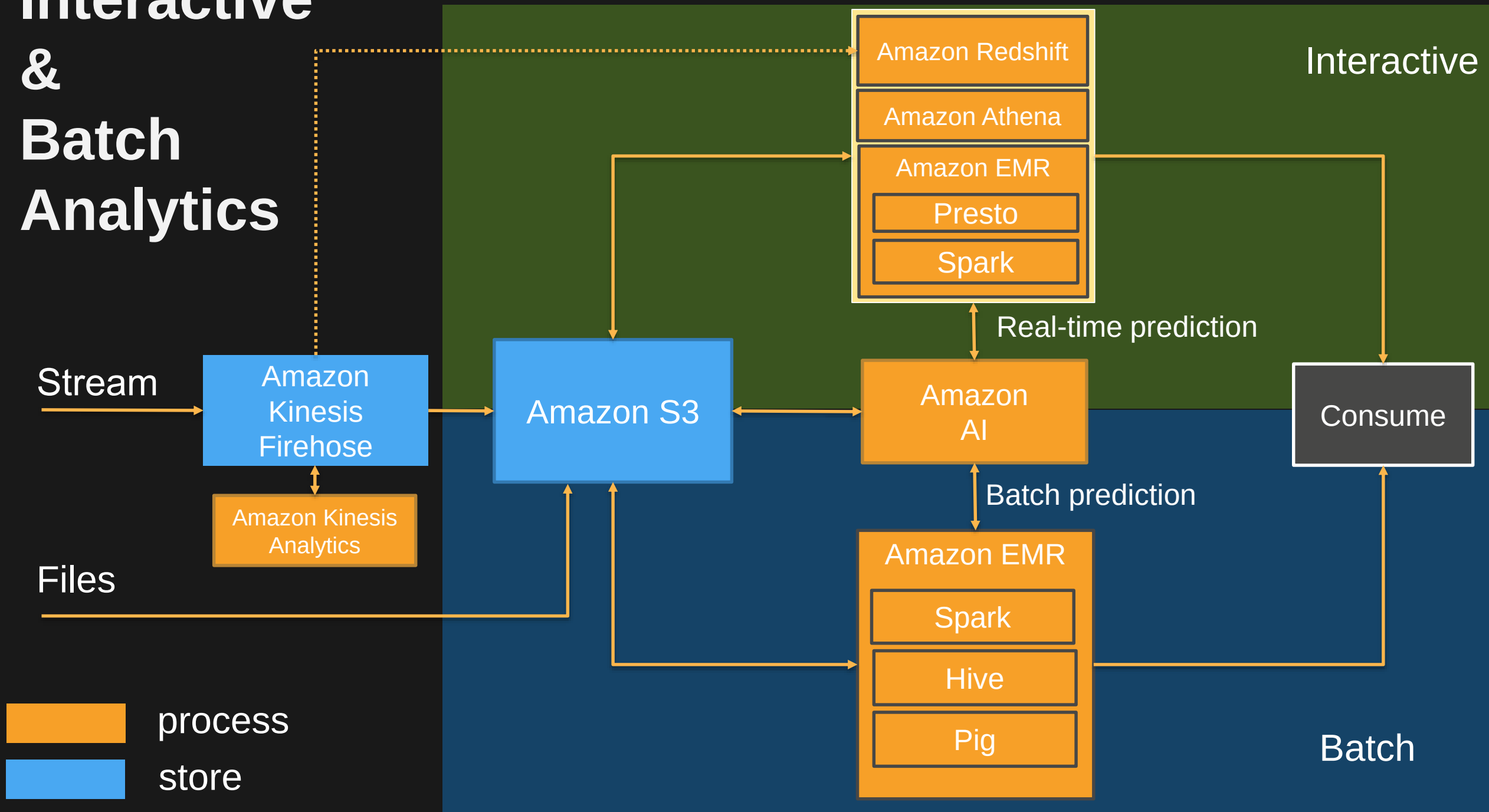
Design Patterns

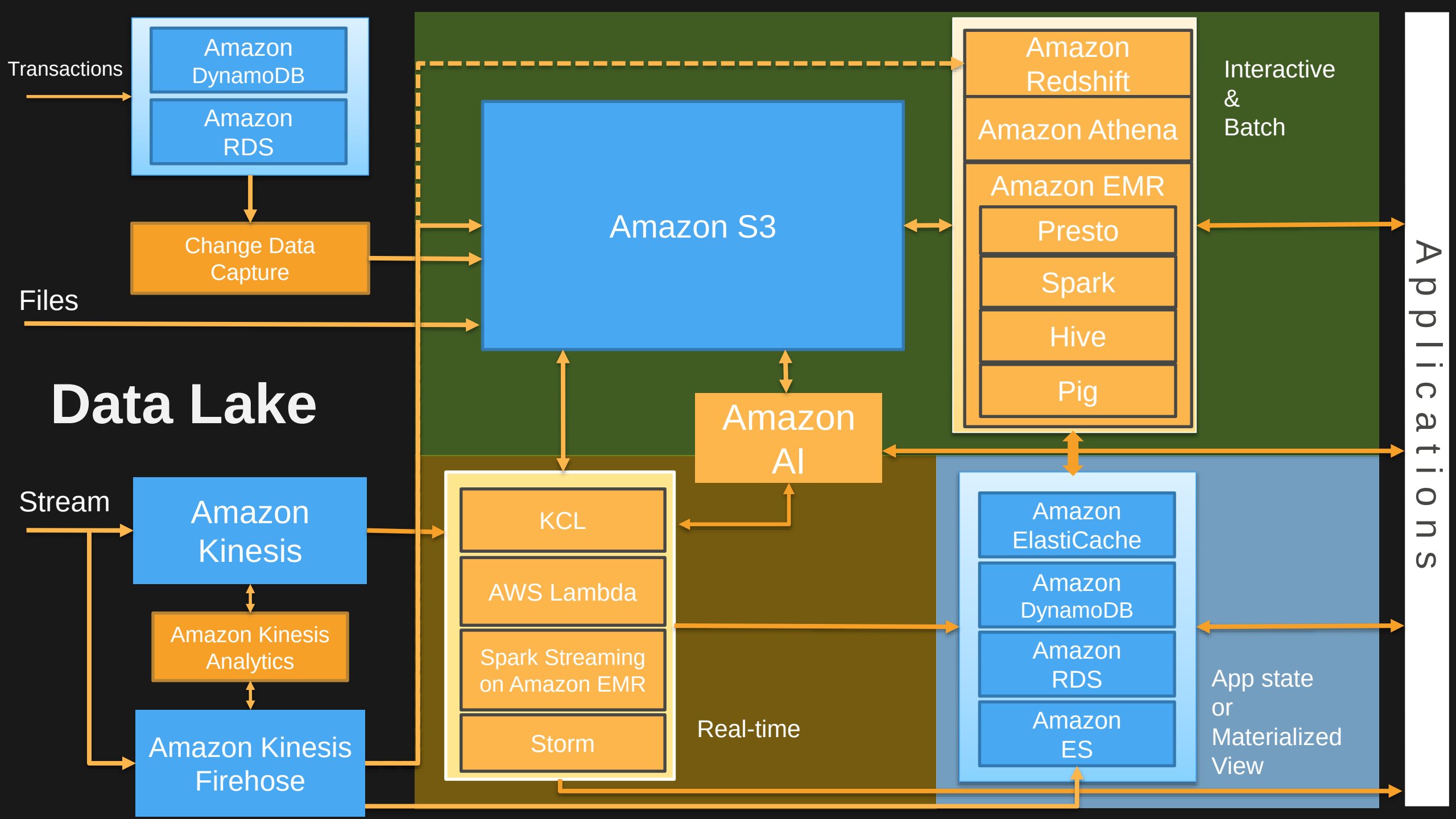

Data temperature
Hot
Cold
data
Fast
Amazon Kinesis
Amazon DynamoDB
Amazon S3
Spark Streaming
Apache Storm
AWS Lambda
KCL apps
Native apps
KCL apps
AWS Lambda
Amazon Redshift
Amazon Redshift
Amazon Athena
Spark Presto
Hive
Hive
Real-time
Interactive
Batch
Processing speed
Slow
Answers

Real-time Analytics



Interactive & Batch Analytics




Transactions
Amazon DynamoDB
Amazon RDS
Change Data Capture
Files
Data Lake
Stream
Amazon Kinesis
Amazon Kinesis Analytics
Amazon Kinesis Firehose
Amazon S3
Amazon AI
KCL
AWS Lambda
Spark Streaming on Amazon EMR
Storm
Real-time
Amazon Redshift
Amazon Athena
Amazon EMR
Presto
Spark
Hive
Pig
Interactive & Batch
Amazon ElastiCache
Amazon DynamoDB
Amazon RDS
Amazon ES
App state or Materialized View
Applications

Summary

Build decoupled systems

- Data → Store → Process → Store → Analyze → Answers

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Leverage AWS managed services

- Scalable/elastic, available, reliable, secure, no/low admin

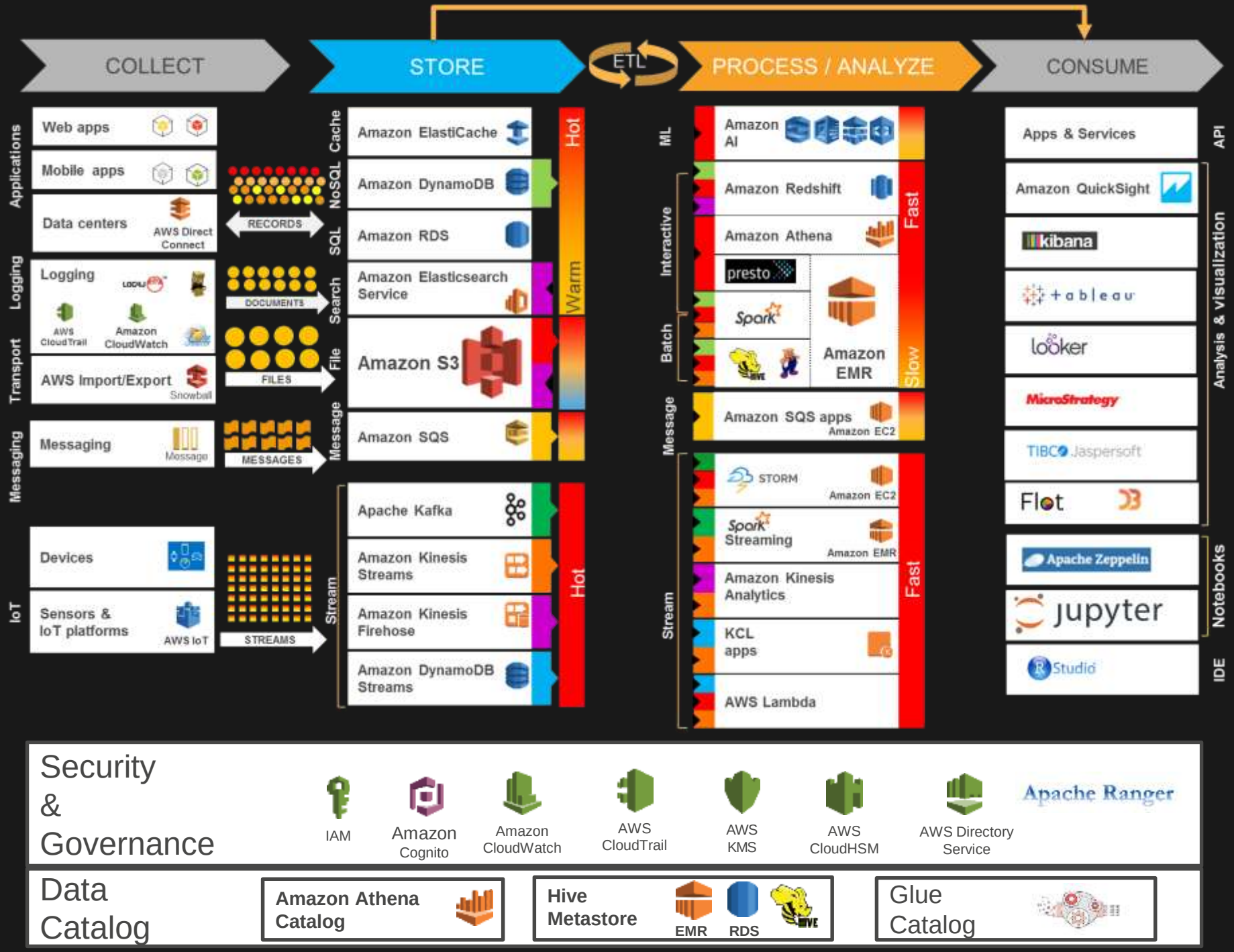
Use log-centric design patterns

- Immutable log, batch, interactive & real-time views

Be cost-conscious

- Big data ≠ big cost

Data Lake Reference Architecture



Resources

- https://aws.amazon.com/blogs/big-data/introducing-the-data-lake-solution-on-aws/
- AWS re:Invent 2016: Netflix: Using Amazon S3 as the fabric of our big data ecosystem (BDM306)
- AWS re:Invent 2016: Deep Dive on Amazon S3 (STG303)
- https://aws.amazon.com/blogs/big-data/reinvent-2016-aws-big-data-machine-learning-sessions/
- https://aws.amazon.com/blogs/big-data/implementing-authorization-and-auditing-using-apache-ranger-on-amazon-emr/

Thank you!