AWS SUMMIT

Big Data Architectural Patterns and Best Practices on AWS

Siva Raghupathy, Sr. Manager, Solutions Architecture, AWS

April 2016

amazon web services

Agenda

Big data challenges

How to simplify big data processing

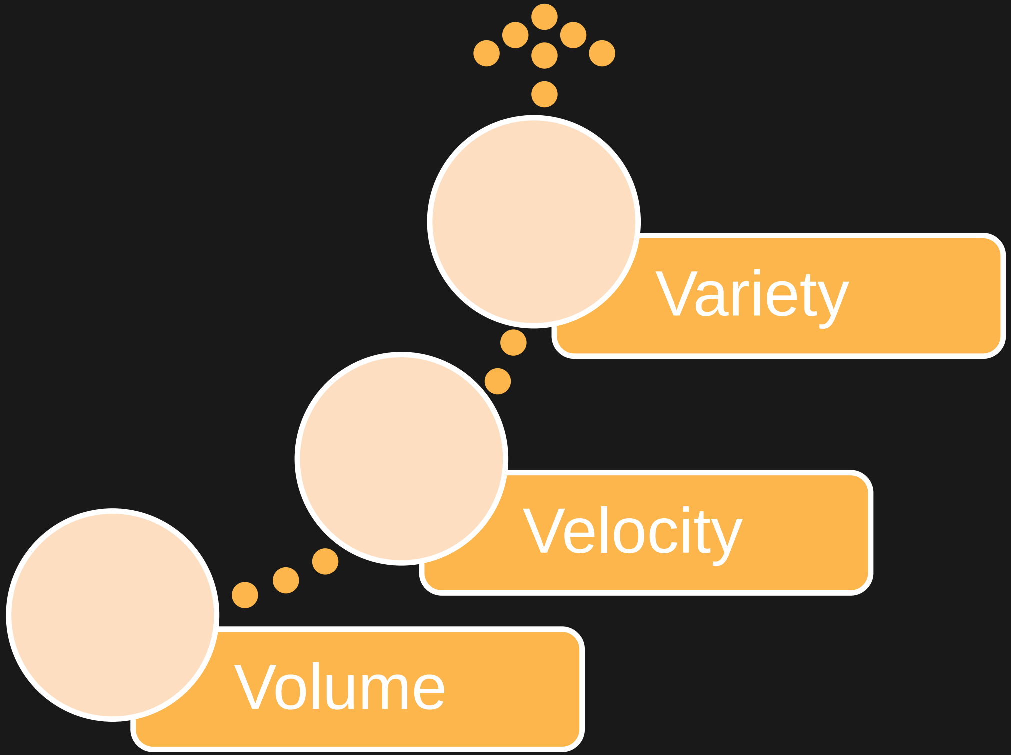
What technologies should you use?

- Why?
- How?

Reference architecture

Design patterns

Ever Increasing Big Data



Variety

Velocity

Volume

Big Data Evolution

Batch processing

Stream processing

Machine learning

Plethora of Tools

EMR

S3

DynamoDB

SQS

Amazon Redshift

Amazon Glacier

RDS

ElastiCache

Amazon Kinesis

Kinesis-enabled app

Data Pipeline

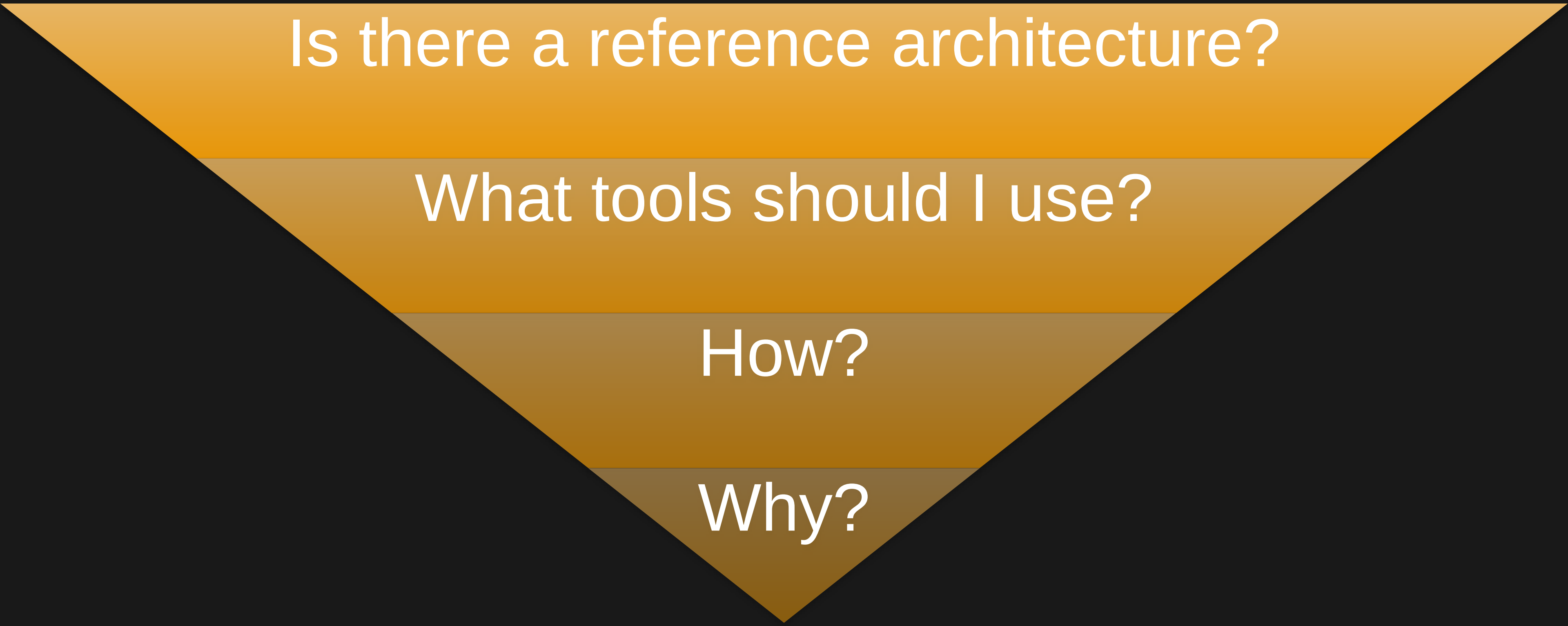
Amazon Elasticsearch Service

Lambda

ML

DynamoDB Streams

Big Data Challenges



Is there a reference architecture?

What tools should I use?

How?

Why?

Architectural Principles

Decoupled “data bus”

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

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Use Lambda architecture ideas

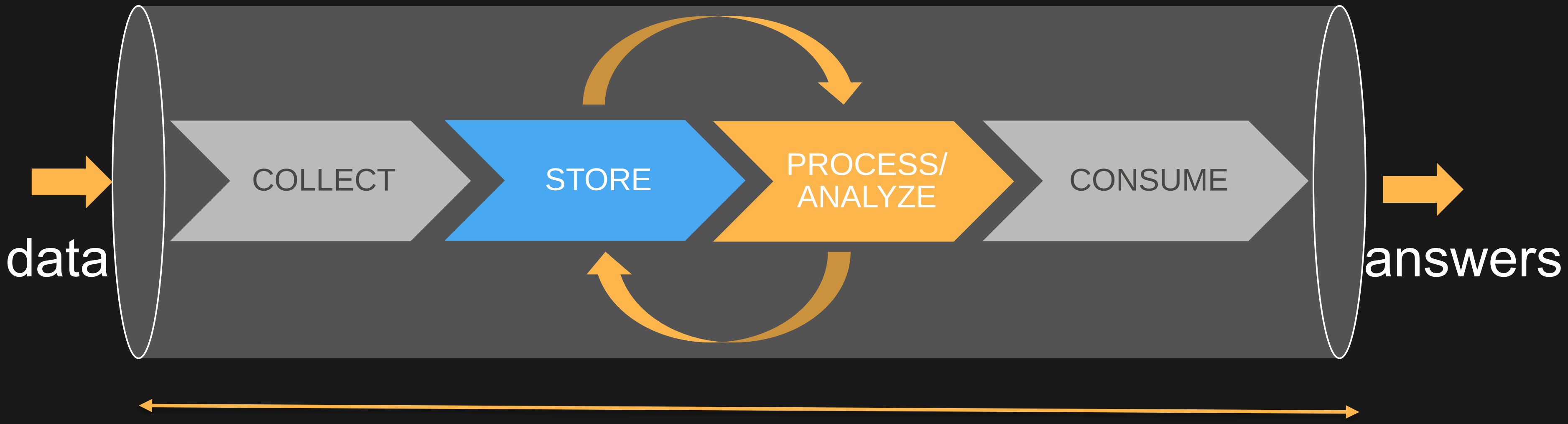
- Immutable (append-only) log, batch/speed/serving layer

Leverage AWS managed services

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

Big data ≠ big cost

Simplify Big Data Processing



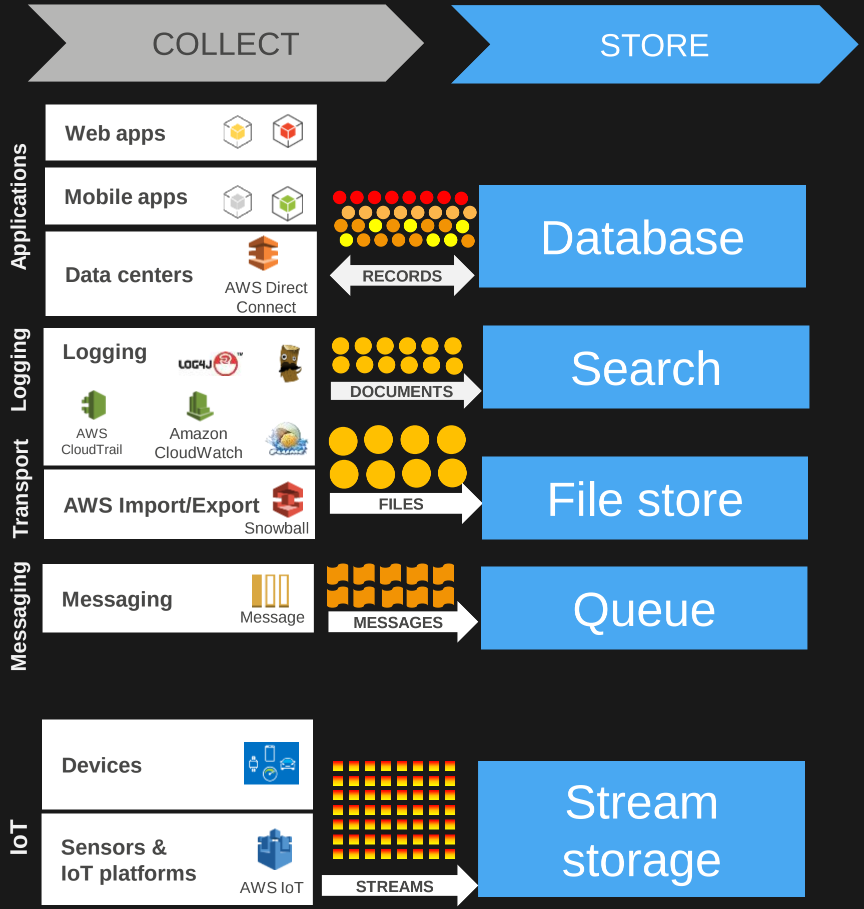
Time to answer (Latency)

Throughput

Cost

COLLECT

Types of Data



COLLECT

STORE

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

Database

Search

File store

Queue

Stream storage

Database records

Search documents

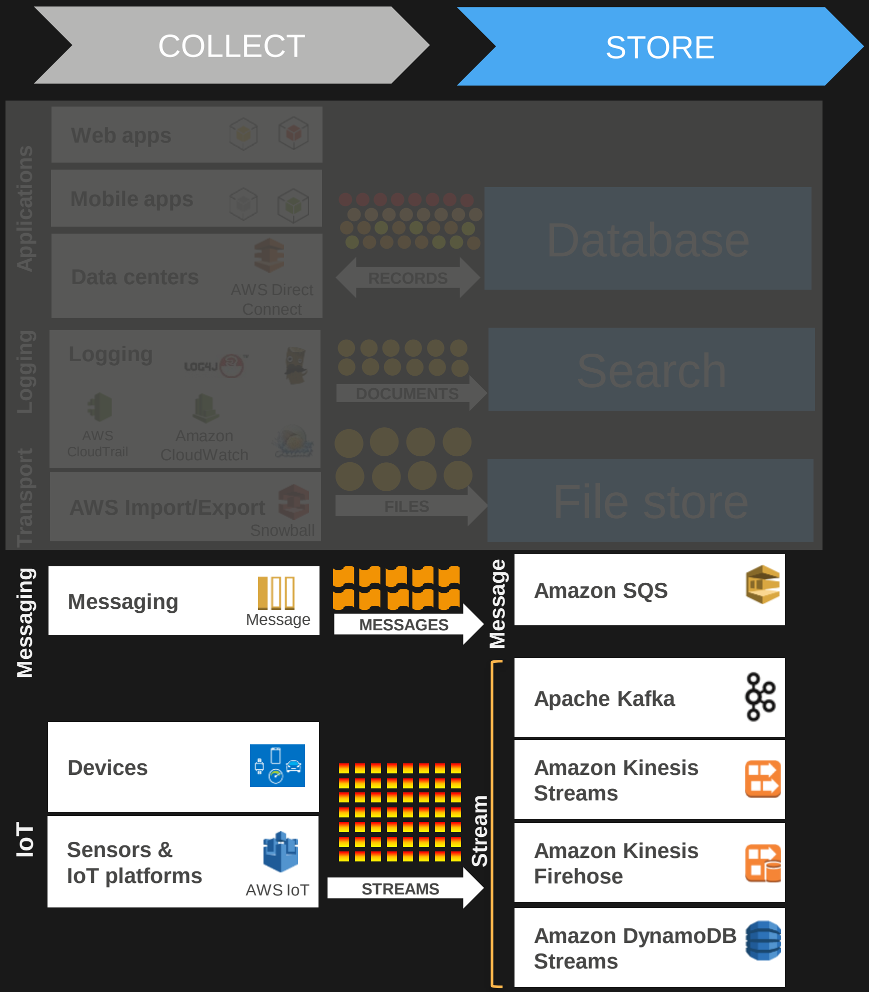
Log files

Messaging events

Devices / sensors / IoT stream

Store

Message & Stream Storage



COLLECT

STORE

Applications

Web apps

Mobile apps

Data centers

AWS Direct Connect

RECORDS

Database

Logging

Logging

AWS CloudTrail

Amazon CloudWatch

DOCUMENTS

Search

Transport

AWS Import/Export

Snowball

FILES

File store

Messaging

Messaging

Message

MESSAGES

Message

Amazon SQS

IoT

Devices

Sensors & IoT platforms

AWS IoT

STREAMS

Stream

Apache Kafka

Amazon Kinesis Streams

Amazon Kinesis Firehose

Amazon DynamoDB Streams

Amazon SQS

- Managed message queue service

Apache Kafka

- High throughput distributed messaging system

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

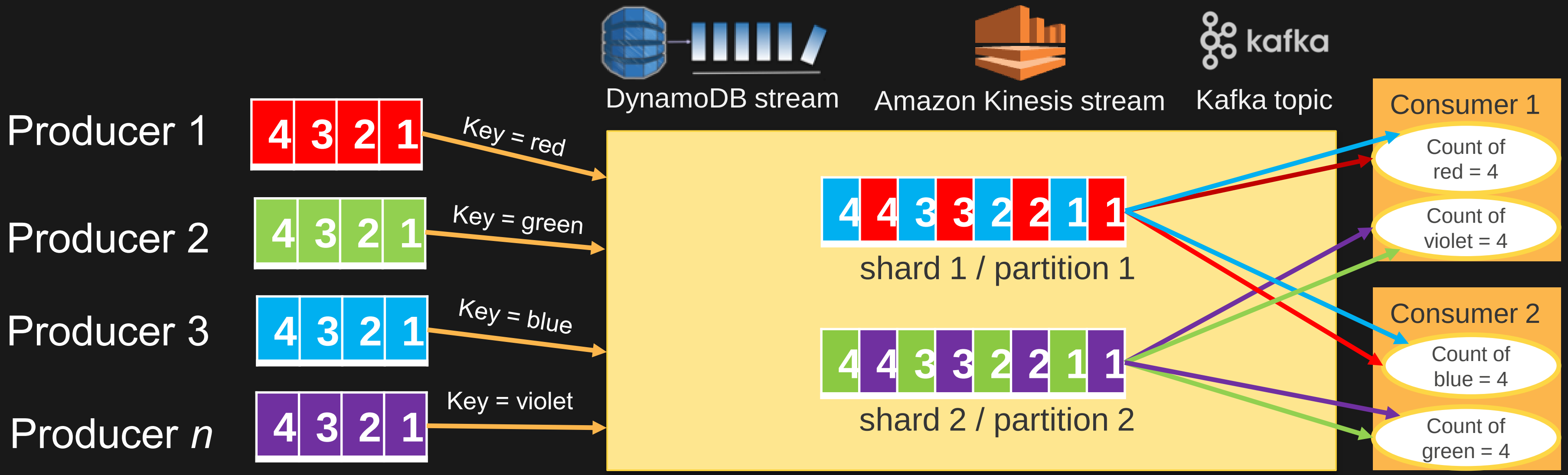
Persistent buffer

Collect multiple streams

Preserve client ordering

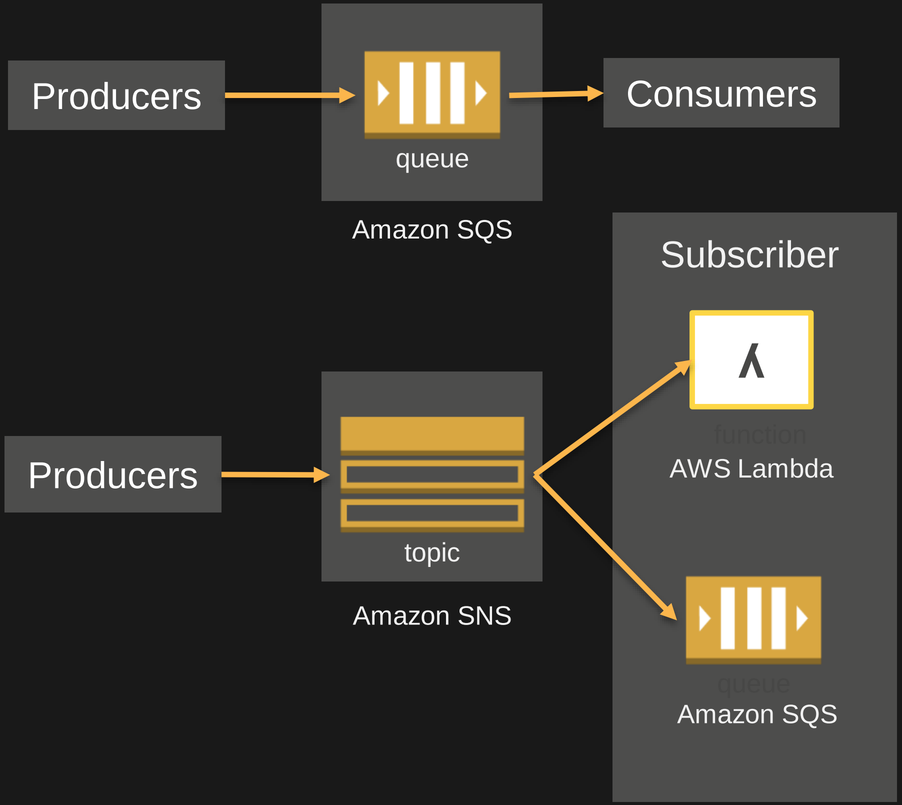
Streaming MapReduce

Parallel consumption



What About Queues & Pub/Sub ?

- Decouple producers & consumers/subscribers
- Persistent buffer
- Collect multiple streams
- **No** client ordering
- **No** parallel consumption for Amazon SQS
  - Amazon SNS can route to multiple queues or λ functions
- **No** streaming MapReduce

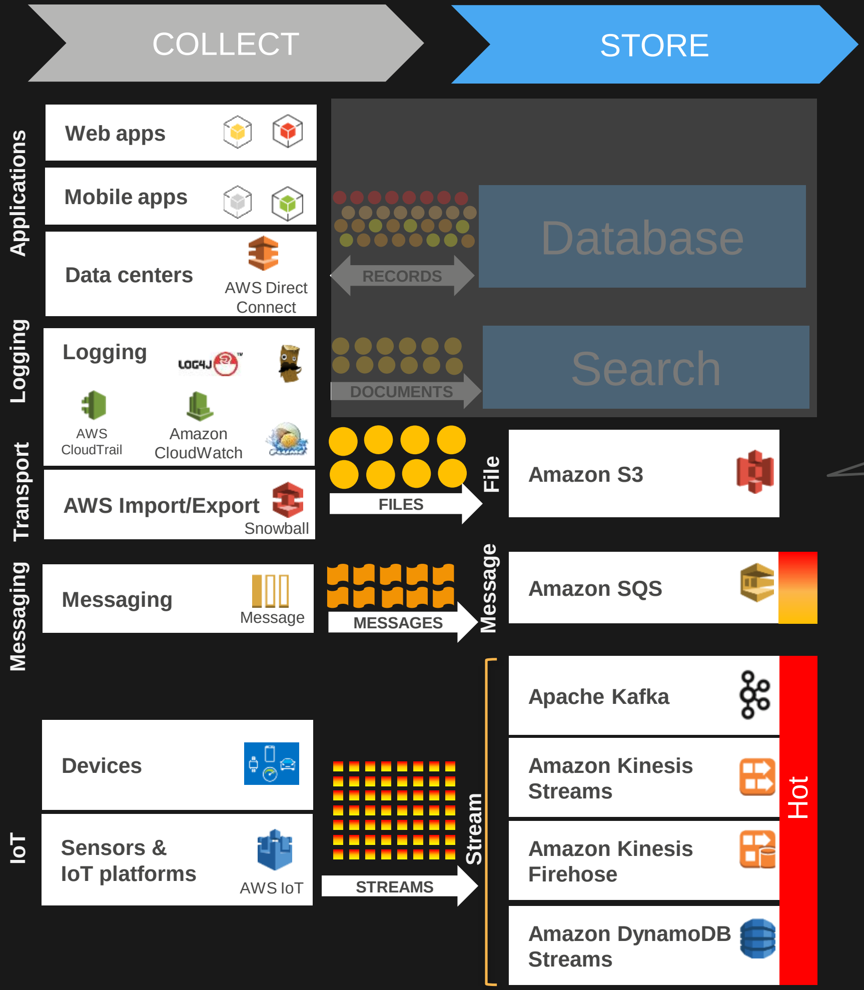


What Stream Storage should I use?

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

Hot Warm

File Storage



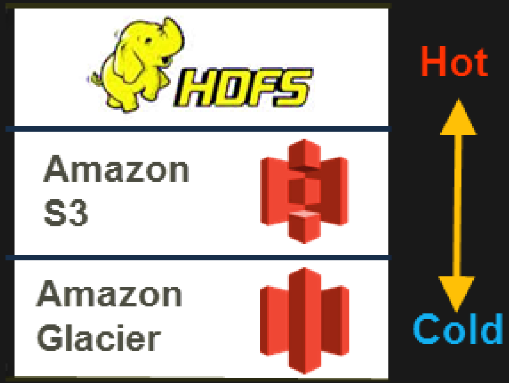
Amazon S3

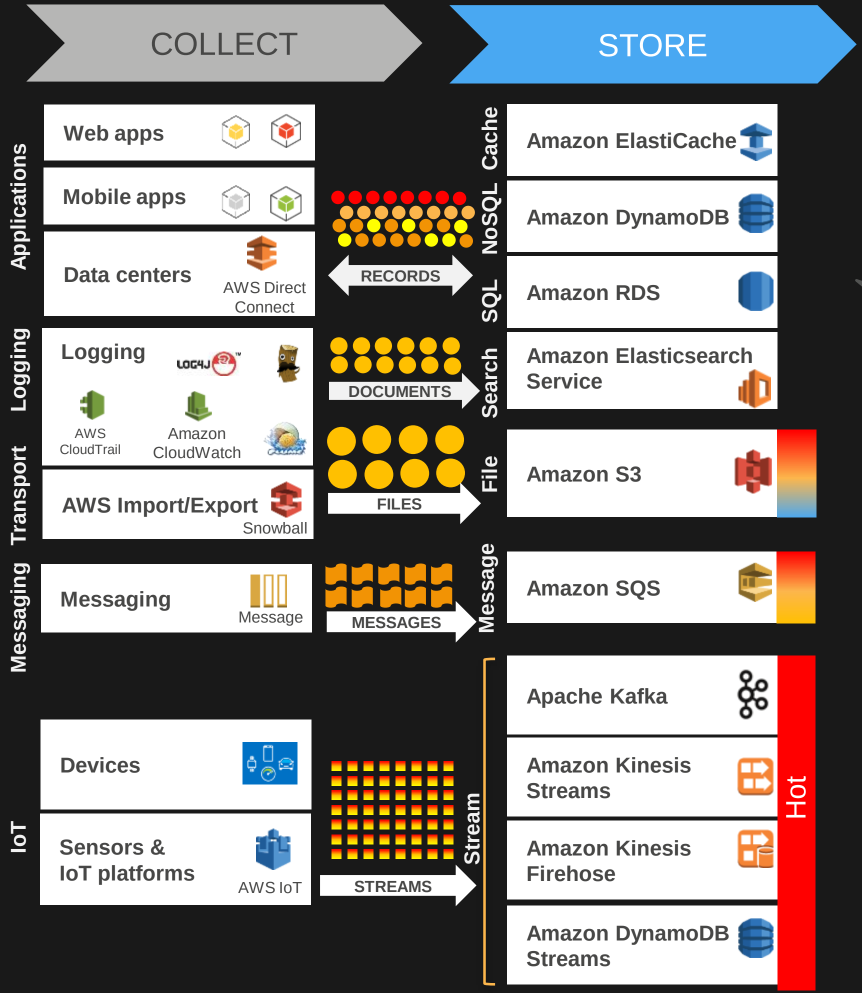
Why is Amazon S3 Good for Big Data?

- Natively supported by big data frameworks (Spark, Hive, Presto, etc.)
- No need to run compute clusters for storage (unlike HDFS)
- Can run transient Hadoop clusters & Amazon EC2 Spot Instances
- Multiple distinct (Spark, Hive, Presto) clusters can use the same data
- Unlimited number of objects
- Very high bandwidth – no aggregate throughput limit
- Highly available – can tolerate AZ failure
- Designed for 99.999999999% durability
- Tired-storage (Standard, IA, Amazon Glacier) via life-cycle policy
- Secure – SSL, client/server-side encryption at rest
- Low cost

What about HDFS & Amazon Glacier?

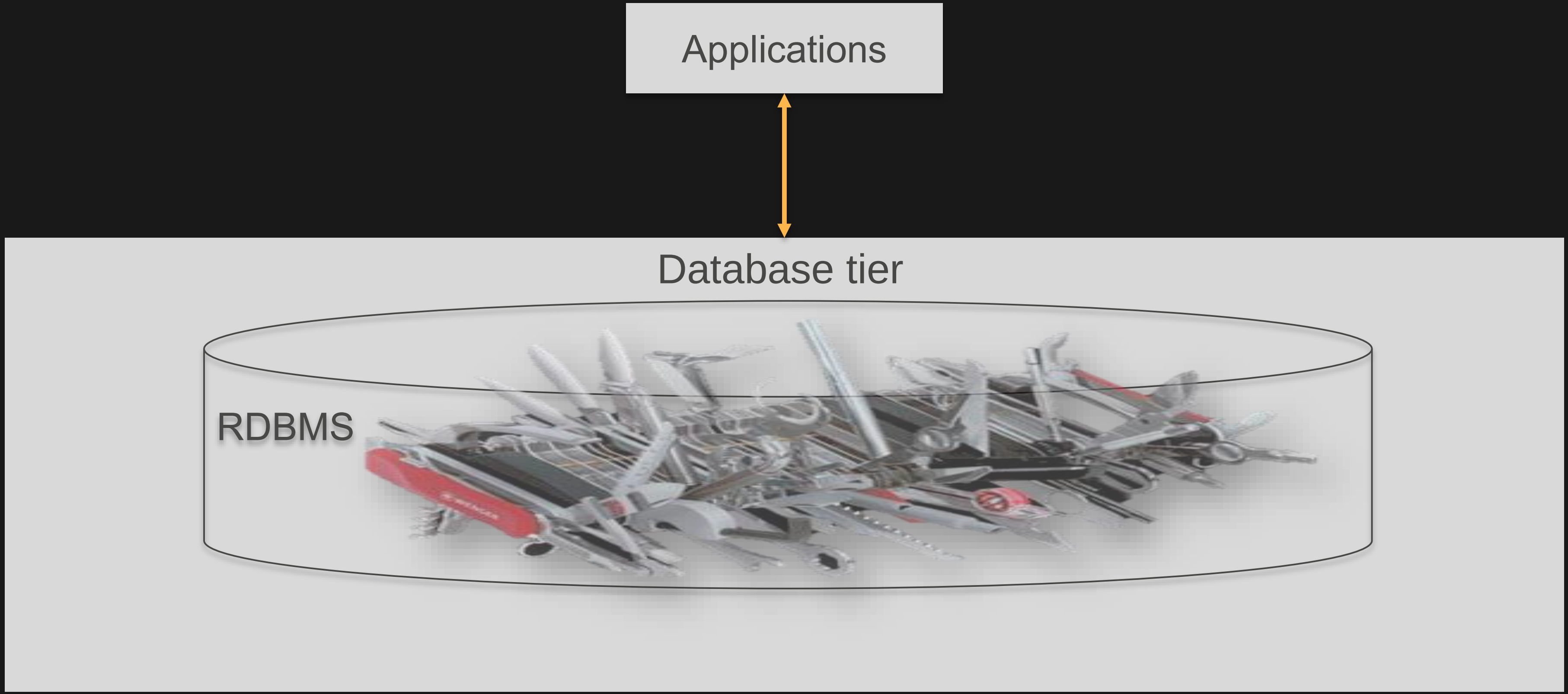
- Use HDFS for very frequently accessed (hot) data
- Use Amazon S3 Standard for frequently accessed data
- Use Amazon S3 Standard – IA for infrequently accessed data
- Use Amazon Glacier for archiving cold data





Cache, database, search

Database Anti-pattern

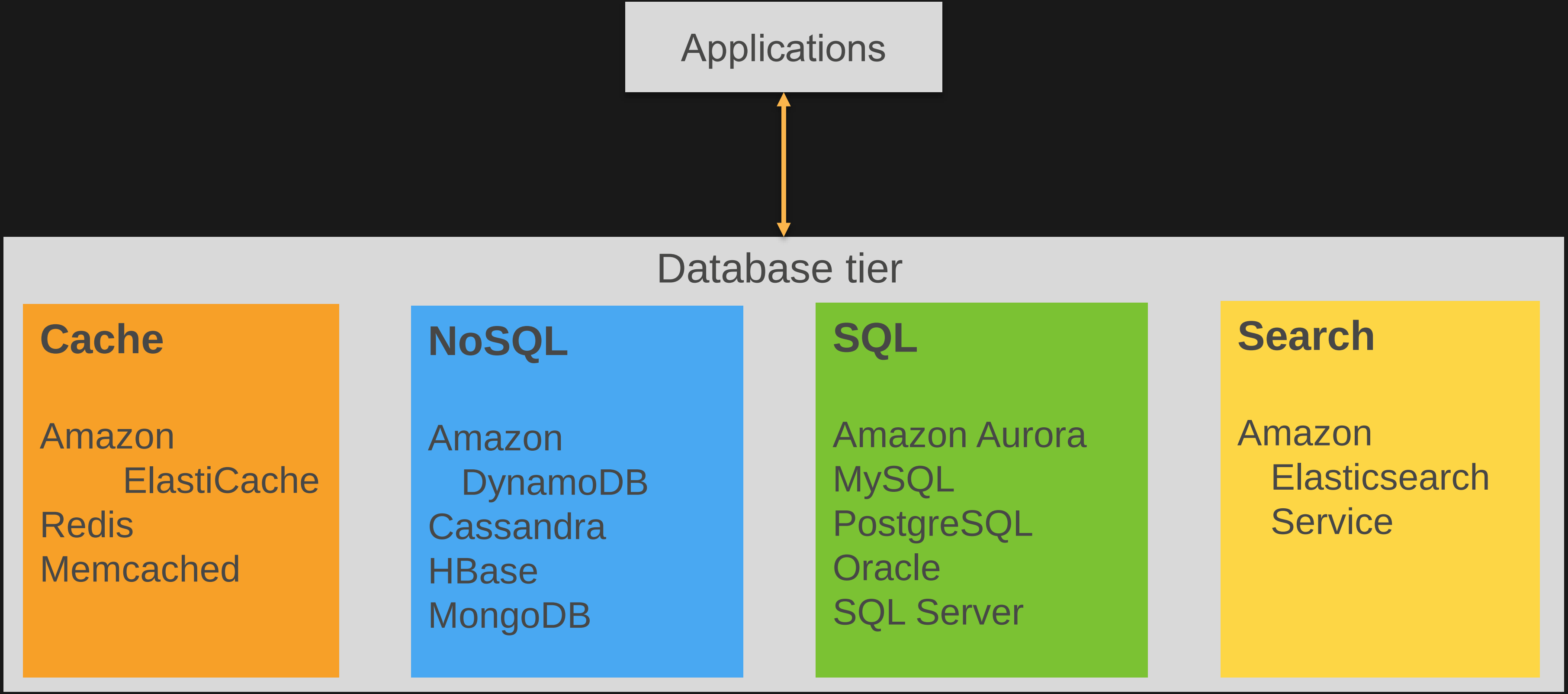


Applications

Database tier

RDBMS

Best Practice - Use the Right Tool for the Job



Applications

Database tier

Cache

Amazon ElastiCache
Redis
Memcached

NoSQL

Amazon DynamoDB
Cassandra
HBase
MongoDB

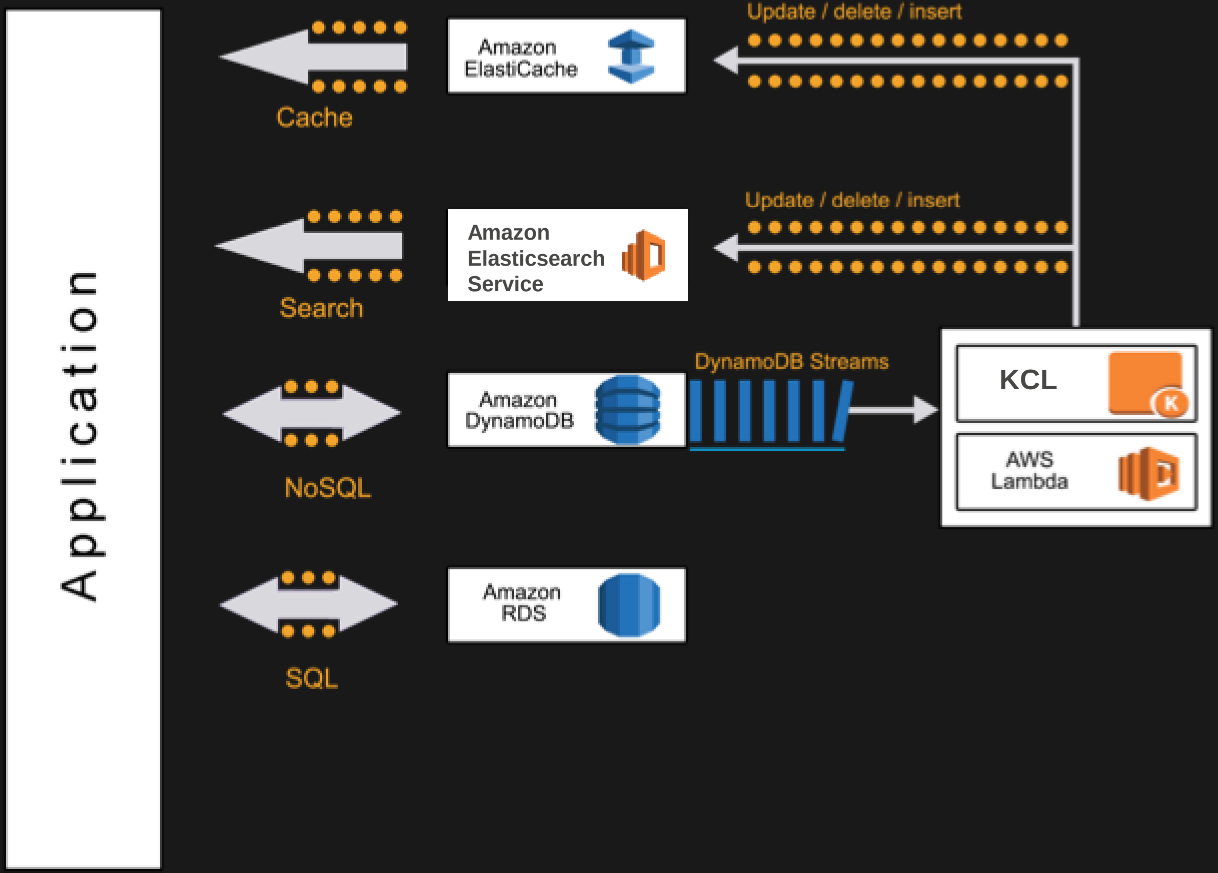
SQL

Amazon Aurora
MySQL
PostgreSQL
Oracle
SQL Server

Search

Amazon Elasticsearch Service

Materialized Views



What Data Store Should I Use?

Data structure → Fixed schema, JSON, key-value

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

Data / access characteristics → Hot, warm, cold

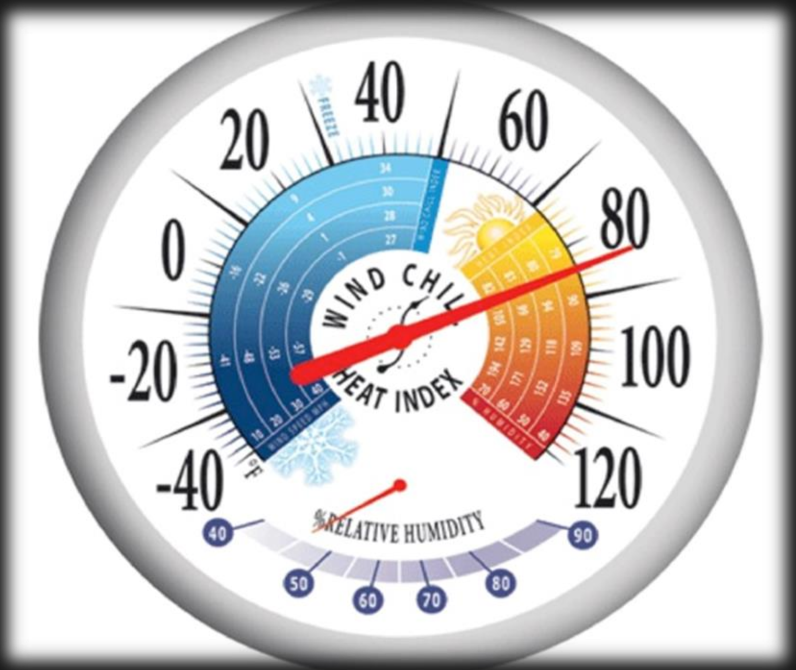
Cost → Right cost

Data Structure and Access Patterns

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

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

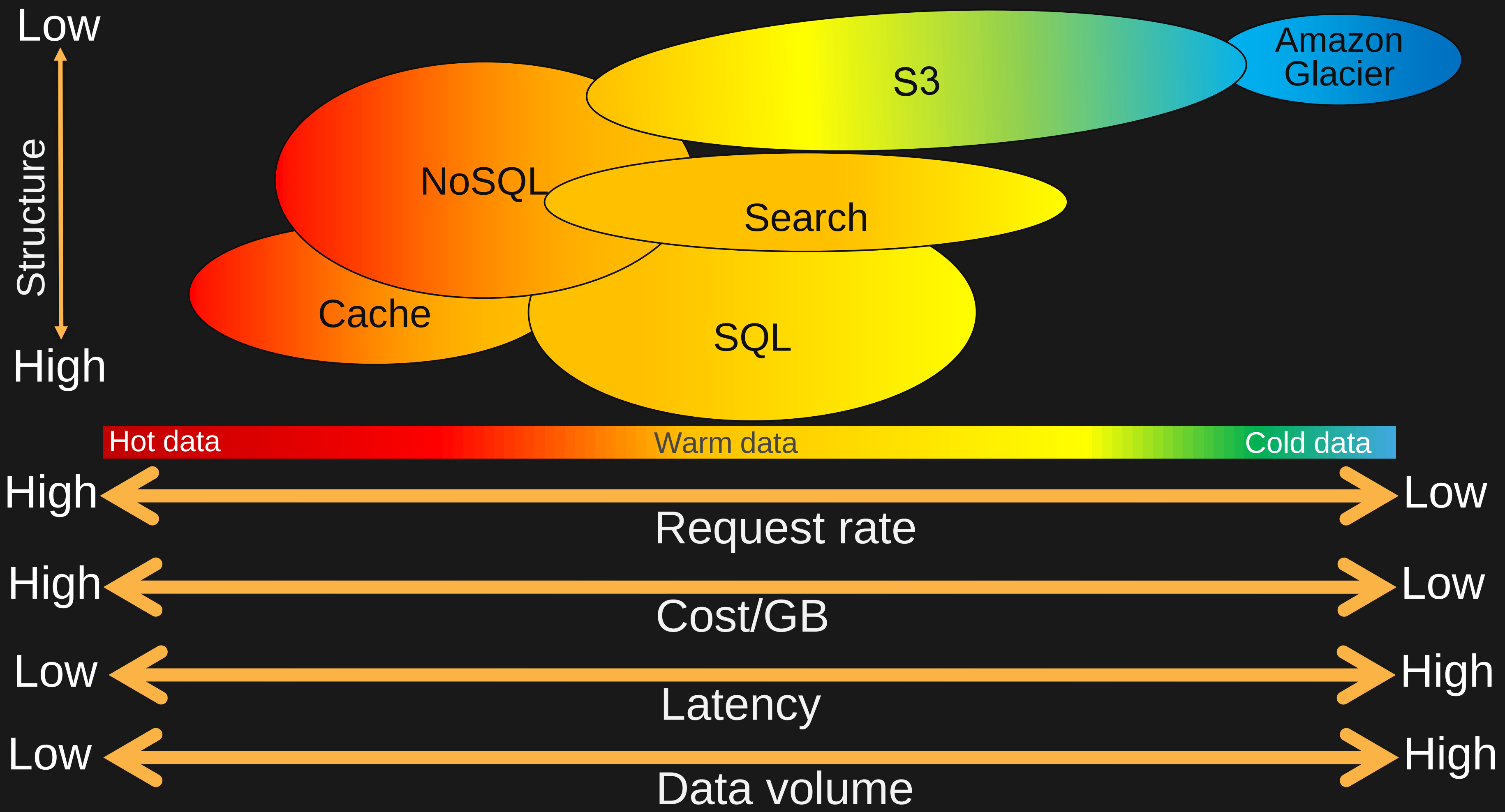
What Is the Temperature of Your Data / Access ?



Data / Access Characteristics: Hot, Warm, Cold

	Hot	Warm	Cold
Volume	MB–GB	GB–TB	PB
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	$$-$	$-¢¢	¢

Hot data Warm data Cold data



Low

Structure

High

NoSQL

S3

Amazon Glacier

Search

Cache

SQL

Hot data

Warm data

Cold data

High ↔ Low
Request rate

High ↔ Low
Cost/GB

Low ↔ High
Latency

Low ↔ High
Data volume

What Data Store Should I Use?

	Amazon ElastiCache	Amazon DynamoDB	Amazon RDS/Aurora	Amazon Elasticsearch	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)	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	$$	¢¢	¢¢	¢¢	¢	¢/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

Hot data — Warm data — Cold data

Cost Conscious Design

Example: Should I use Amazon S3 or Amazon DynamoDB?

“I’m currently scoping out a project that will greatly increase my team’s use of Amazon S3. Hoping you could answer some questions. The current iteration of 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?

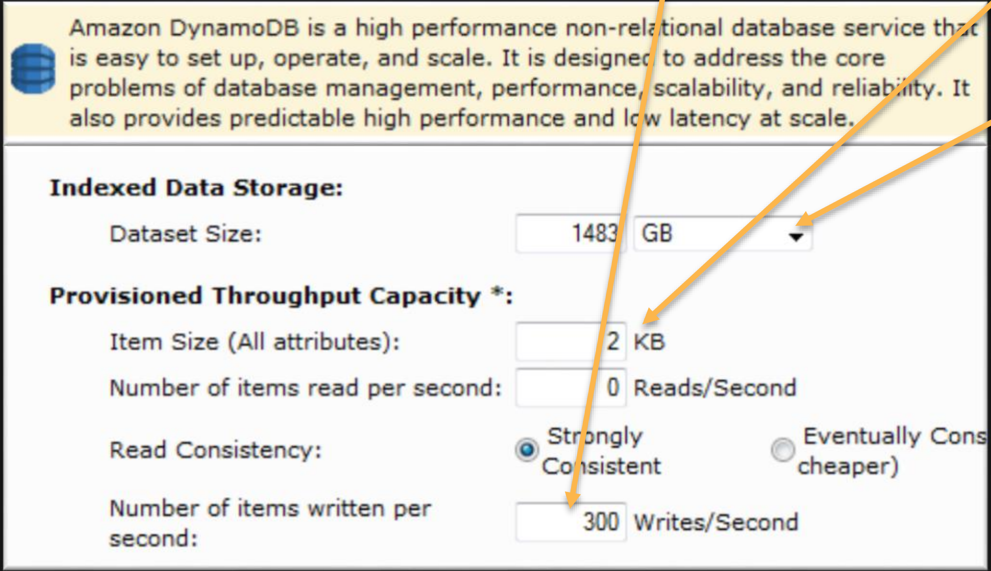
amazon web services

Simple Monthly Calculator

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

Amazon S3 or Amazon DynamoDB?

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



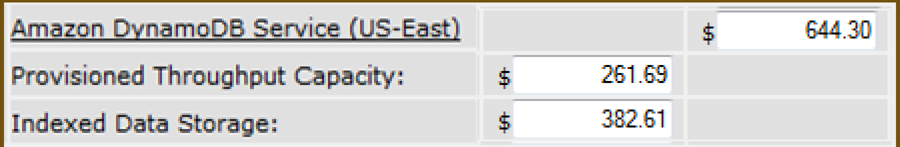
Amazon DynamoDB is a high performance non-relational database service that is easy to set up, operate, and scale. It is designed to address the core problems of database management, performance, scalability, and reliability. It also provides predictable high performance and low latency at scale.

Indexed Data Storage:

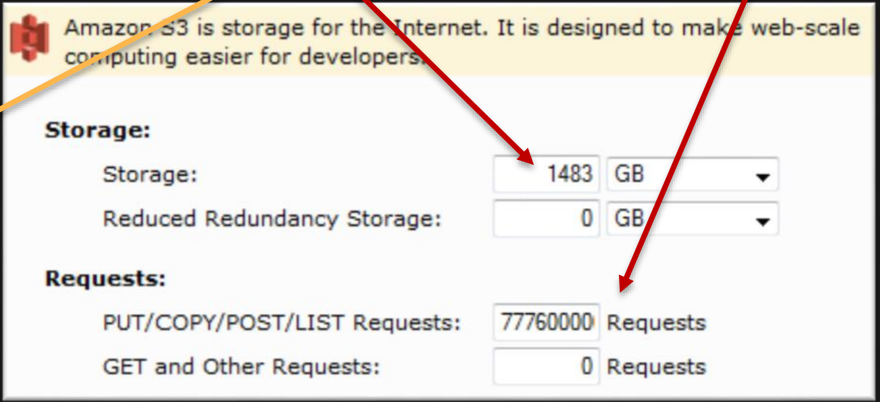
Dataset Size:	1483	GB

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	



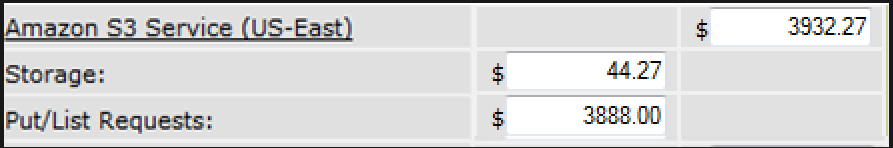
Amazon S3 is storage for the Internet. It is designed to make web-scale computing easier for developers.

Storage:

Storage:	1483	GB
Reduced Redundancy Storage:	0	GB

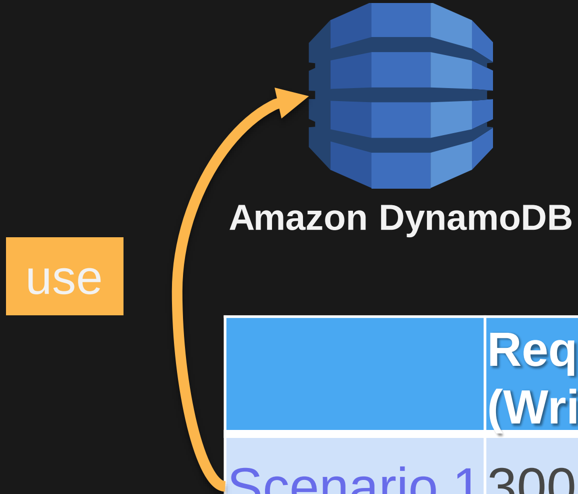
Requests:

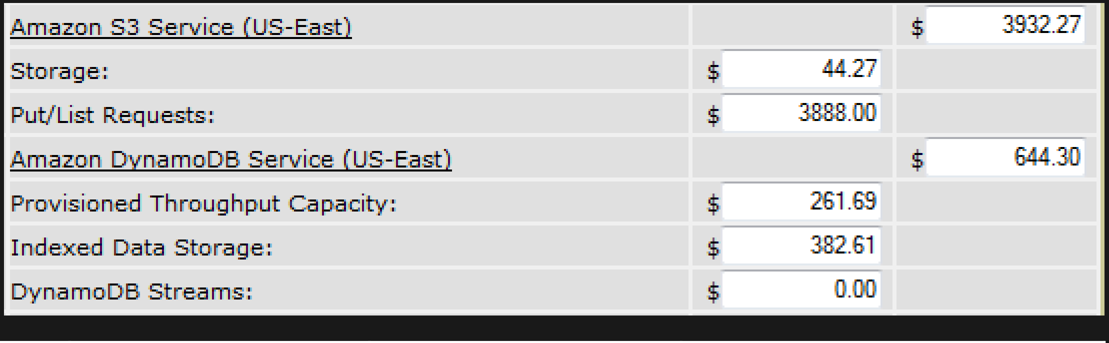
PUT/COPY/POST/LIST Requests:	77760000	Requests
GET and Other Requests:	0	Requests



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

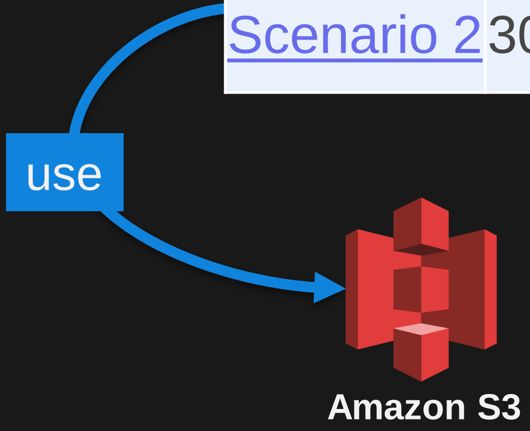
amazon web services SIMPLE MONTHLY CALCULATOR

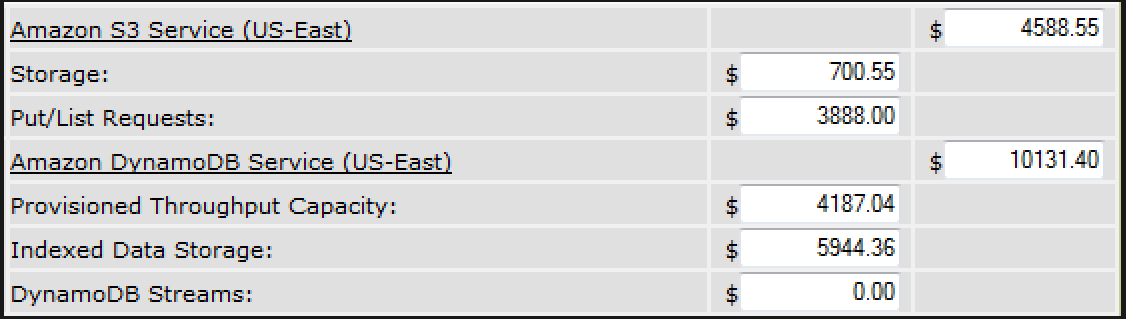




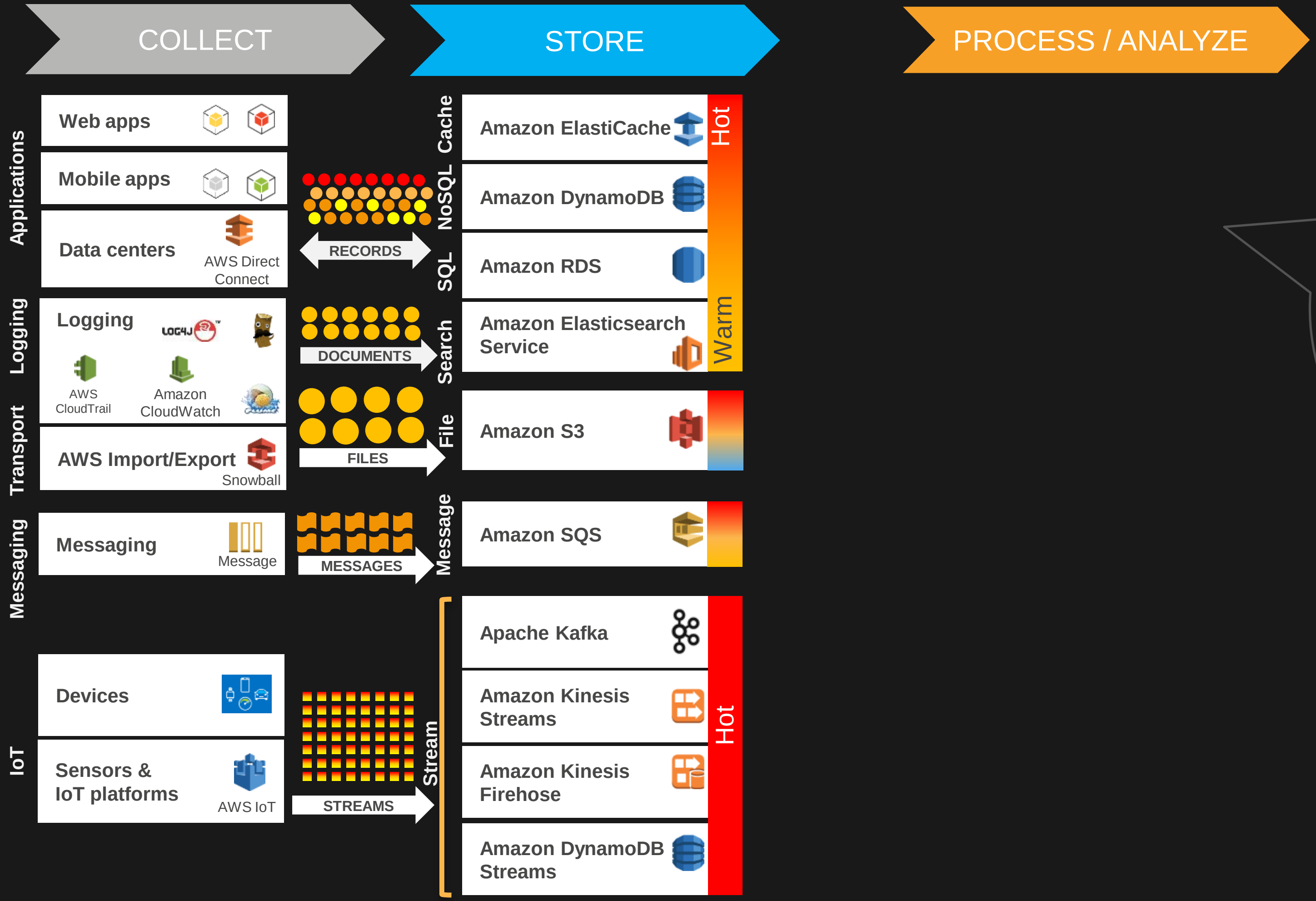
Amazon S3 Service (US-East)		$ 3932.27
Storage:	$ 44.27	
Put/List Requests:	$ 3888.00	
Amazon DynamoDB Service (US-East)		$ 644.30
Provisioned Throughput Capacity:	$ 261.69	
Indexed Data Storage:	$ 382.61	
DynamoDB Streams:	$ 0.00	

	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





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	



COLLECT

STORE

PROCESS / ANALYZE

Applications
- Web apps
- Mobile apps
- Data centers — AWS Direct Connect

Logging
- Logging — LOG4J — 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 — Amazon ElastiCache

NoSQL — Amazon DynamoDB

SQL — Amazon RDS

Search — Amazon Elasticsearch Service

Hot

Warm

File — Amazon S3

Message — Amazon SQS

Stream
- Apache Kafka
- Amazon Kinesis Streams
- Amazon Kinesis Firehose
- Amazon DynamoDB Streams

Hot

Process / analyze

PROCESS / ANALYZE

Process / Analyze

- Batch - Minutes or hours on cold data
  - Daily/weekly/monthly reports
- Interactive – Seconds on warm/cold data
  - Self-service dashboards
- Messaging – Milliseconds or seconds on hot data
  - Message/event buffering
- Streaming - Milliseconds or seconds on hot data
  - Billing/fraud alerts, 1 minute metrics

Predictions via Machine Learning

ML gives computers the ability to learn without being explicitly programmed

Machine learning algorithms:

Supervised learning ← “teach” program

- Classification ← Is this transaction fraud? (yes/no)
- Regression ← Customer life-time value?

Unsupervised learning ← Let it learn by itself

- Clustering ← Market segmentation

Tools and Frameworks

Machine Learning

- Amazon ML, Amazon EMR (Spark ML)

Interactive

- Amazon Redshift, Amazon EMR (Presto, Spark)

Batch

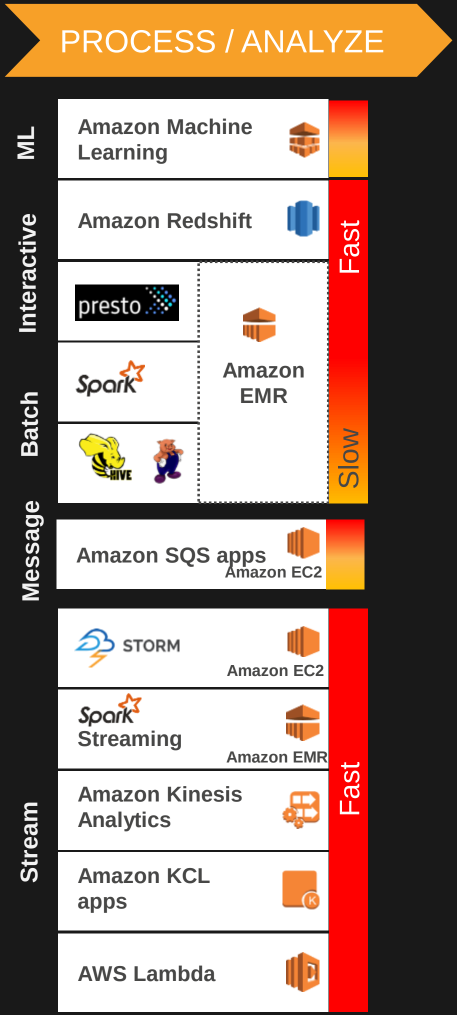
- Amazon EMR (MapReduce, Hive, Pig, Spark)

Messaging

- Amazon SQS application on Amazon EC2

Streaming

- Micro-batch: Spark Streaming, KCL
- Real-time: Amazon Kinesis Analytics, Storm, AWS Lambda, KCL



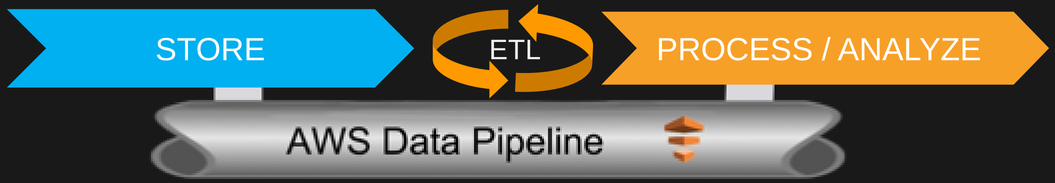
What Streaming / Messaging Technology Should I Use?

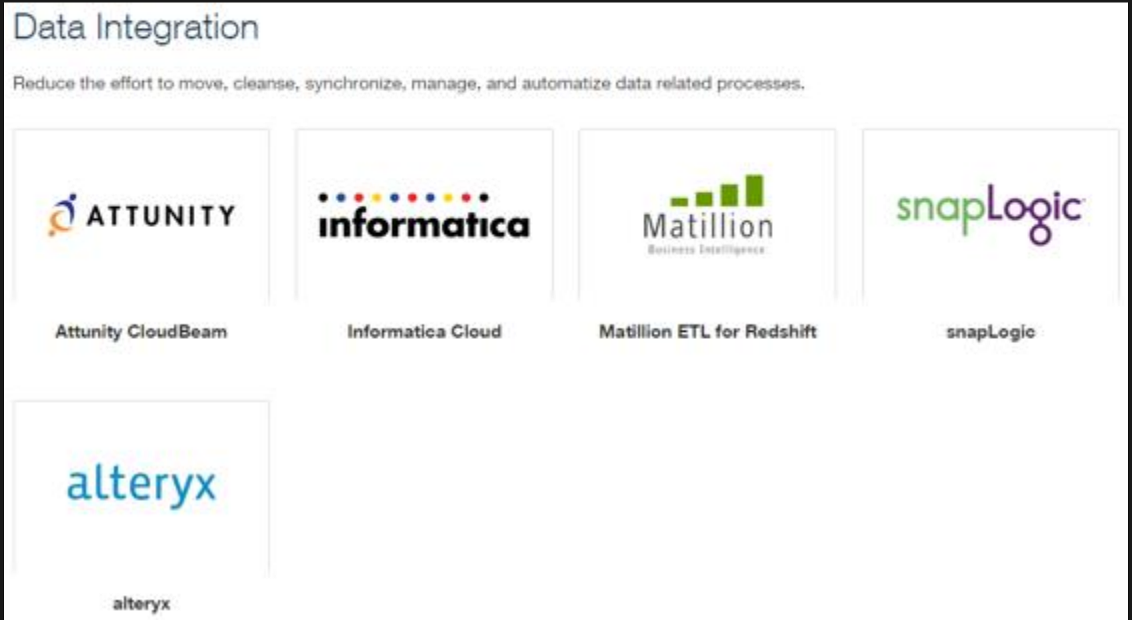
	Spark Streaming	Apache Storm	Kinesis KCL Application	AWS Lambda	Amazon SQS Apps
Scale	~ Nodes	~ Nodes	~ Nodes	Automatic	~ Nodes
Micro-batch or Real-time	Micro-batch	Real-time	Near-real-time	Near-real-time	Near-real-time
AWS managed service	Yes (EMR)	No (EC2)	No (KCL + EC2 + Auto Scaling)	Yes	No (EC2 + Auto Scaling)
Scalability	No limits ~ nodes	No limits ~ nodes	No limits ~ nodes	No limits	No limits
Availability	Single AZ	Configurable	Multi-AZ	Multi-AZ	Multi-AZ
Programming languages	Java, Python, Scala	Any language via Thrift	Java, via MultiLang Daemon (.NET, Python, Ruby, Node.js)	Node.js, Java, Python	AWS SDK languages (Java, .NET, Python, …)
	Fast		Fast		Fast

What Analytics Technology Should I Use?

	Amazon Redshift	Amazon EMR		
		Presto	Spark	Hive
Query latency	Low	Low	Low	High
Durability	High	High	High	High
Data volume	1.6 PB max	~Nodes	~Nodes	~Nodes
AWS managed	Yes	Yes	Yes	Yes
Storage	Native	HDFS / S3	HDFS / S3	HDFS / S3
SQL compatibility	High	High	Low (SparkSQL)	Medium (HQL)
	Fast	Fast	Fast	Slow

What About ETL?





Data Integration

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

Attunity CloudBeam

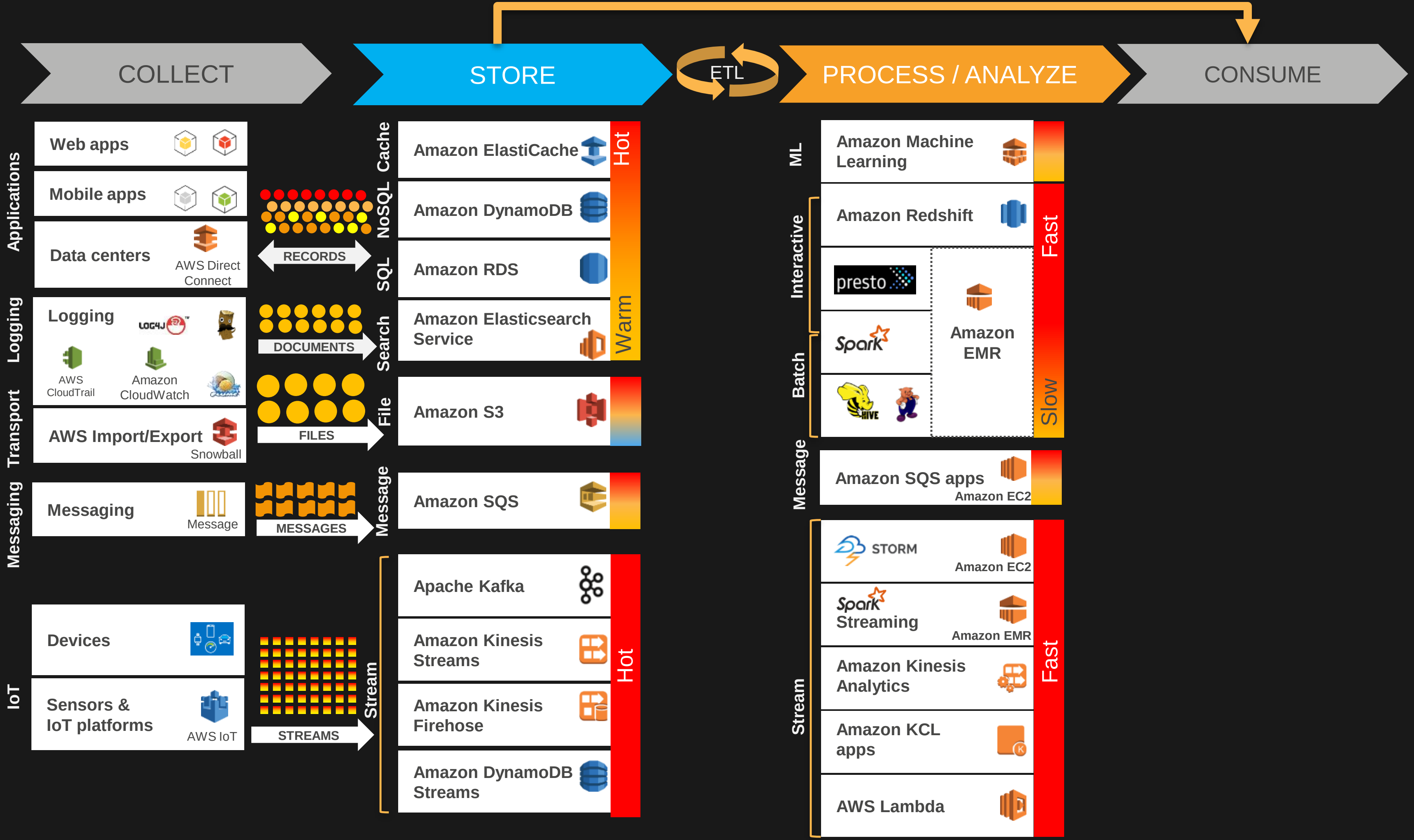
Informatica Cloud

Matillion ETL for Redshift

snapLogic

alteryx

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



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

Amazon ElastiCache

NoSQL

Amazon DynamoDB

SQL

Amazon RDS

Search

Amazon Elasticsearch Service

Hot

Warm

File

Amazon S3

Message

Amazon SQS

Stream

Apache Kafka

Amazon Kinesis Streams

Amazon Kinesis Firehose

Amazon DynamoDB Streams

Hot

ML

Amazon Machine Learning

Interactive

Amazon Redshift

presto

Batch

Spark

HIVE

Amazon EMR

Fast

Slow

Message

Amazon SQS apps

Amazon EC2

Stream

STORM

Amazon EC2

Spark Streaming

Amazon EMR

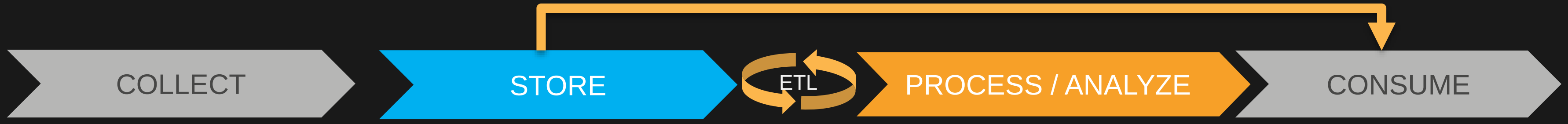
Amazon Kinesis Analytics

Amazon KCL apps

AWS Lambda

Fast

CONSUME

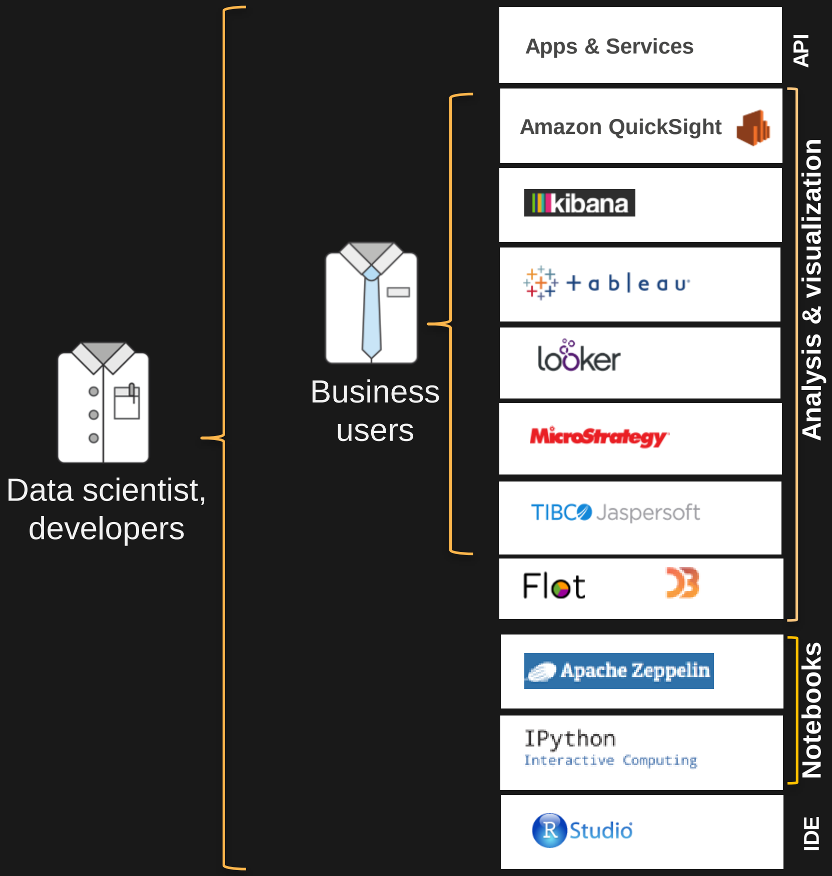


Applications & API

Analysis and visualization

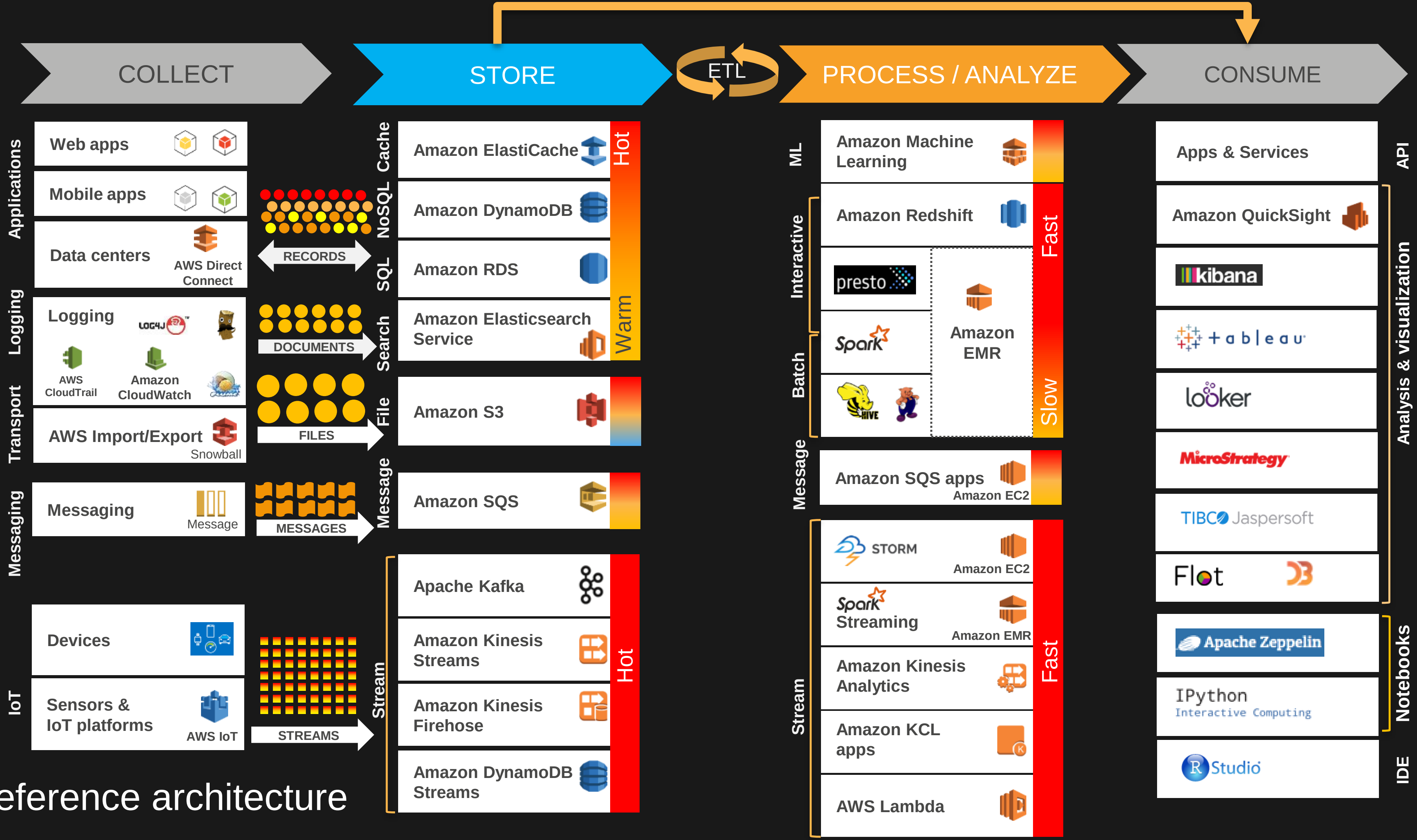
Notebooks

IDE



Putting It All Together

Reference architecture



COLLECT → STORE → ETL → PROCESS / ANALYZE → CONSUME

COLLECT

Applications
- Web apps
- Mobile apps
- Data centers — AWS Direct Connect

Logging
- Logging — LOG4J — AWS CloudTrail — Amazon CloudWatch

Transport
- AWS Import/Export Snowball

Messaging
- Messaging — Message

IoT
- Devices
- Sensors & IoT platforms — AWS IoT

RECORDS

DOCUMENTS

FILES

MESSAGES

STREAMS

STORE

Cache
- Amazon ElastiCache

NoSQL
- Amazon DynamoDB

SQL
- Amazon RDS

Search
- Amazon Elasticsearch Service

Hot

Warm

File
- Amazon S3

Message
- Amazon SQS

Stream
- Apache Kafka
- Amazon Kinesis Streams
- Amazon Kinesis Firehose
- Amazon DynamoDB Streams

Hot

PROCESS / ANALYZE

ML
- Amazon Machine Learning

Interactive
- Amazon Redshift
- presto

Batch
- Spark
- HIVE

Amazon EMR

Fast

Slow

Message
- Amazon SQS apps — Amazon EC2

Stream
- STORM — Amazon EC2
- Spark Streaming — Amazon EMR
- Amazon Kinesis Analytics
- Amazon KCL apps
- AWS Lambda

Fast

CONSUME

API
- Apps & Services

Analysis & visualization
- Amazon QuickSight
- kibana
- tableau
- looker
- MicroStrategy
- TIBCO Jaspersoft
- Flot — D3

Notebooks
- Apache Zeppelin
- IPython Interactive Computing

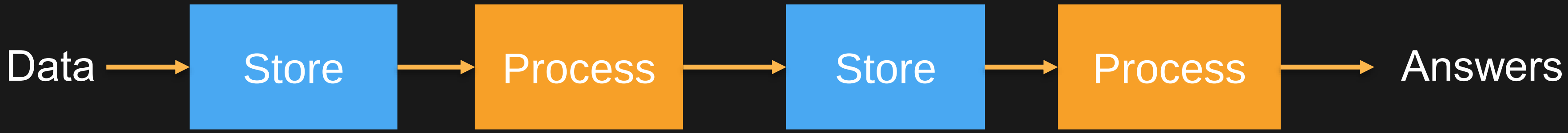
IDE
- RStudio

Design Patterns

Primitive: Multi-Stage Decoupled “Data Bus”

Multiple stages

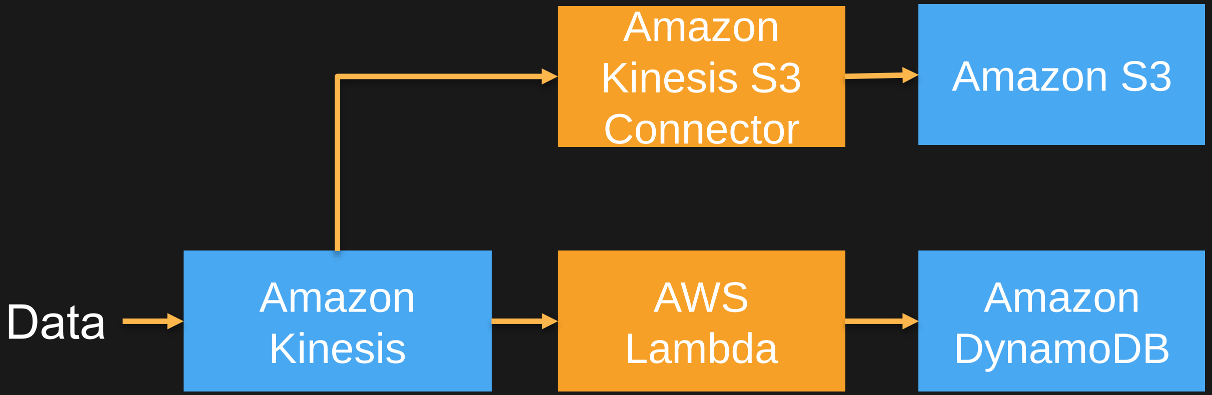
Storage decouples multiple processing stages



🟧 process

🟦 store

Primitive: Multiple Stream Processing Applications Can Read from Amazon Kinesis



Data

Amazon Kinesis

Amazon Kinesis S3 Connector

Amazon S3

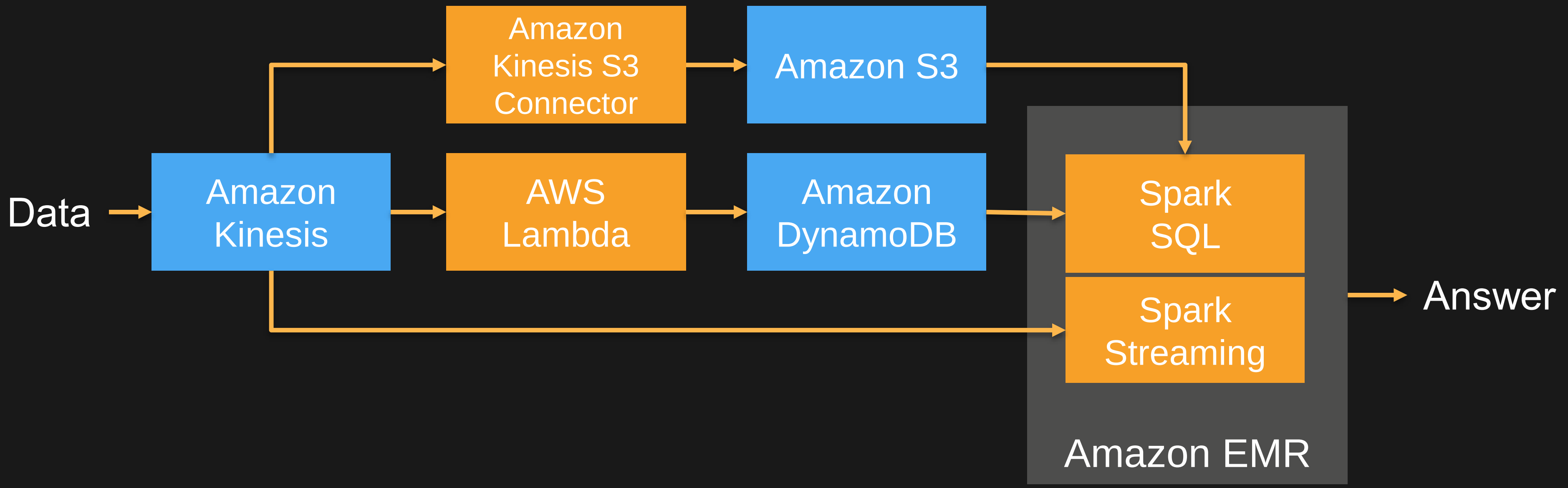
AWS Lambda

Amazon DynamoDB

🟧 process

🟦 store

Primitive: Analysis Frameworks Could Read from Multiple Data Stores



Data

Amazon Kinesis

Amazon Kinesis S3 Connector

Amazon S3

AWS Lambda

Amazon DynamoDB

Spark SQL

Spark Streaming

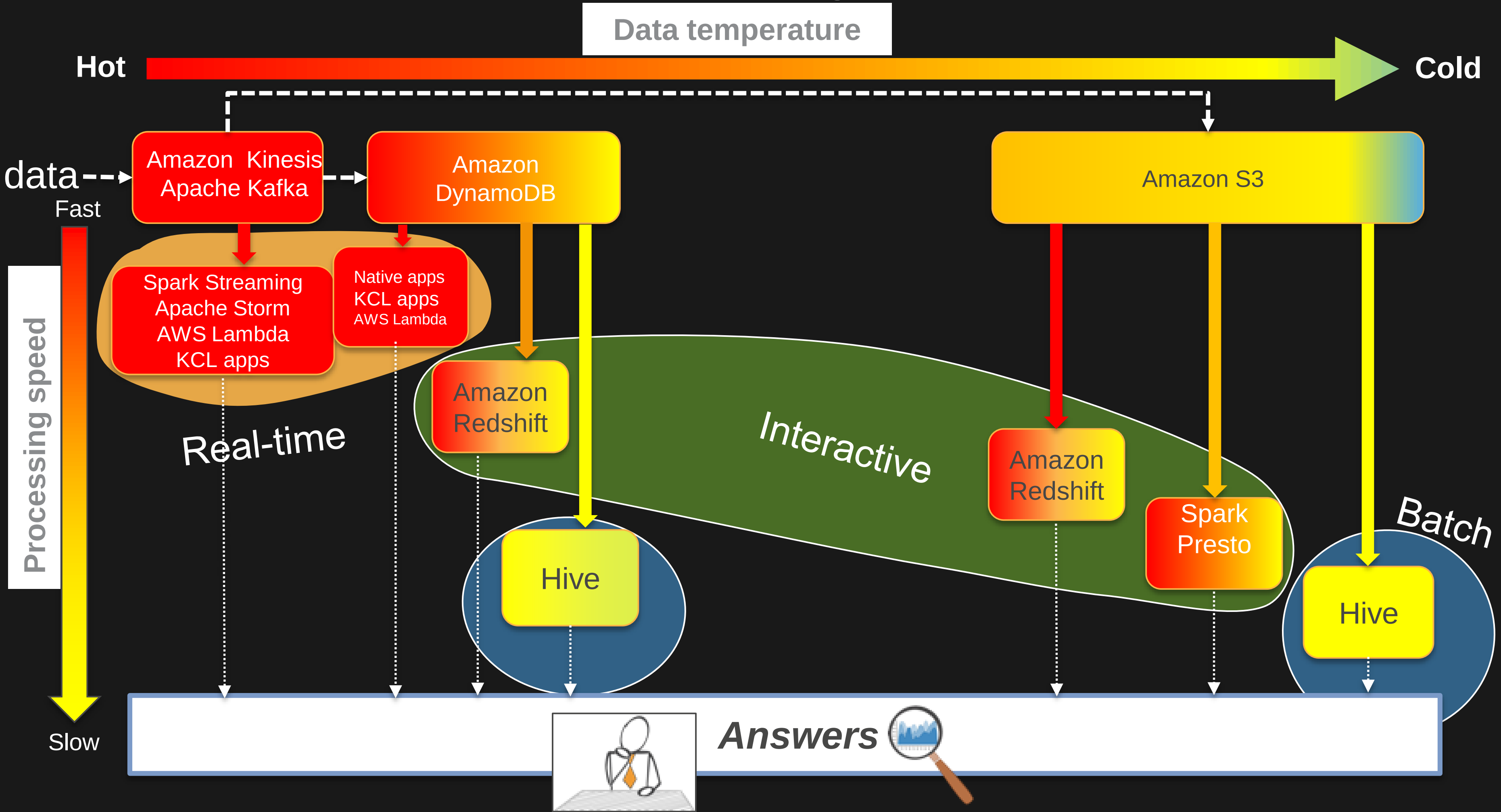
Amazon EMR

Answer

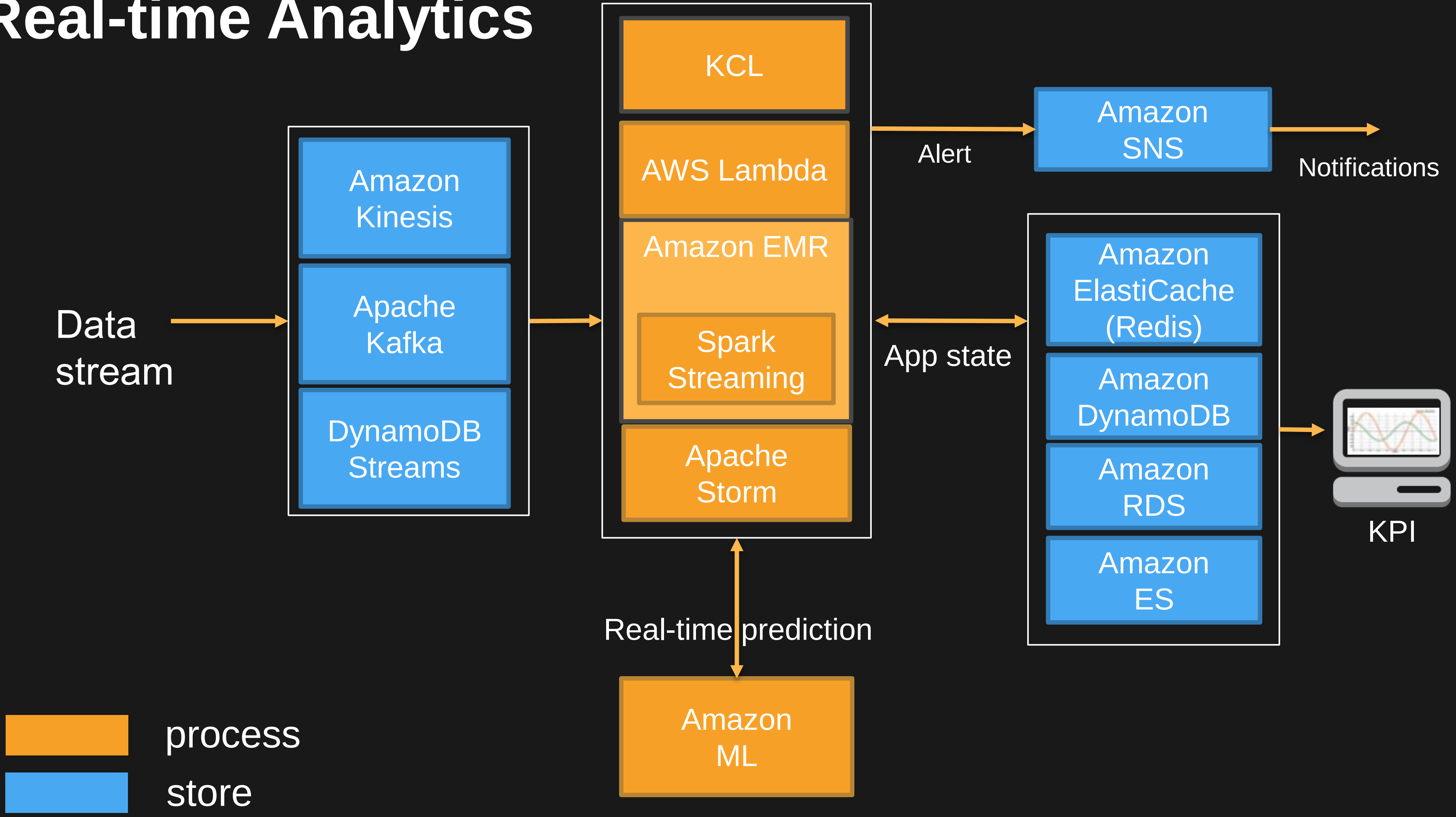
🟧 process

🟦 store

Data Temperature vs. Processing Speed



Real-time Analytics



Data stream

Amazon Kinesis

Apache Kafka

DynamoDB Streams

KCL

AWS Lambda

Amazon EMR

Spark Streaming

Apache Storm

Alert

Amazon SNS

Notifications

App state

Amazon ElastiCache (Redis)

Amazon DynamoDB

Amazon RDS

Amazon ES

KPI

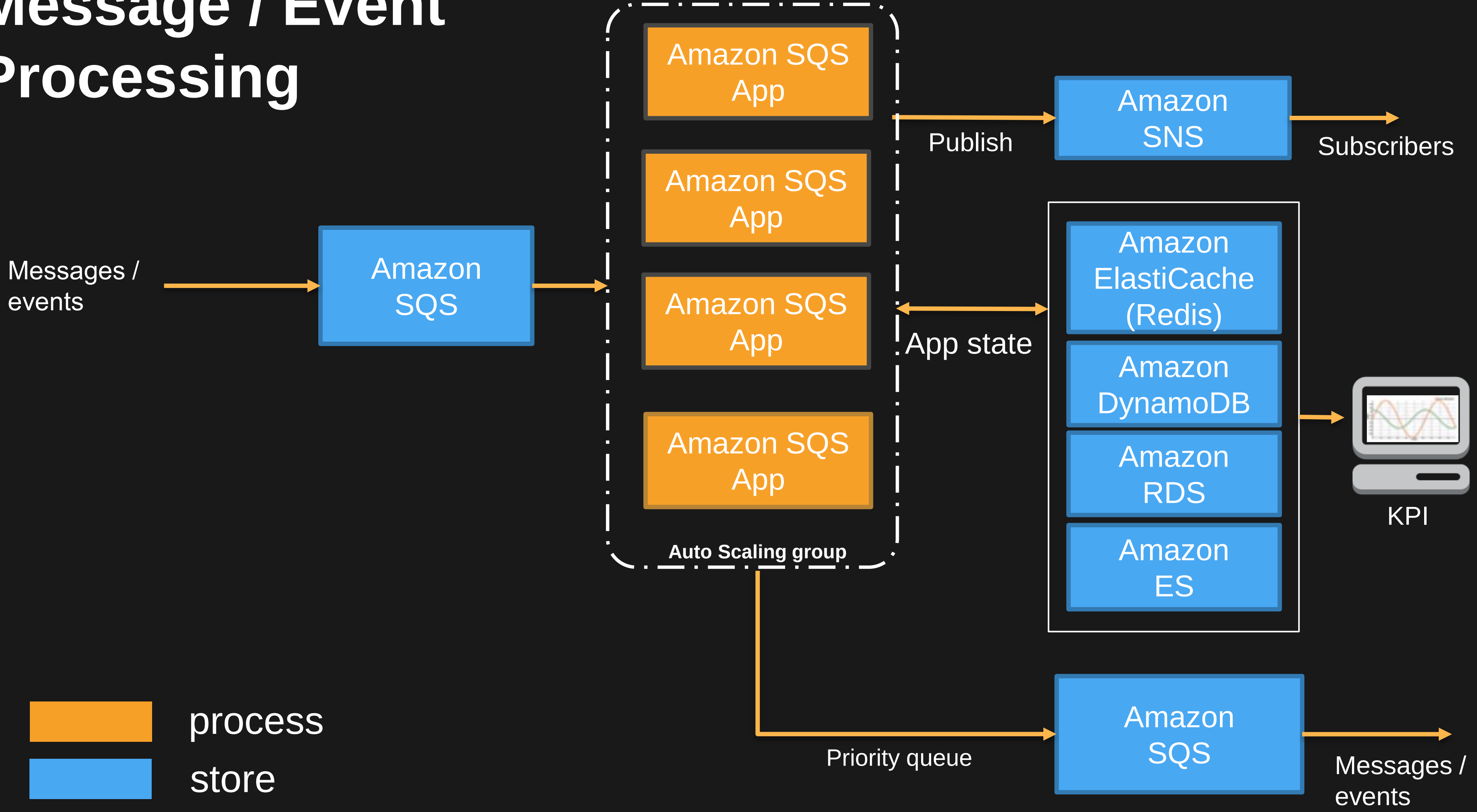
Real-time prediction

Amazon ML

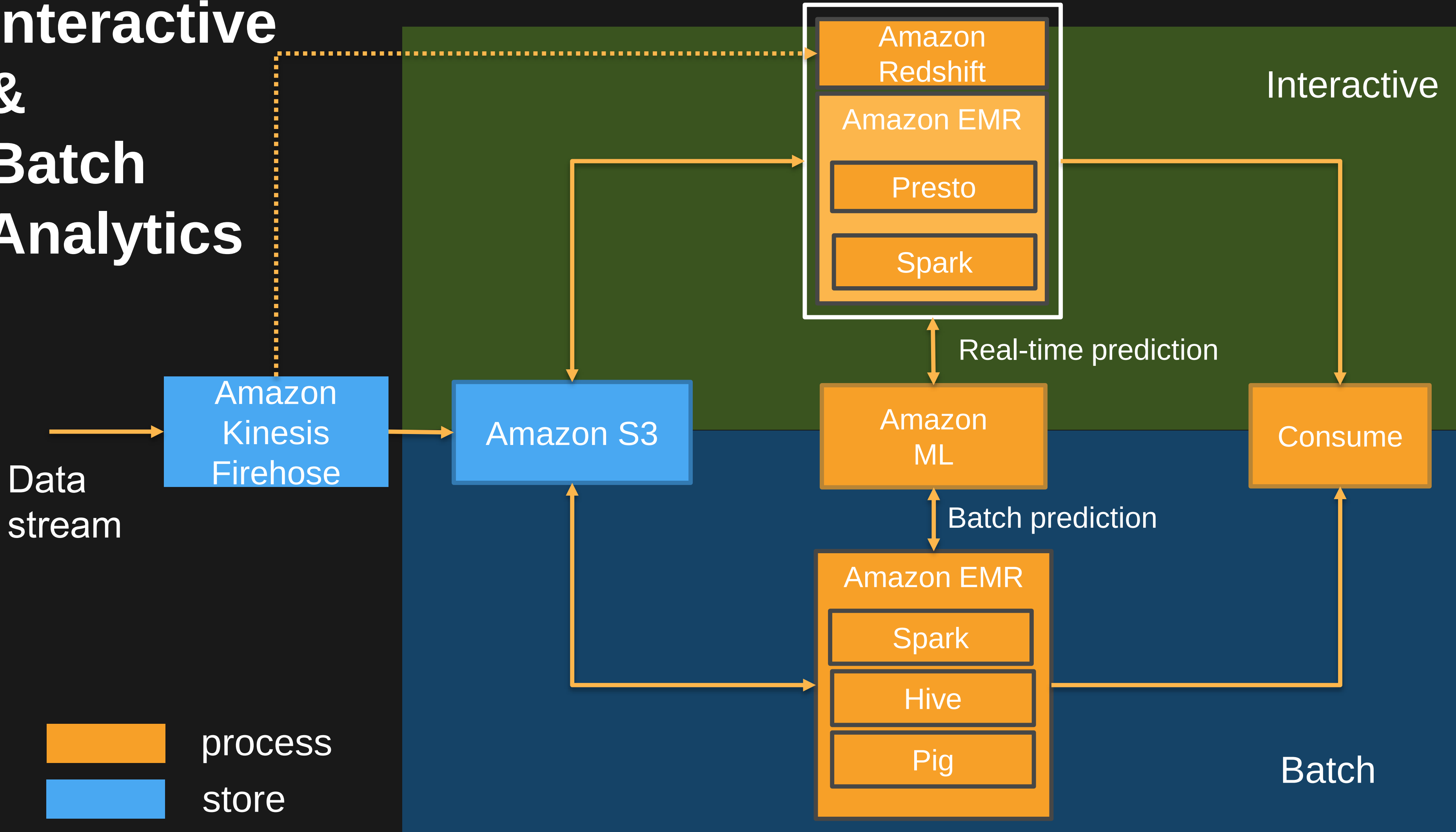
🟧 process

🟦 store

Message / Event Processing



Interactive & Batch Analytics



Data stream

Amazon Kinesis Firehose

Amazon S3

Amazon Redshift

Amazon EMR

Presto

Spark

Interactive

Real-time prediction

Amazon ML

Batch prediction

Consume

Amazon EMR

Spark

Hive

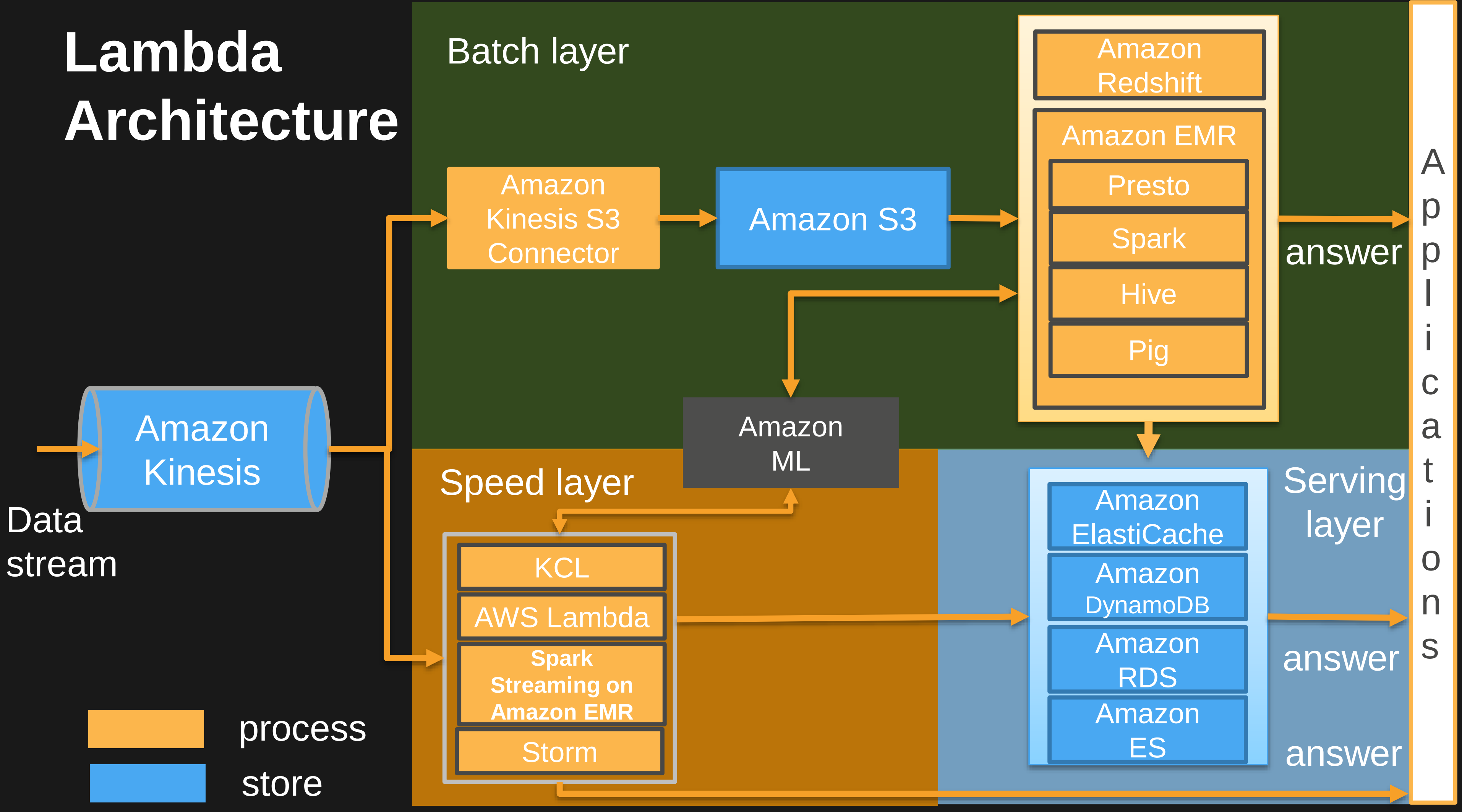
Pig

Batch

🟧 process

🟦 store

Lambda Architecture



Data stream

Amazon Kinesis

Batch layer

Amazon Kinesis S3 Connector

Amazon S3

Amazon Redshift

Amazon EMR

Presto

Spark

Hive

Pig

answer

Amazon ML

Speed layer

KCL

AWS Lambda

Spark Streaming on Amazon EMR

Storm

Serving layer

Amazon ElastiCache

Amazon DynamoDB

Amazon RDS

Amazon ES

answer

answer

Applications

🟧 process

🟦 store

Summary

Decoupled “data bus”

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

Use the right tool for the job

- Data structure, latency, throughput, access patterns

Use Lambda architecture ideas

- Immutable (append-only) log, batch/speed/serving layer

Leverage AWS managed services

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

Big data ≠ big cost

AWS

SUMMIT

Thank you!

aws.amazon.com/big-data

amazon
web services