



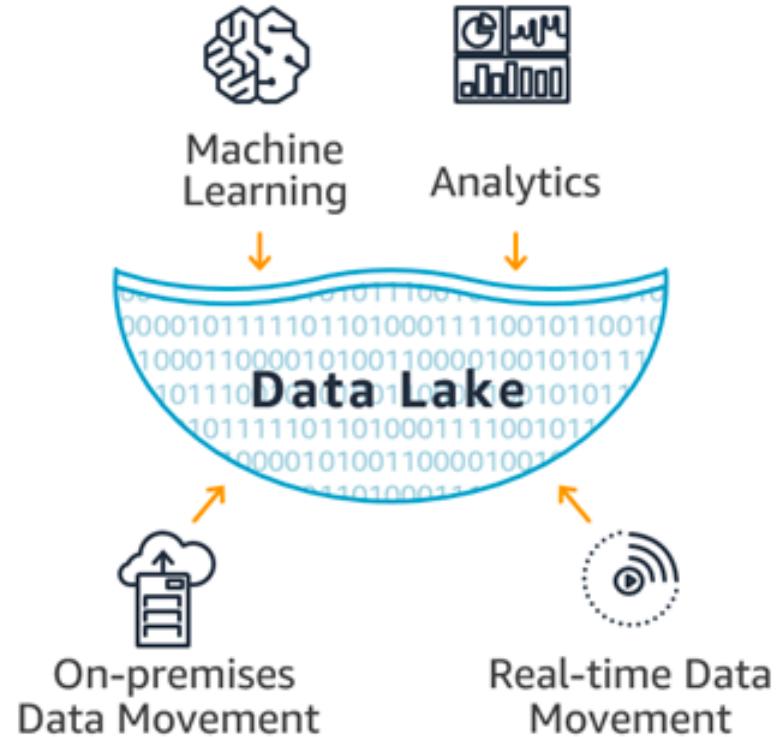
# Immersion Day – Building a Data Lake on AWS

# Workshop Agenda

- Overview of the Data Lake
- Hydrating the Data Lake
- Lab - Hydrating the Data Lake
- Working Within the Data Lake
- Lab - Transforming and Query Data in the Data
- Consuming the Data Lake – Reporting, Analytics, Machine Learning
- Lab – Extending Consuming Data Lake for Machine Learning
- Automating Data Lake with AWS Lake Formation
- Lab- Data Lake Automation

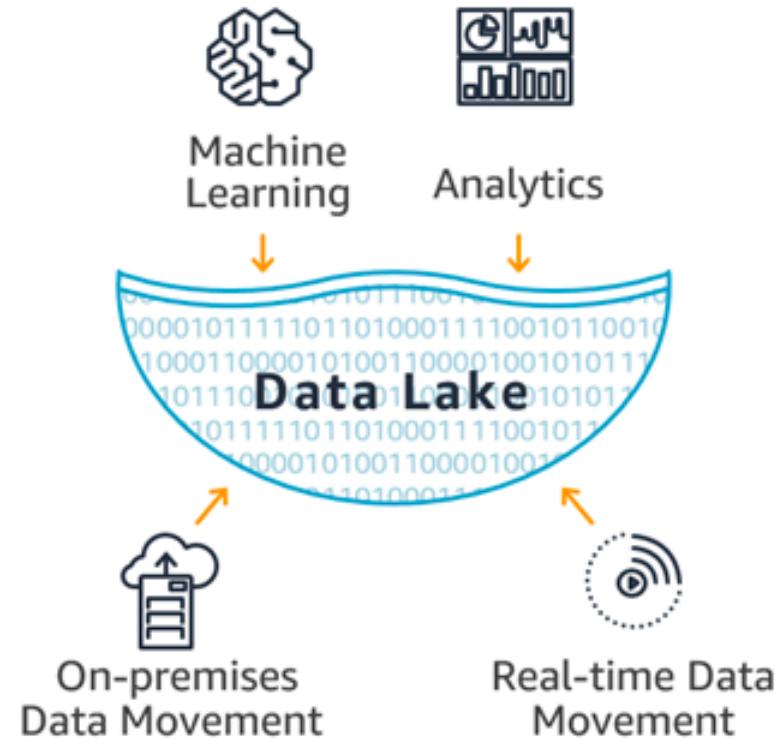
# What is a Data Lake?

- A **centralized repository** for both structured and unstructured data
- Store data **as-is** in open-source file formats to enable direct analytics

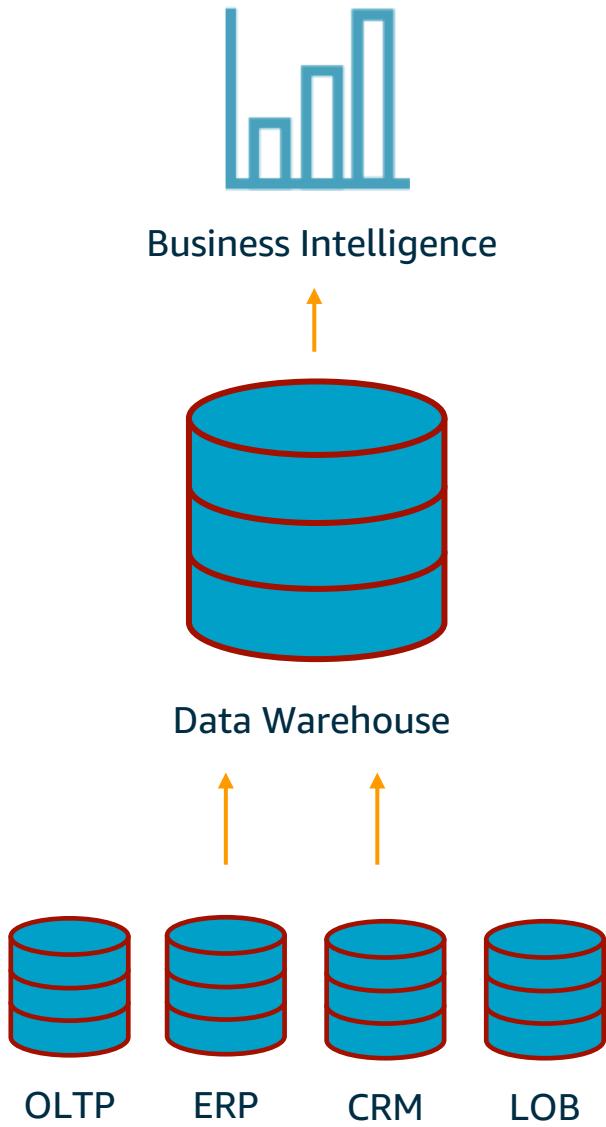


# Why a Data Lake?

- Decouple storage from compute, allowing you to scale
- Enable advanced analytics across all of your data sources
- Reduce complexity in ETL and operational overhead
- Future extensibility as new database and analytics technologies are invented



# Traditionally, Analytics Looked Like This



Relational Data

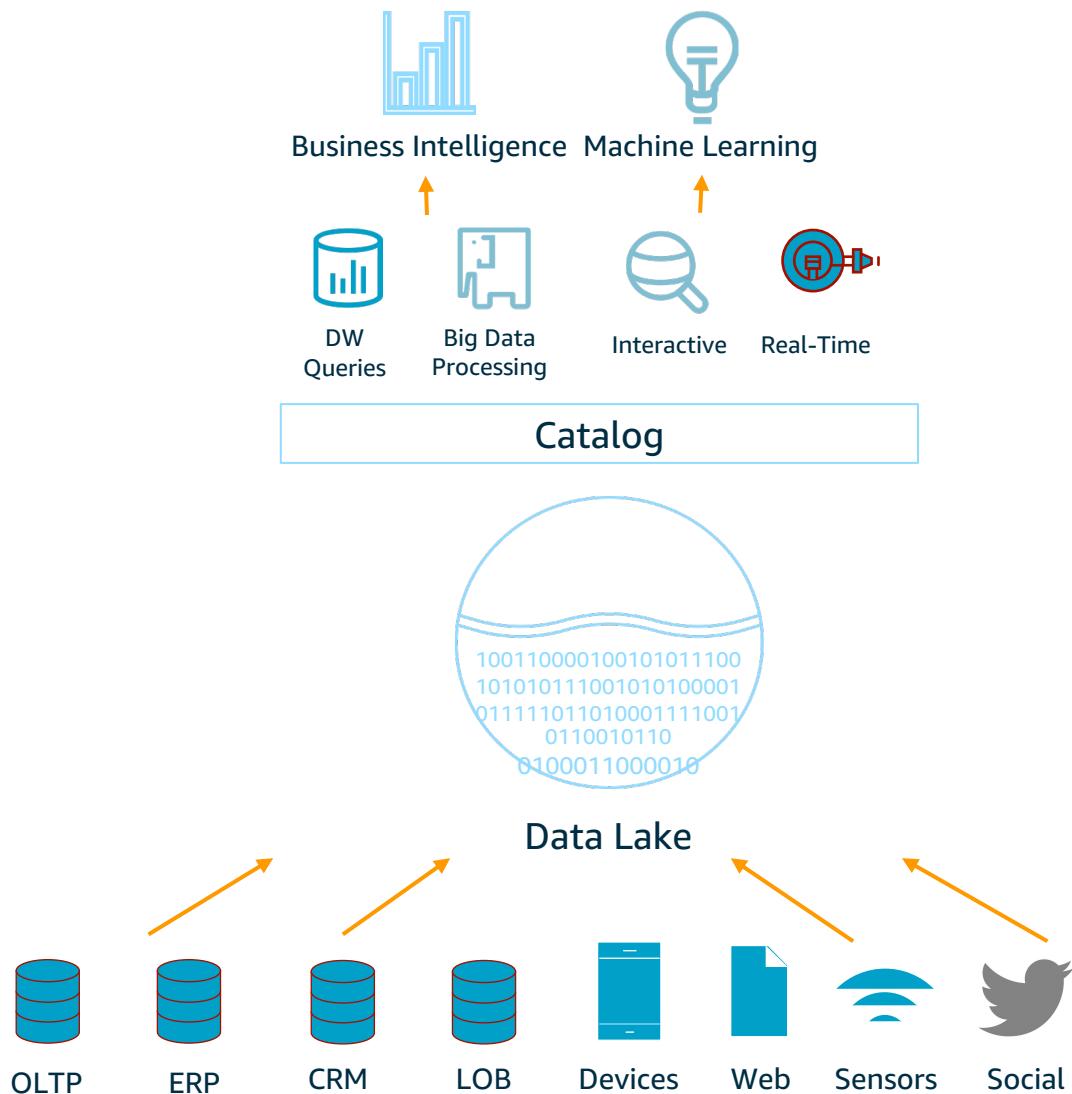
TBs-PBs Scale

Schema Defined Prior to Data Load

Operational and Ad Hoc Reporting

Large Initial Capex + \$\$K / TB / Year

# Data Lakes Extend the Traditional Approach



## TB-EBs Scale

All Data in one place, a Single Source of Truth

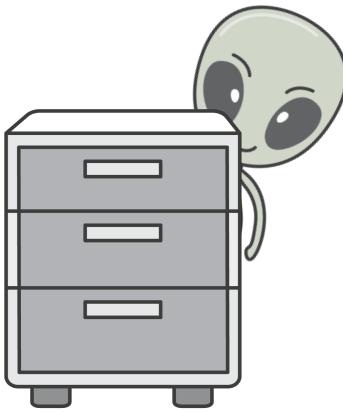
Relational and Non-Relational Data

Decouples (low cost) Storage and Compute

Schema on Read

Diverse Analytical Engines

# Benefits of a Data Lake – All Data in One Place

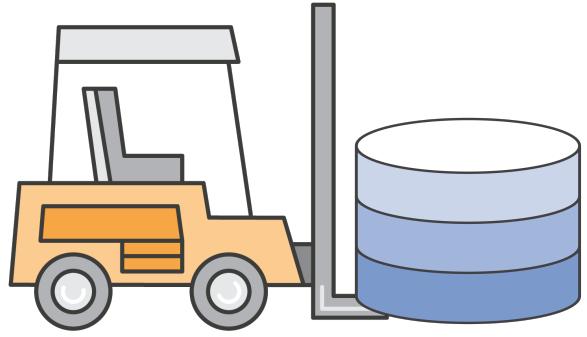


“Why is the data distributed in many locations? Where is the single source of truth ?”

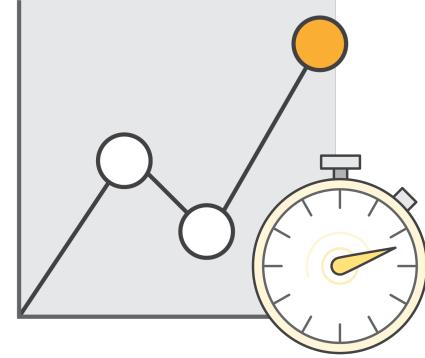


Store and analyze all of your data, from all of your sources, in one centralized location.

# Benefits of a Data Lake – Quick Ingest

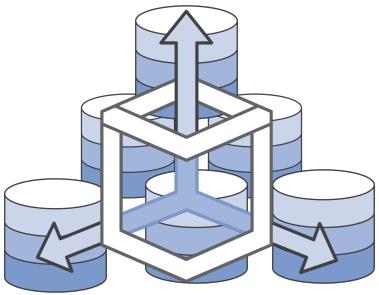


"How can I collect data quickly from various sources and store it efficiently?"

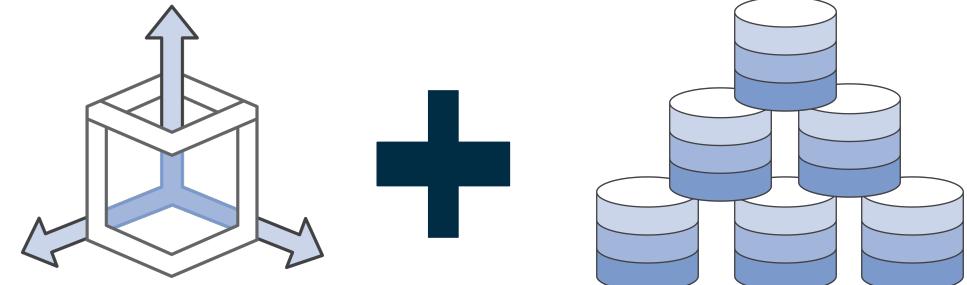


Quickly ingest data without needing to force it into a pre-defined schema.

# Benefits of a Data Lake – Storage vs Compute

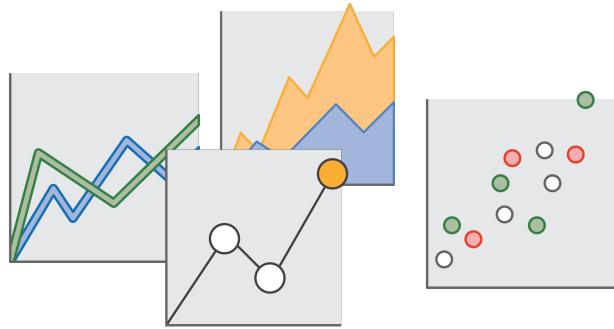


"How can I scale up with the volume of data being generated?"

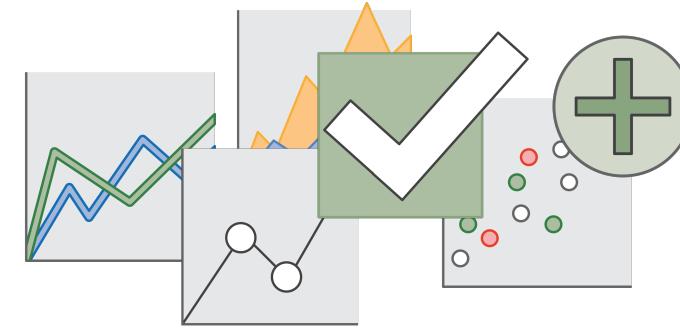


Separating your storage and compute allows you to scale each component as required

# Benefits of a Data Lake – Schema on Read



"Is there a way I can apply multiple analytics and processing frameworks to the same data?"



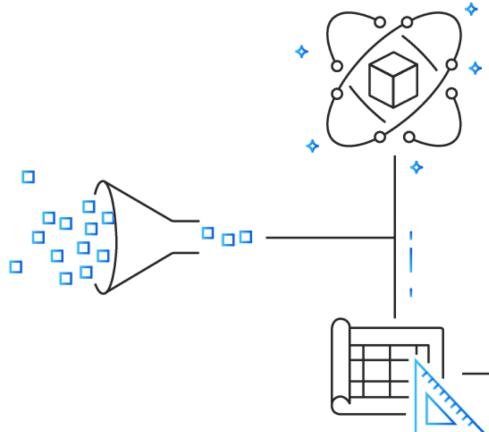
A Data Lake enables ad-hoc analysis by applying schemas on read, not write.

# Building a Data Lake on AWS

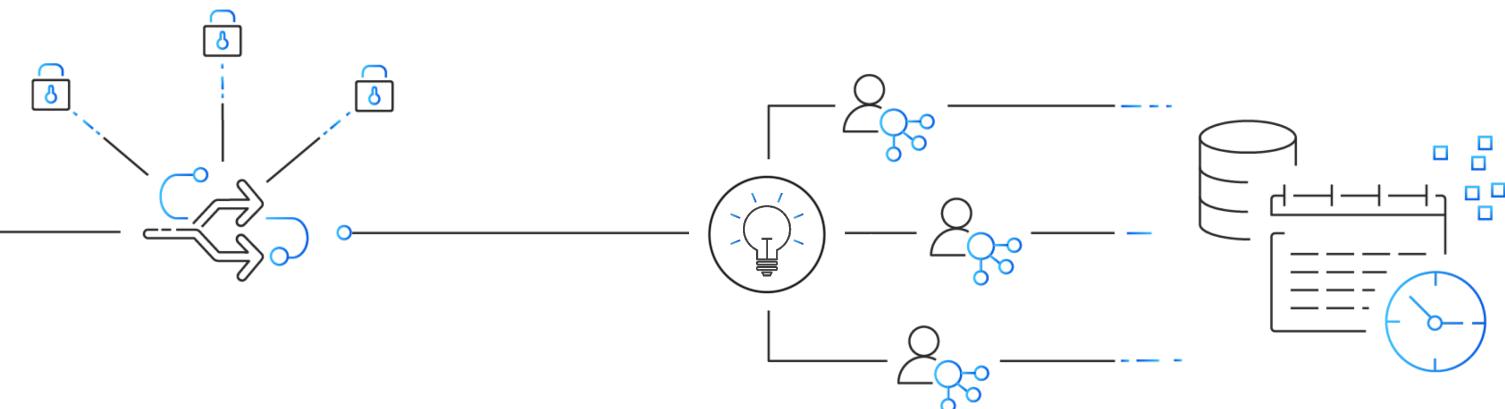
# Why AWS?

Implementing a Data Lake architecture requires a **broad set of tools and technologies** to serve an increasingly **diverse set of applications and use cases**.

## 1 Set up storage



## Typical steps for building a data lake



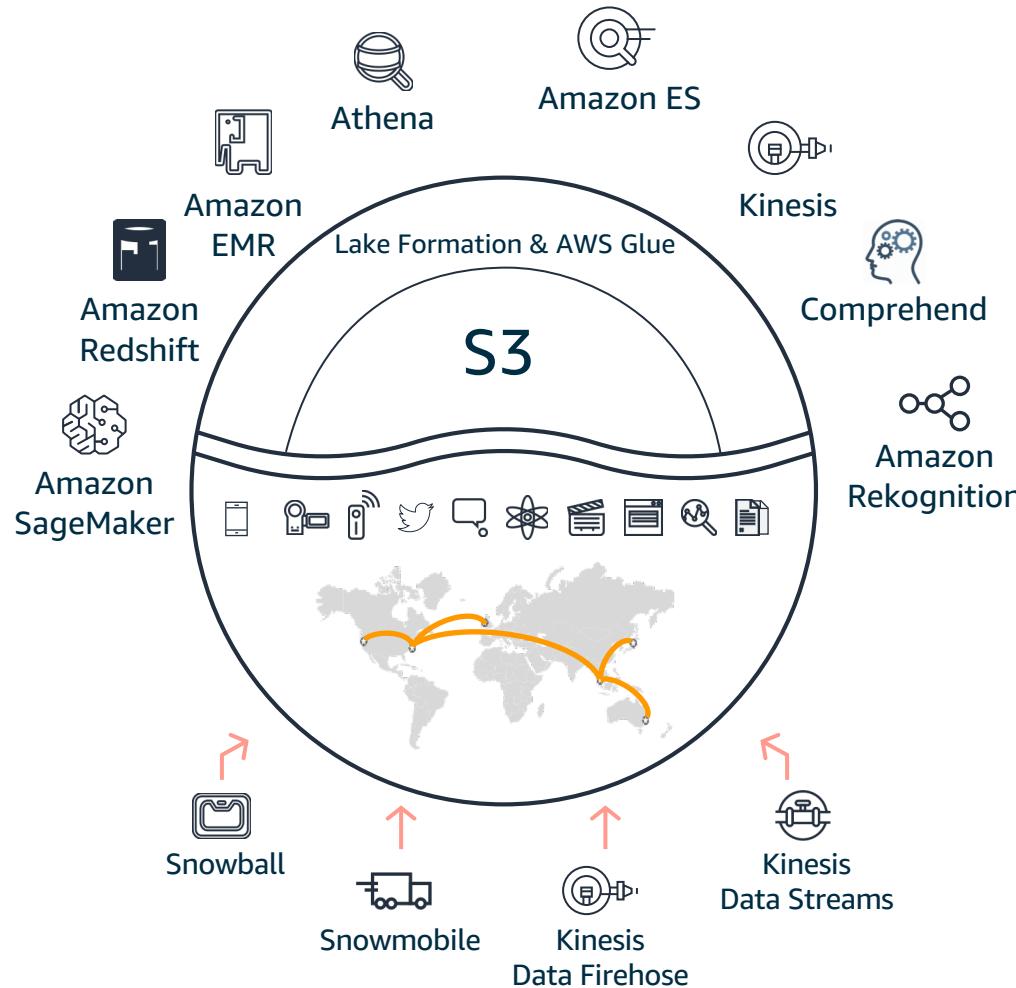
## 2 Move data

## 3 Cleanse, prep, and catalog data

## 4 Configure and enforce security and compliance policies

## 5 Make data available for analytics

# Robust data lake infrastructure with AWS



Durable and available; exabyte scale

Secure, compliant, auditable

Object-level controls for fine-grained access

Fast performance by retrieving subsets of data

Decoupling of compute and storage

On-demand resources, tiering, cost choices

# Why Amazon S3 for a Data Lake?



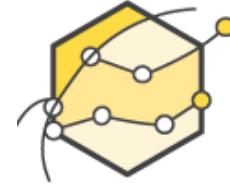
## Durable

Designed for **11 9s** of durability



## Available

Designed for **99.99%** availability



## High performance

- Multiple upload
- Range GET



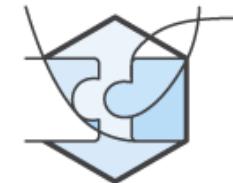
## Easy to use

- Simple REST API
- AWS SDKs
- Read-after-create consistency
- Event notification
- Lifecycle policies



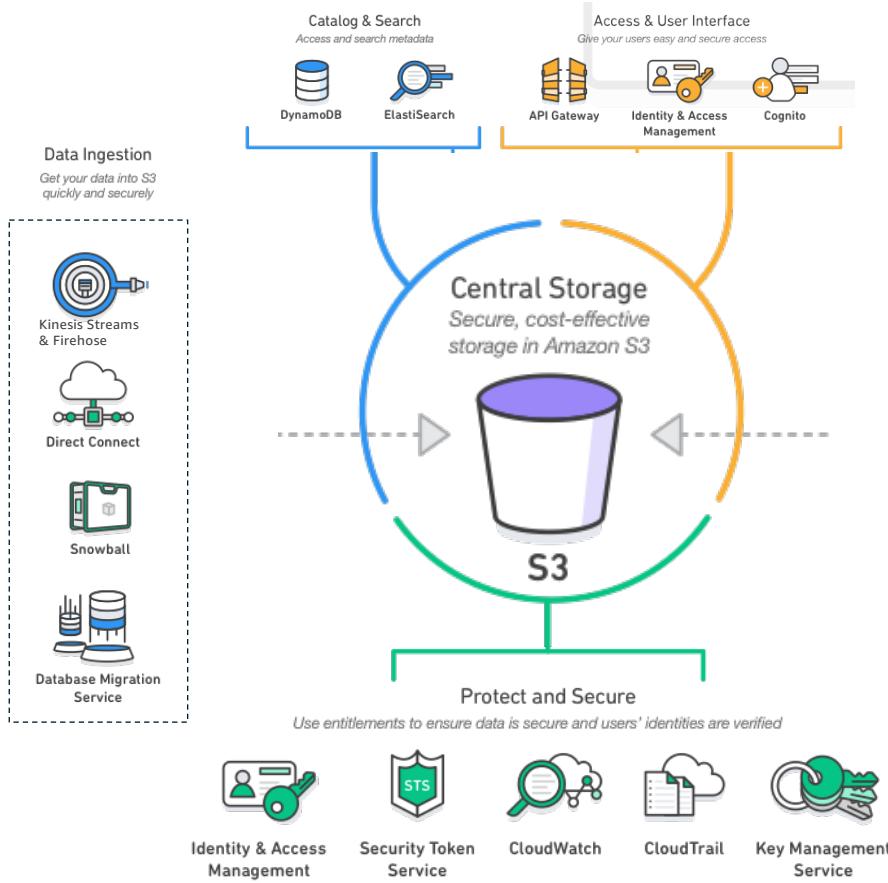
## Scalable

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



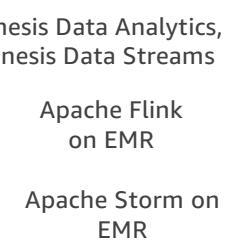
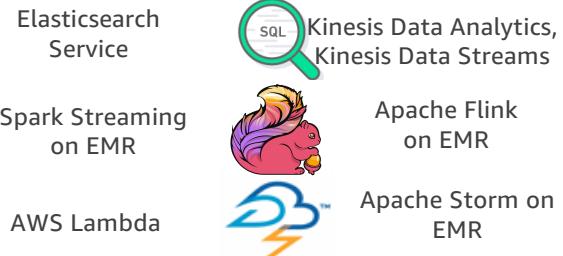
## Integrated

- Amazon EMR
- Amazon Redshift
- Amazon DynamoDB
- Amazon SageMaker
- Many more



## Processing & Analytics

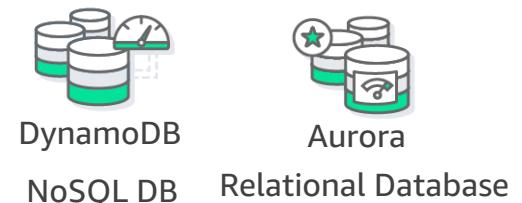
### Real-time



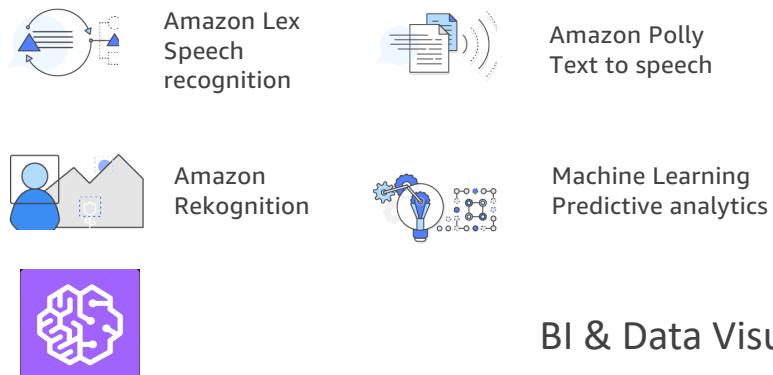
### Analytics



### Transactional & RDBMS



### AI & Predictive



### BI & Data Visualization



# What can you do with a Data Lake?

# Query Directly with Amazon Athena

Athena   **Query Editor**   Saved Queries   History   Data sources   Workgroup : primary   Settings   Tutorial   Help   What's new 10+

**Data source** [awsdatacatalog](#) [Connect data source](#)

**Database** ticketdata [Filter tables and views...](#)

**Tables (22)** [Create table](#)

- [mlb\\_data](#)
- [name\\_data](#)
- [nfl\\_data](#)
- [nfl\\_stadium\\_data](#)
- [parquet\\_person](#)
- [parquet\\_sport\\_location](#)
- [parquet\\_sport\\_team](#)
- [parquet\\_sporting\\_event](#)
- [parquet\\_sporting\\_event\\_ticket](#)
- [person](#)
- [player](#)
- [seat](#)
- [seat\\_type](#)
- [sport\\_division](#)
- [sport\\_league](#)
- [sport\\_location](#)
- [sport\\_team](#)
- [sporting\\_event](#)
- [sporting\\_event\\_ticket](#)
- [sporting\\_event\\_ticket\\_1bb4a008b349ed873527a4c2b9f8ac5f](#)
- [ticket\\_purchase\\_hist](#)
- [ticket\\_purchase\\_hist\\_95f83e3d847527d7c4e84a4949d62d2b](#)

**Views (2)** [Create view](#)

- [sporting\\_event\\_info](#)
- [sporting\\_event\\_ticket\\_info](#)

New query 1   New query 2   New query 3   **New query 4** [+](#)

```
1 SELECT "mlb_id", "mlb_name", "mlb_pos", "mlb_team", "mlb_team_long", "bats", "throws", "birth_year", "bp_id", "bref_id", "bref_name", "cbs_id", "cbs_name", "cbs_pos", "espn_id", "espn_name", "espn_pos", "fg_id", "fg_name", "lahman_id", "nfbc_id", "nfbc_name", "nfbc_pos", "retro_id", "retro_name", "debut", "yahoo_id", "yahoo_name", "yahoo_pos", "mlb_depth"
2 FROM "ticketdata"."mlb_data"
3 LIMIT 1000
```

[Run query](#)   [Save as](#)   [Create](#) (Run time: 1.77 seconds, Data scanned: 395.04 KB)   [Format query](#)   [Clear](#)

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

**Results**

	mlb_id	mlb_name	mlb_pos	mlb_team	mlb_team_long	bats	throws	birth_year	bp_id	bref_id	bref_name	cbs_id	cbs_name
1	+5.065600000000000e+05	Alexi Amarista	3B	SD	San Diego Padres	L	R	1989	+5.588900000000000e+04	amarial01	Alexi Amarista	1735053	Alexi Am
2	+4.582100000000000e+05	Alexi Casilla	2B	TB	Tampa Bay Rays	S	R	1984	+4.579900000000000e+04	casilal01	Alexi Casilla	1103724	Alexi Ca
3	+4.683960000000000e+05	Alexi Ogando	P	ATL	Atlanta Braves	R	R	1983	+4.991000000000000e+04	ogandal01	Alexi Ogando	1174266	Alexi O
4	+4.696860000000000e+05	Alfredo Aceves	P	NYY	New York Yankees	R	R	1982	+4.692800000000000e+04	aceveal01	Alfredo Aceves	1638980	Alfredo
5	+4.516280000000000e+05	Alfredo Figaro	P	TEX	Texas Rangers	R	R	1984	+5.245300000000000e+04	figaral01	Alfredo Figaro	1654334	Alfredo
6	+5.540540000000000e+05	Alfredo Gonzalez	C	CWS	Chicago White Sox	R	R	1992	+6.920100000000000e+04			2210083	Alfredo
7	+5.012450000000000e+05	Alfredo Marte	LF	BAL	Baltimore Orioles	R	R	1989	+5.175500000000000e+04	marteal01	Alfredo Marte	1956711	Alfredo
8	+4.305800000000000e+05	Alfredo Simon	P	CIN	Cincinnati Reds	R	R	1981	+4.558100000000000e+04	simonal01	Alfredo Simon	448968	Alfredo
9	+4.553780000000000e+05	Ali Solis	C	LAD	Los Angeles Dodgers	R	R	1987	+5.236200000000000e+04	solisal01	Ali Solis	1946524	Ali Solis
10	+4.888520000000000e+05	Allan Dykstra	1B	TB	Tampa Bay Rays	L	R	1987	+5.781300000000000e+04	dykstal01	Allan Dykstra	1960245	Allan D
11	+5.018000000000000e+05	Allen Craig	1B	BOS	Boston Red Sox	R	R	1984	+5.104300000000000e+04	craigal01	Allen Craig	1661498	Allen C
12	+5.439030000000000e+05	Allen Webster	P	PIT	Pittsburgh Pirates	R	R	1990	+5.878700000000000e+04	webstal01	Allen Webster	1799467	Allen W
13	+6.072370000000000e+05	Amir Garrett	P	CIN	Cincinnati Reds	R	L	1992	+7.094600000000000e+04			2053474	Amir Ga
14	+4.448430000000000e+05	Andre Ethier	RF	LAD	Los Angeles Dodgers	L	L	1982	+4.596200000000000e+04	ethiean01	Andre Ethier	490390	Andre E
15	+5.165890000000000e+05	Andre Rienzo	P	MIA	Miami Marlins	R	R	1988	+5.663800000000000e+04	rienzan01	Andre Rienzo	2027388	Andre R
16	+5.927430000000000e+05	Andreton Simmons	SS	LAA	Los Angeles Angels	R	R	1989	+6.711900000000000e+04	simmoan01	Andreton Simmons	1918584	Andreton
17	+4.332170000000000e+05	Andres Blanco	2B	PHI	Philadelphia Phillies	S	R	1984	+3.283500000000000e+04	blancan01	Andres Blanco	392116	Andres Bla

# Analyze with Hadoop on Amazon EMR

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

**Step 1: Software and Steps**

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

---

### Software Configuration

Vendor  Amazon  MapR

Release   

<input checked="" type="checkbox"/> Hadoop 2.6.0	<input checked="" type="checkbox"/> Hive 1.0.0	<input type="checkbox"/> Mahout 0.11.0
<input checked="" type="checkbox"/> Zeppelin-Sandbox 0.5.5	<input type="checkbox"/> Hue 3.7.1	<input checked="" type="checkbox"/> Spark 1.5.2
<input type="checkbox"/> Ganglia 3.6.0	<input type="checkbox"/> Presto-Sandbox 0.125	<input type="checkbox"/> Oozie-Sandbox 4.2.0
<input type="checkbox"/> Pig 0.14.0		

---

**Scala**

```
val movieLensHomeDir = "s3://emr.examples/movieLens/"

val movies = sc.textFile(movieLensHomeDir + "movies.dat").map { line =>
    val fields = line.split("::")
    // format: (movieId, movieName)
    (fields(0).toInt, fields(1))
}.collect.toMap

val ratings = sc.textFile(movieLensHomeDir + "ratings.dat").map { line =>
    val fields = line.split("::")
    // format: (timestamp % 10, Rating(userId, movieId, rating))
    (fields(3).toLong % 10, Rating(fields(0).toInt, fields(1).toInt, fields(2).toDouble))
}
```

 **Zeppelin** Notebook • Interpreter Connected

# Welcome to Zeppelin.

This is a live tutorial, you can run the code yourself. (Shift+Enter to Run)

Took 1 seconds.

### Load Data Into Table

```
Import org.apache.commons.io.IOUtils
Import java.net.URL
Import java.nio.charset.Charset
bankText: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[32] at parallelize at <console>:65
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string, marital: string, education: string, balance: int]
```

Took 5 seconds. (outdated)

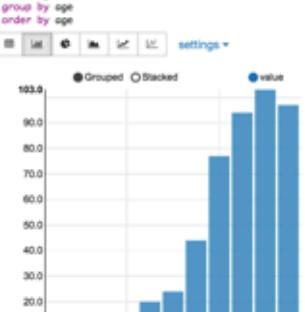
**FINISHED**   

### Naql

```
select age, count(1) value
from bank
where age < 38
group by age
order by age
```

**FINISHED**   

settings ▾



Took 5 seconds. (outdated)

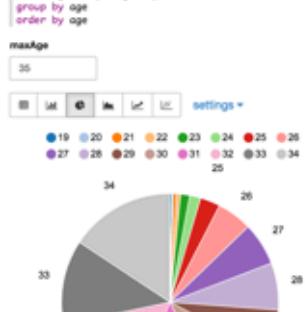
### Naql

```
select age, count(1) value
from bank
where age < ${mixAge-38}
group by age
order by age
```

**FINISHED**   

mixAge  
36

settings ▾



Took 1 seconds. (outdated)

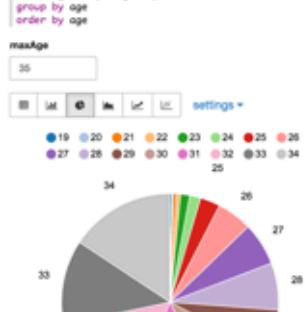
### Naql

```
select age, count(1) value
from bank
where marital="${marital}single,separated,marrying"
group by age
```

**FINISHED**   

marital  
single

settings ▾



Took 1 seconds. (outdated)

### Naql

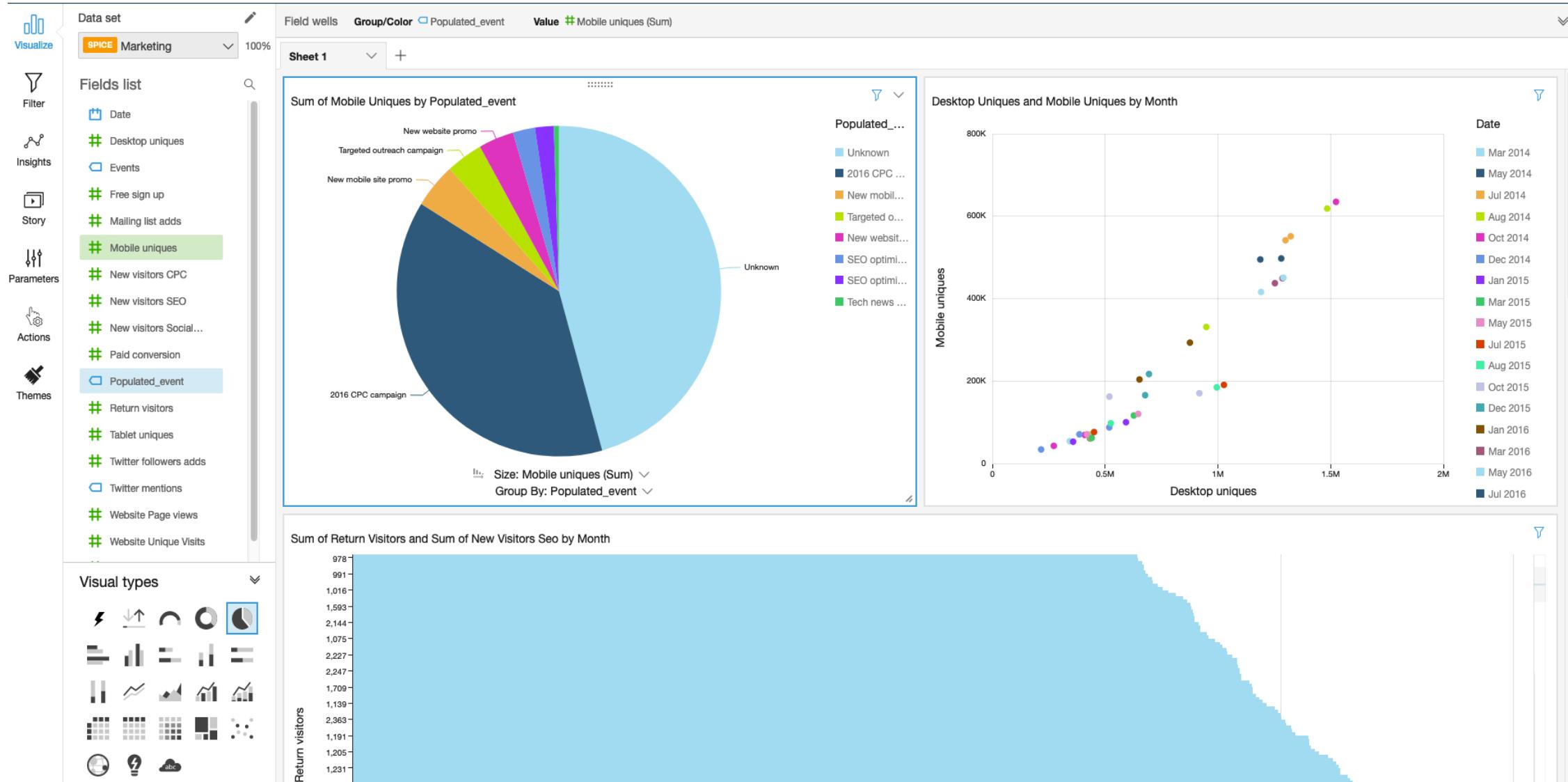
```
select age, count(1) value
from bank
```

**FINISHED**   



Took 1 seconds. (outdated)

# Create Visualizations with Amazon QuickSight



# Train ML Models with Amazon SageMaker

Amazon SageMaker X

Amazon SageMaker > Notebook instances > Create notebook instance

## Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more ↗](#)

### Notebook instance settings

Notebook instance name  
ML-Notebook  
Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type  
ml.t2.medium

Elastic Inference [Learn more ↗](#)  
ml.eia1.medium

▲ Additional configuration

### Permissions and encryption

IAM role  
Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20200113T115452

Root access - optional

Enable - Give users root access to the notebook  
 Disable - Don't give users root access to the notebook  
Lifecycle configurations always have root access

Encryption key - optional  
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

testCMK

# Create a Central Data Catalog with AWS Glue

AWS Glue

Data catalog

Databases

**Tables**

Connections

Crawlers

Classifiers

Settings

ETL

Workflows

Jobs

ML Transforms

Triggers

Dev endpoints

Notebooks

Security

Security configurations

Tutorials

Add crawler

Explore table

Add job

Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Add tables ▾

Action ▾

Filter by attributes or search by keyword

Save view

Showing: 1 - 60

Name	Database	Location	Classification	Last updated
2018	digitwinschema	s3://digitwin-mock-data/2018/	csv	11 November 2018 9:24 PM UTC-8
areas_json	legislators	s3://awsglue-datasets/examples/us-le...	json	8 May 2019 5:16 PM UTC-7
countries_json	legislators	s3://awsglue-datasets/examples/us-le...	json	8 May 2019 5:16 PM UTC-7
digitwin_mock_data	digitwinschema	s3://digitwin-mock-data/	csv	11 November 2018 8:14 AM UTC-8
events_json	legislators	s3://awsglue-datasets/examples/us-le...	json	8 May 2019 5:16 PM UTC-7
glueanddb	dbtos3	s3://glueanddb/	parquet	7 May 2019 6:39 PM UTC-7
green_abe_12345	vendordata	s3://abe-12345/	csv	4 June 2019 10:45 AM UTC-7
memberships_json	legislators	s3://awsglue-datasets/examples/us-le...	json	8 May 2019 5:16 PM UTC-7
mlb_data	ticketdata	s3://dmslab-student-dmslabs3bucket-...	csv	30 May 2019 9:37 AM UTC-7
name_data	ticketdata	s3://dmslab-student-dmslabs3bucket-...	csv	30 May 2019 9:37 AM UTC-7
nfl_data	ticketdata	s3://dmslab-student-dmslabs3bucket-...	csv	30 May 2019 9:37 AM UTC-7
nfl_stadium_data	ticketdata	s3://dmslab-student-dmslabs3bucket-...	csv	30 May 2019 9:37 AM UTC-7
nytaxidata_saursh	lakeformationdata	s3://nytaxidata-saursh/	csv	5 April 2019 7:42 PM UTC-7
organizations_json	legislators	s3://awsglue-datasets/examples/us-le...	json	8 May 2019 5:16 PM UTC-7
parquet_person	ticketdata	s3://dmslab-student-dmslabs3bucket-...	parquet	30 May 2019 3:33 PM UTC-7
parquet_sport_location	ticketdata	s3://dmslab-student-dmslabs3bucket-...	parquet	30 May 2019 3:33 PM UTC-7
parquet_sport_team	ticketdata	s3://dmslab-student-dmslabs3bucket-...	parquet	30 May 2019 3:33 PM UTC-7

# Load into Downstream Services



## Amazon Redshift

Run complex analytic queries against petabytes of structured data



## Amazon Aurora

A MySQL and PostgreSQL compatible relational database built for the cloud



## Amazon DynamoDB

A NoSQL database service that delivers consistent, single-digit millisecond latency at any scale.



## Amazon Elasticsearch

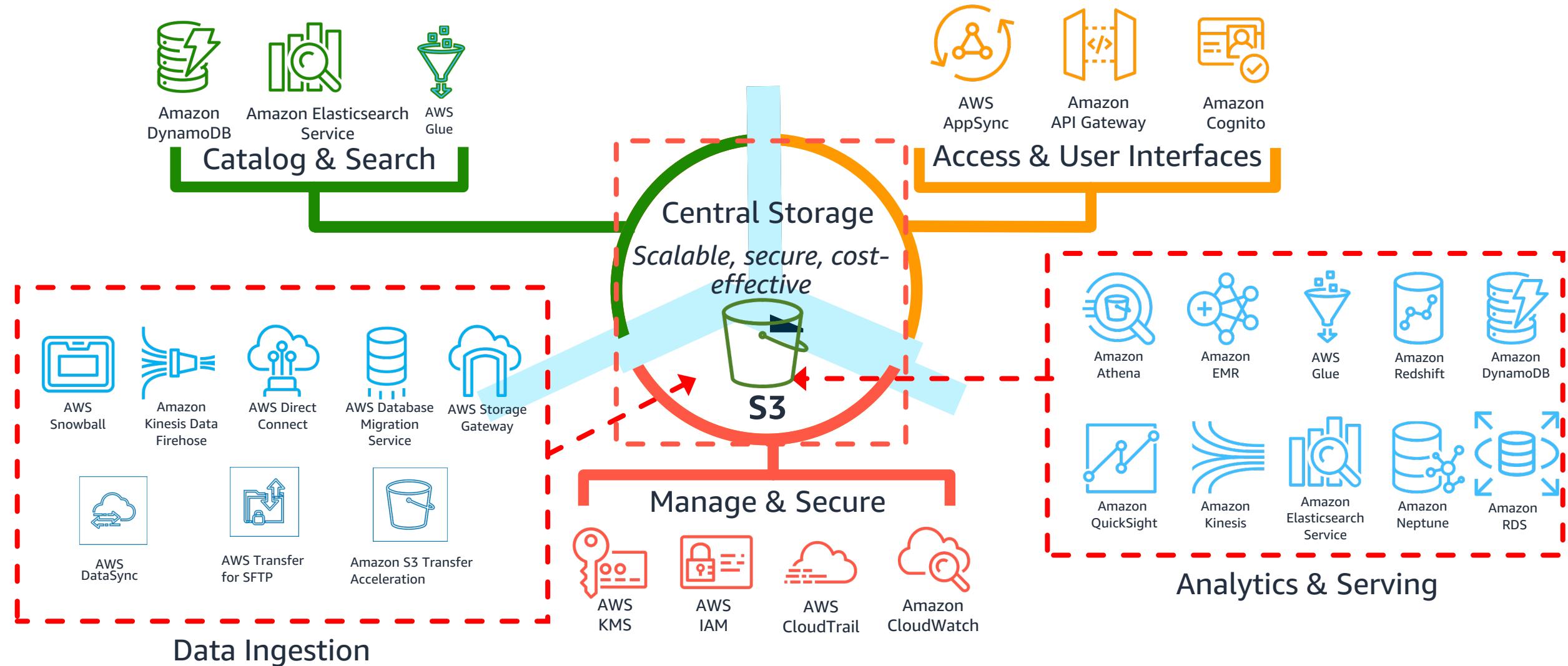
Delivers Elasticsearch's real-time analytics capabilities alongside the availability, scalability, and security that production workloads require.



## Amazon SageMaker

fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models quickly

# Today's Focus



# Thanks! You