

# **Immersion Day – Building a Data Lake on AWS**

Adebimpe (Bims) Daniells

Nov 6, 2019

# Learning Objectives

- Learn the steps to build Data Lake on AWS
- Perform Data Cataloguing, Querying & Visualization
- Understand how AWS analytics fit together
- Machine Learning Integration with Data Lake

# Workshop Agenda

1. Overview of the Data Lake
2. Hydrating the Data Lake
3. Lab – Hydrating the Data Lake
4. Working Within the Data Lake
5. Lab - Querying the Data Lake with Amazon Athena and Amazon QuickSight
6. Consuming the Data Lake – Reporting, Analytics, Machine Learning
7. Demo - Machine Learning with Amazon SageMaker

Data is a strategic asset  
for every organization

66 The world's most valuable  
resource is  
**no longer oil, but data.\*** 99





There is **more data** than people think

**Data**

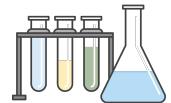
grows  
**>10x**  
every 5 years

**Data platforms need to**

live for  
**15**  
years

scale  
**1,000x**

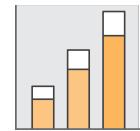
\* IDC, Data Age 2015: The Evolution of Data to Life-Critical Don't Focus on Big Data, Focus on the Data That's Big, April 2017



Data Scientists



Business Users



Analysts



Applications

Secure

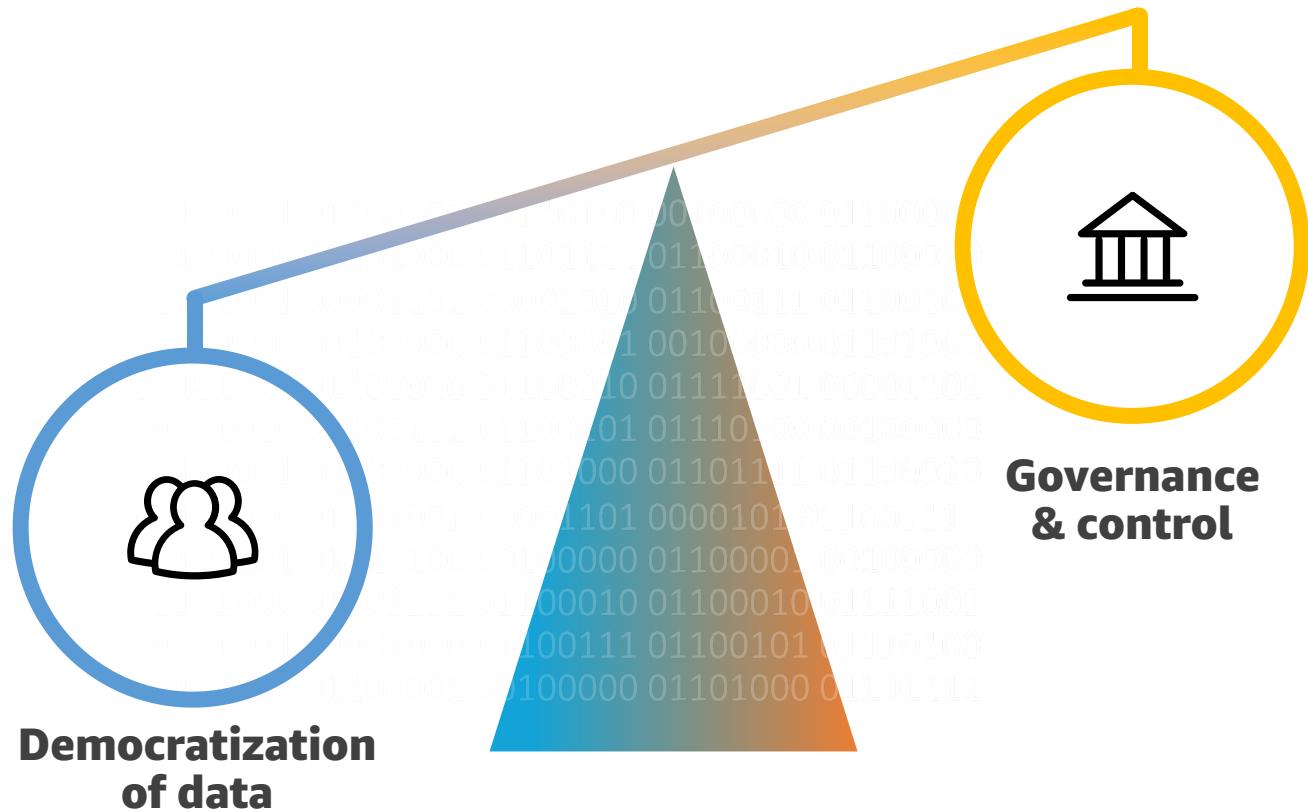
Real time

Flexible

Scalable

There are **more** people accessing data

And **more** requirements for making data available



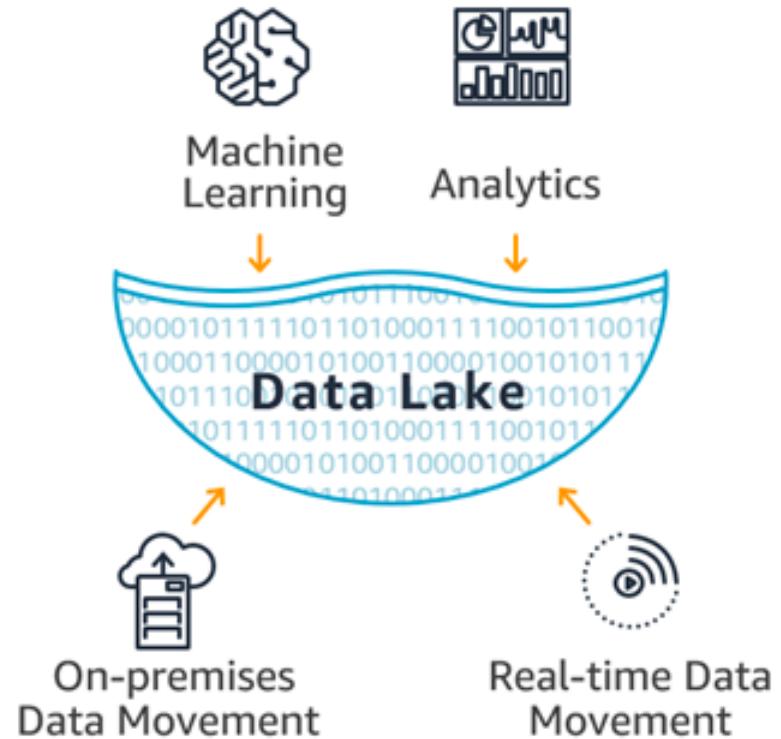
**There are more  
people working  
with data than ever  
before**

---

How do I provide democratized access to data to enable informed decisions while at the same time enforce data governance and prevent mismanagement of the data?

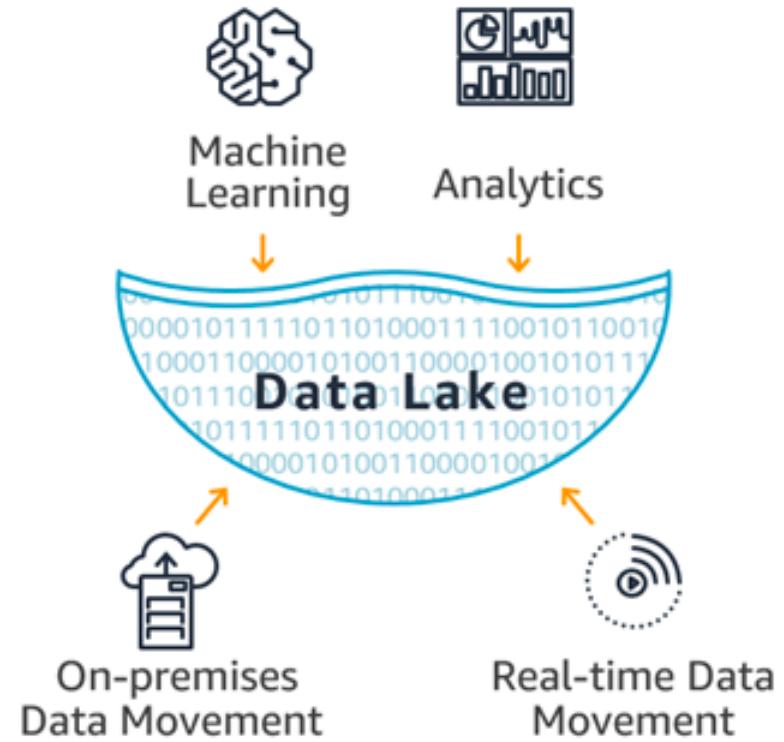
# What is a Data Lake?

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

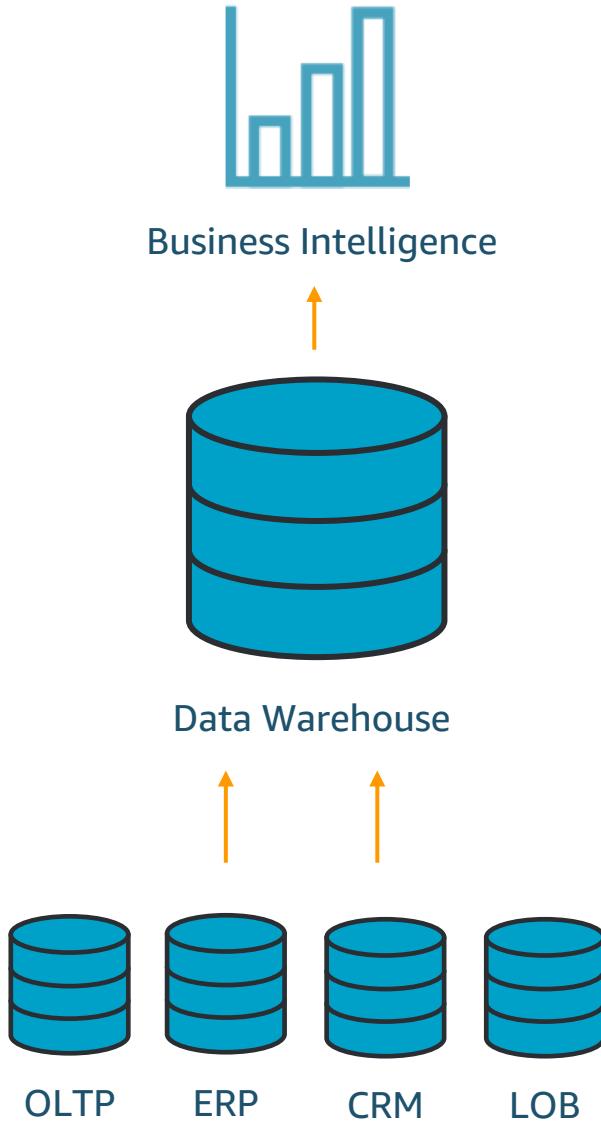


# Why a Data Lake?

- 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

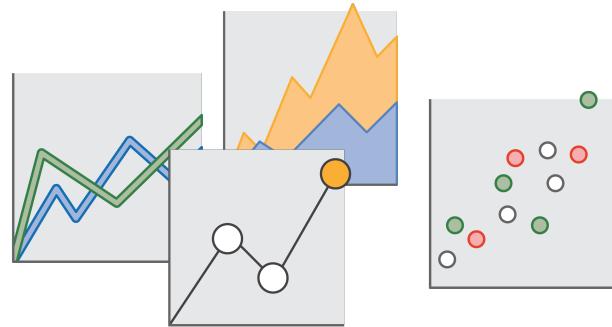
Large Initial Capex + \$\$K / TB/ Year

# Data lakes help you cost-effectively scale

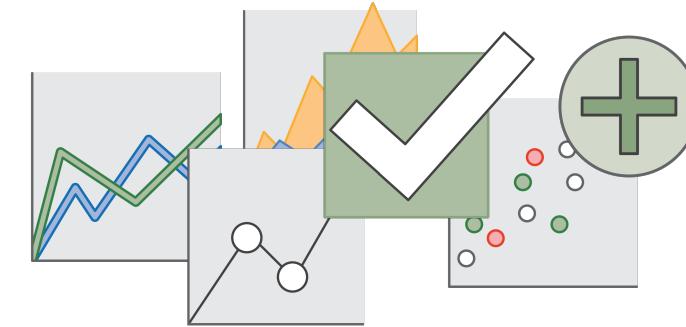


- Extends or evolves their data warehouses
- Durable and available; exabyte scale
- Secure, compliant, auditable
- Run any type of analytics from DW to predictive
- Decoupling of compute and storage
- On-demand resources, tiering, cost choices

# Benefits of a Data Lake – Schema on Read



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



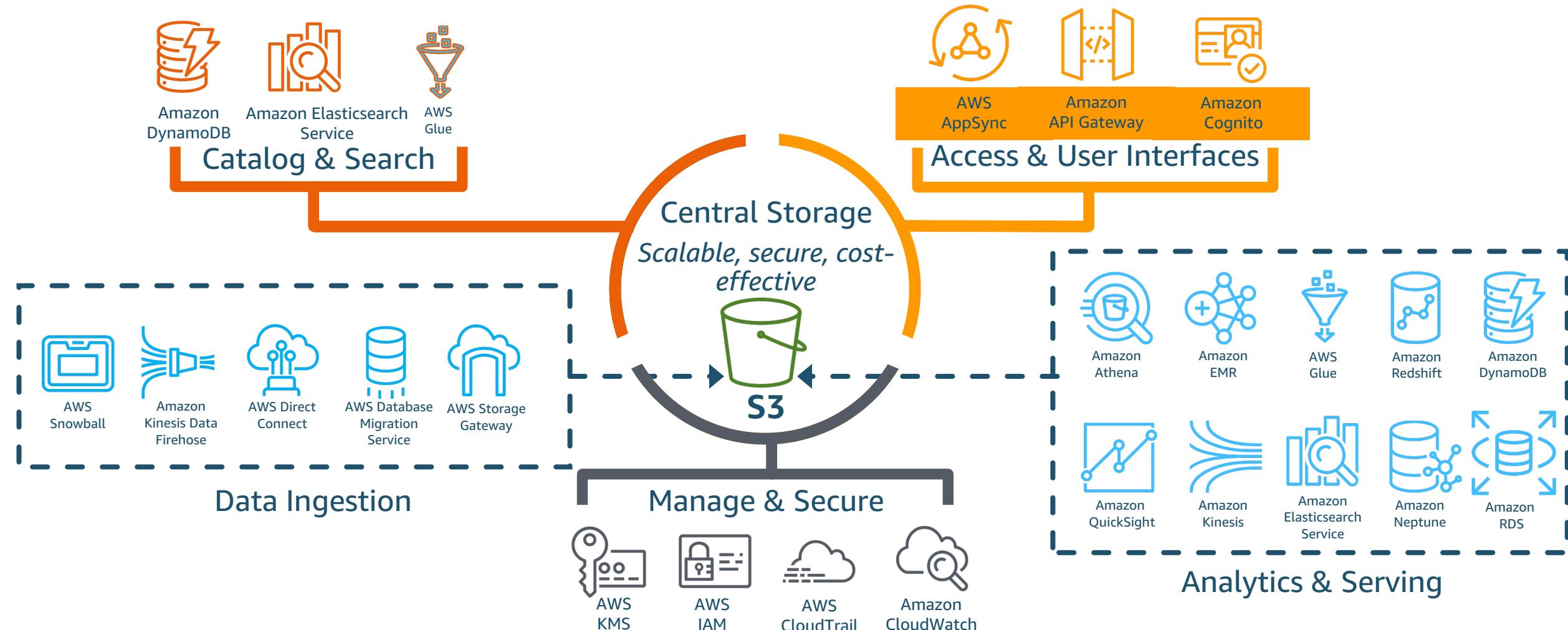
A Data Lake enables ad-hoc analysis by applying schemas on read, not write.

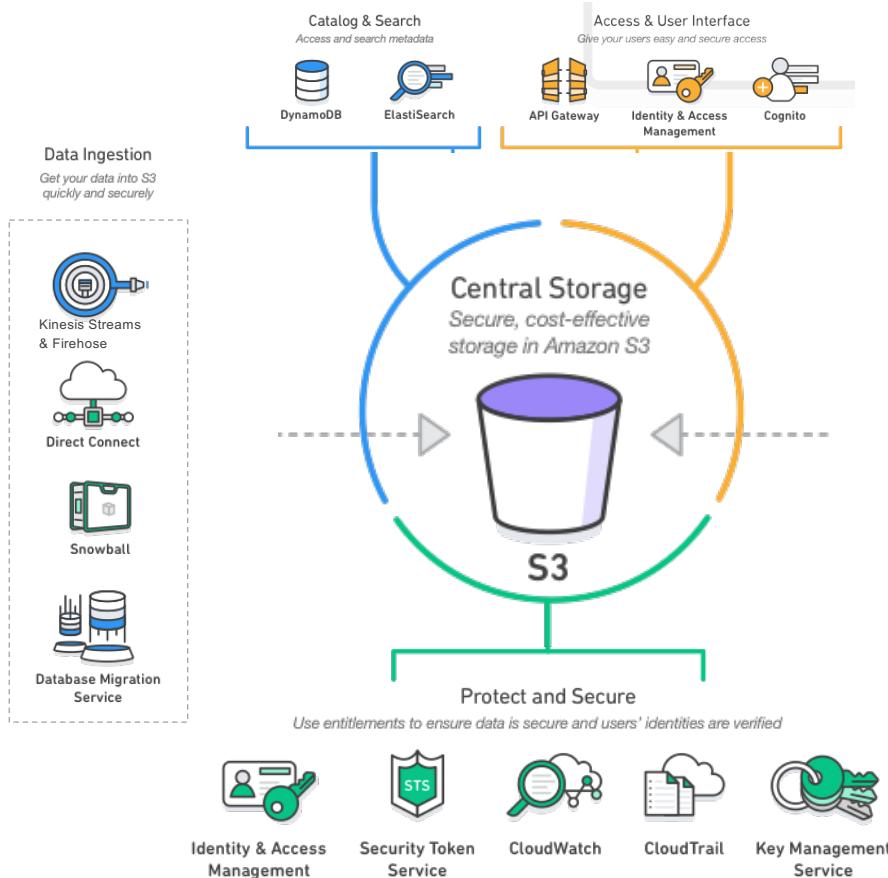
# Building a Data Lake on AWS

# 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**.

# Data Lake on AWS





## Processing & Analytics

### Real-time

- Elasticsearch Service
- Kinesis Data Analytics, Kinesis Data Streams
- Apache Flink on EMR
- Apache Storm on EMR
- Spark Streaming on EMR
- AWS Lambda

### AI & Predictive

- Amazon Lex Speech recognition
- Amazon Polly Text to speech
- Amazon Rekognition
- Machine Learning Predictive analytics
- SageMaker

### Analytics

- EMR Hadoop, Spark, Presto
- Redshift Data Warehouse
- Athena Query Service

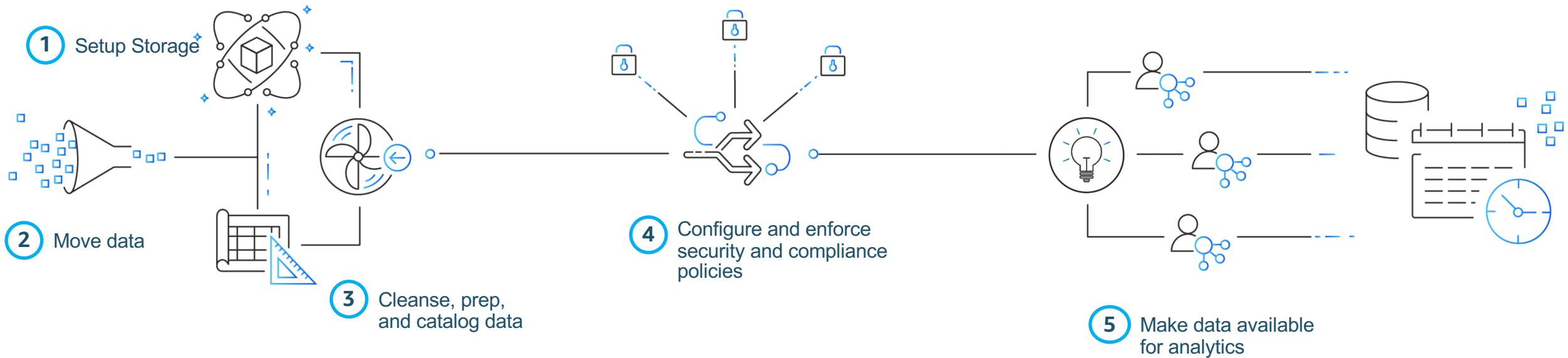
### Transactional & RDBMS

- DynamoDB
- NoSQL DB
- Aurora
- Relational Database

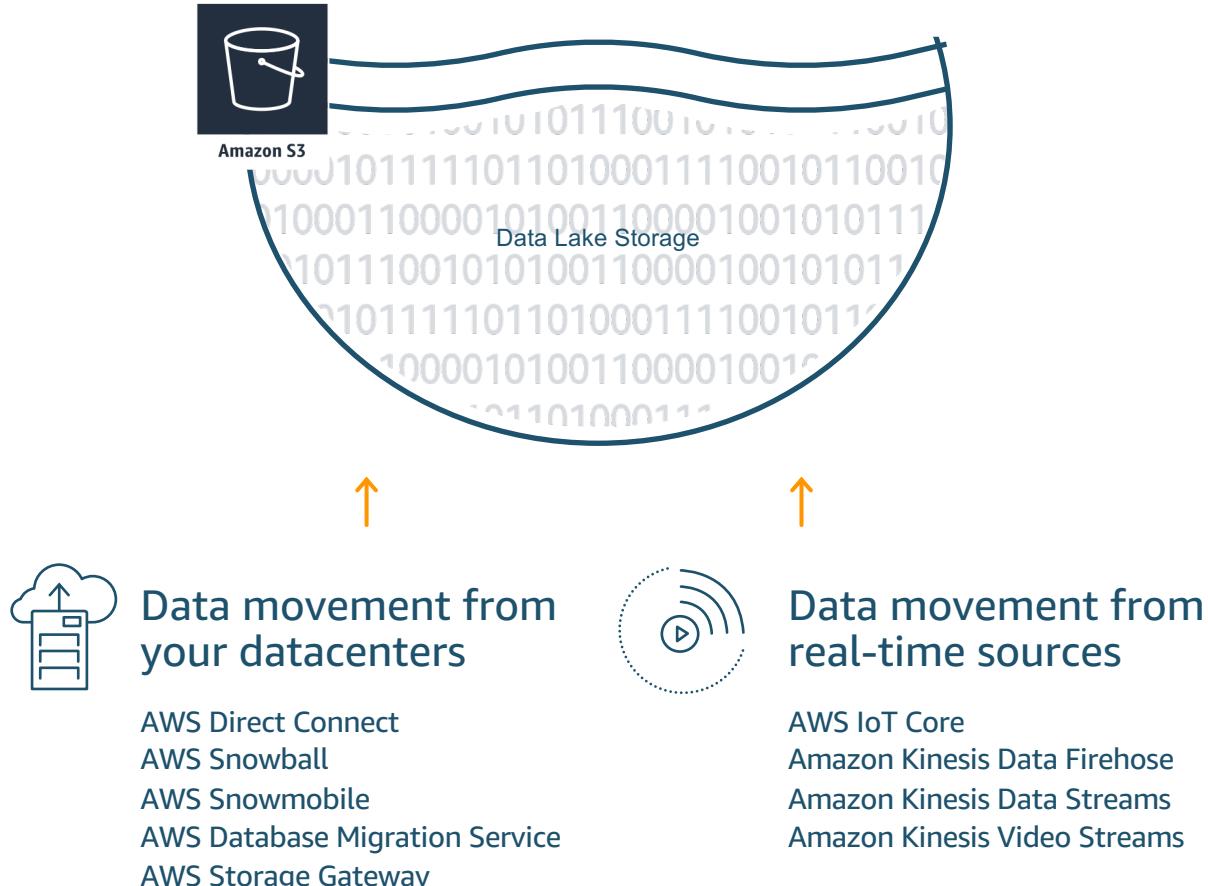
### BI & Data Visualization

- Amazon QuickSight
- Tableau
- MicroStrategy
- kibana

# Typical steps of building a data lake



# Ways to move data into the Data Lake



## Data movement from on-premises datacenters

Dedicated Network connection

Secure appliances

Ruggedized Shipping Container

Database migration

Gateway that lets applications write to the cloud

## Data movement from real-time sources

Connect devices to AWS

Real-time data streams

Real-time video streams

# Why Amazon S3 for a Data Lake?



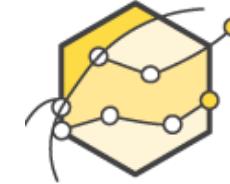
## Durable

Designed for **11 9s** of durability



## Available

Designed for **99.99%** availability



## High performance

- Multiple upload
- Range GET



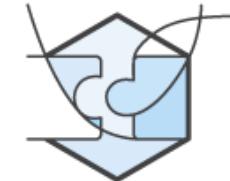
## Easy to use

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



## Scalable

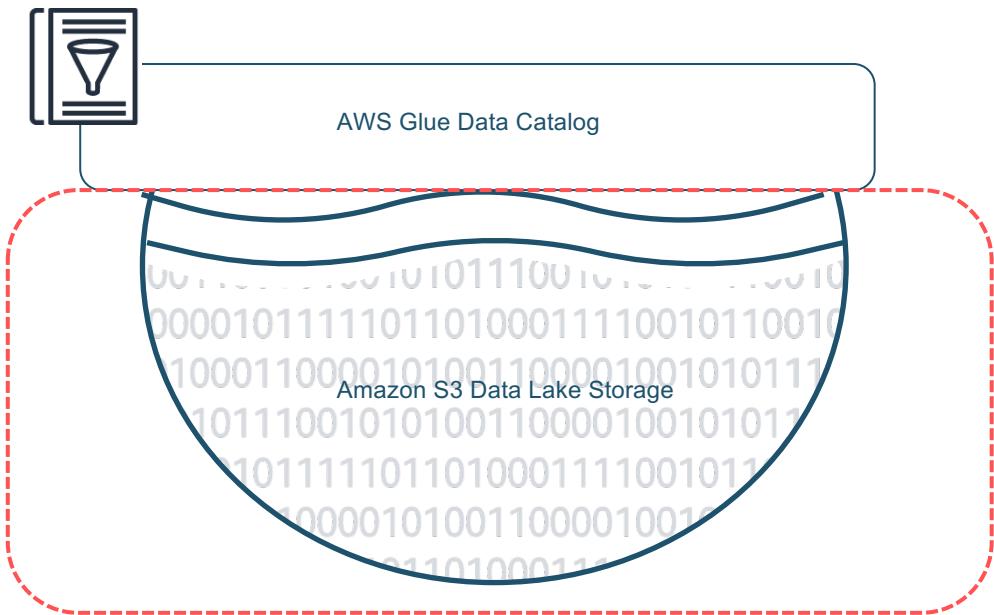
- Store as much as you need
- Scale storage and compute independently
- No minimum usage commitments



## Integrated

- Amazon EMR
- Amazon Redshift
- Amazon DynamoDB
- Hive
- Spark
- Presto

# Secure data access



By default, all Amazon S3 buckets and objects are private

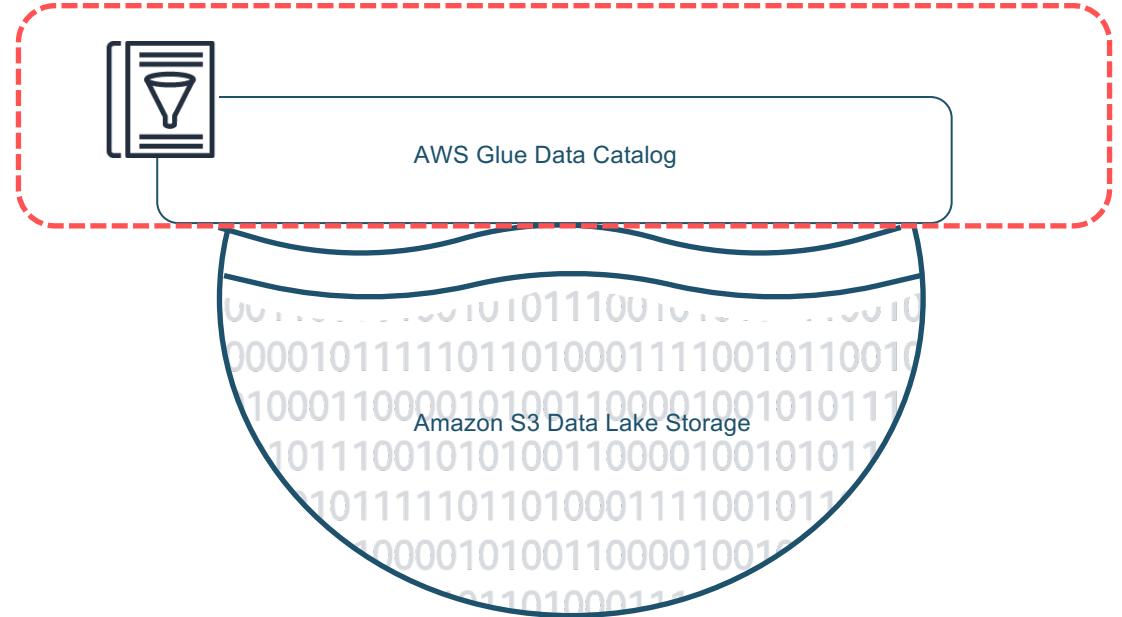
Secure data access with a combination of resource-based policies such as Amazon S3 bucket policies and IAM user policies

Use AWS KMS to enable client and server-side encryption for your data

Use object tagging (Classification = PHI) in conjunction with IAM



# And secure the metadata access



**Identity-based policies (AWS IAM policies)**

Managed by IAM

Attached to IAM principals, e.g. IAM users, IAM roles

**Resource-based policies**

Managed by AWS Glue

One policy per account/catalog, similar to Amazon S3's bucket policies



Required to allow cross account

# Common data categories and use cases



## Relational

Referential integrity,  
ACID transactions,  
schema-on-write



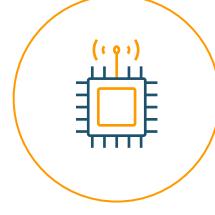
## Key-value

High throughput,  
low-latency reads  
and writes,  
endless scale



## Document

Store documents and  
quickly access  
querying on  
any attribute



## In-memory

Query by key with  
microsecond latency



## Graph

Quickly and easily  
create and navigate  
relationships  
between data



## Time-series

Collect, store, and  
process data  
sequenced by time



## Ledger

Complete,  
immutable, and  
verifiable history  
of all changes to  
application data

### Common use cases

Lift and shift, ERP,  
CRM, finance

Real-time bidding,  
shopping cart, social,  
product catalog,  
customer  
preferences

Content  
management,  
personalization,  
mobile

Leaderboards,  
real-time  
analytics, caching

Fraud detection,  
social networking,  
recommendation  
engine

IoT applications,  
event tracking

Systems  
of record, supply  
chain, health care,  
registrations,  
financial

# Common data categories and use cases



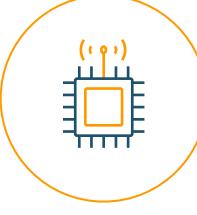
**Relational**



**Key-value**



**Document**



**In-memory**



**Graph**



**Time-series**



**Ledger**



**RDS**



**DynamoDB**



**DocumentDB**



**ElastiCache**



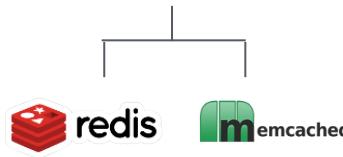
**Neptune**



**Timestream**



**QLDB**



# When to use which service?

Situation	Solution
Existing application	<p>Use your existing engine on RDS</p> <ul style="list-style-type: none"><li>• MySQL → Amazon Aurora, RDS for MySQL</li><li>• PostgreSQL → Amazon Aurora, RDS for PostgreSQL</li><li>• MariaDB → Amazon Aurora, RDS for MariaDB</li><li>• Oracle → Use SCT to determine complexity → Amazon Aurora, RDS for Oracle</li><li>• SQL Server → Use SCT to determine complexity → Amazon Aurora, RDS for SQL Server</li></ul>
New application	<ul style="list-style-type: none"><li>• If you can avoid relational features → DynamoDB</li><li>• If you need relational features → Amazon Aurora</li></ul>
Fraud detection, Knowledge graphs, Network / IT Operations	<ul style="list-style-type: none"><li>• Amazon Neptune</li></ul>
In-memory store/cache	<ul style="list-style-type: none"><li>• Amazon ElastiCache</li></ul>
Time series data	<ul style="list-style-type: none"><li>• Amazon Timestream</li></ul>
Track every application change, crypto verifiable. Have a central trust authority	<ul style="list-style-type: none"><li>• Amazon Quantum Ledger Database (QLDB)</li></ul>
Don't have a trusted central authority	<ul style="list-style-type: none"><li>• Amazon Managed Blockchain</li></ul>
Data Warehouse & BI	<ul style="list-style-type: none"><li>• Amazon Redshift, Amazon Redshift Spectrum, and Amazon QuickSight</li></ul>
Adhoc analysis of data in S3	<ul style="list-style-type: none"><li>• Amazon Athena and Amazon QuickSight</li></ul>
Apache Spark, Hadoop, HBase (needle in a haystack type queries)	<ul style="list-style-type: none"><li>• Amazon EMR</li></ul>

# Building data lakes can still take months

# Build a secure data lake in days with AWS Lake Formation

**Move, store, catalog, and  
clean your data faster**



Move, store, catalog,  
and clean your data faster  
with Machine Learning

**Enforce security policies  
across multiple services**



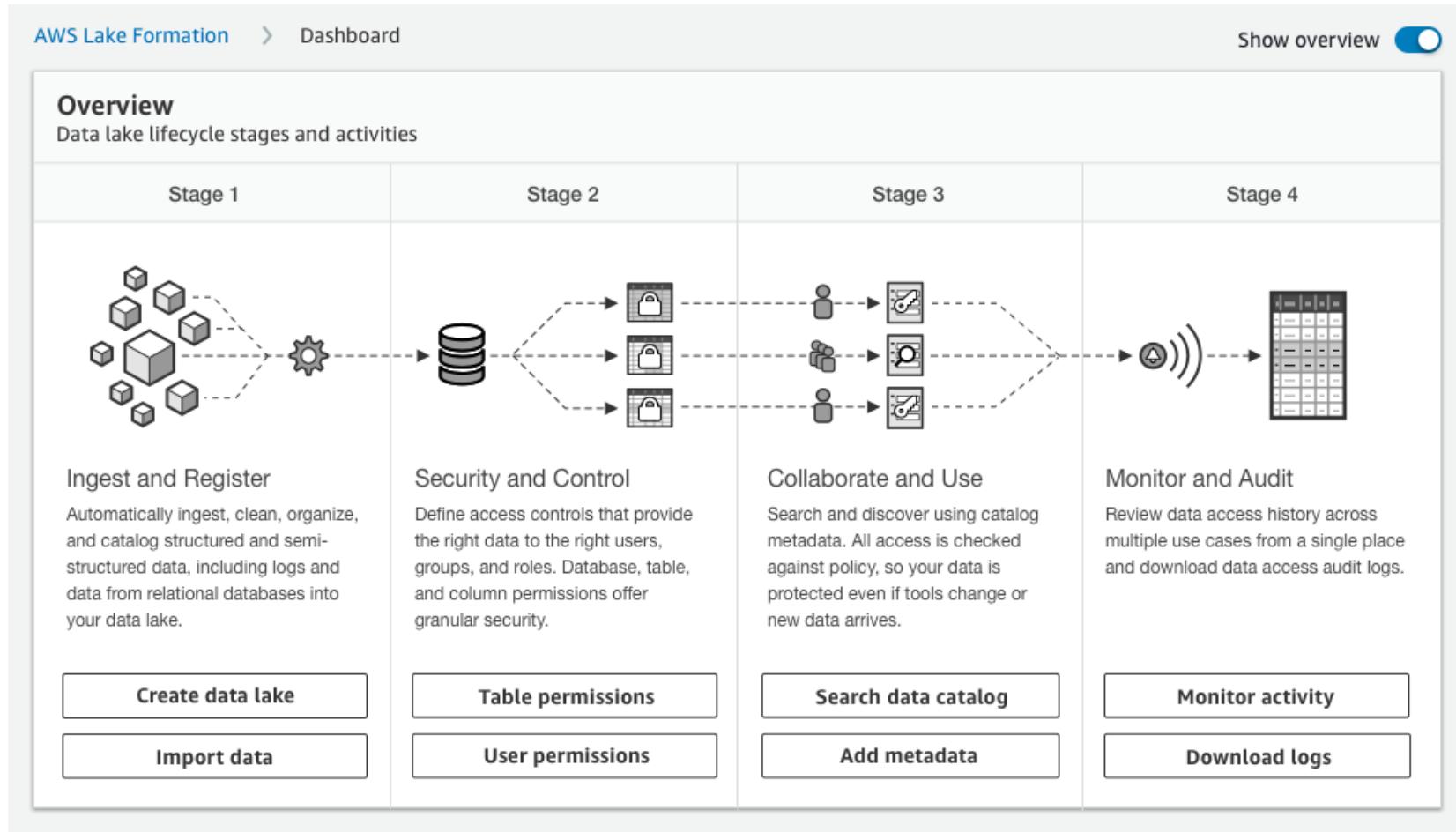
Enforce security policies across  
multiple services

**Gain and manage new  
insights**

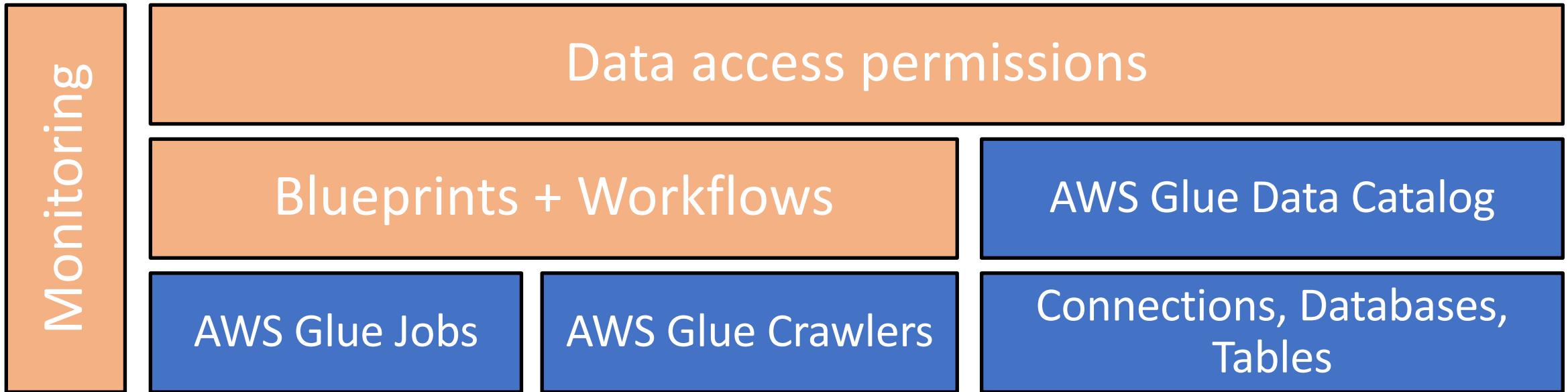


Empower analyst and data  
scientist to gain and manage new  
insights

# How it works



# AWS Lake Formation builds on AWS Glue capabilities



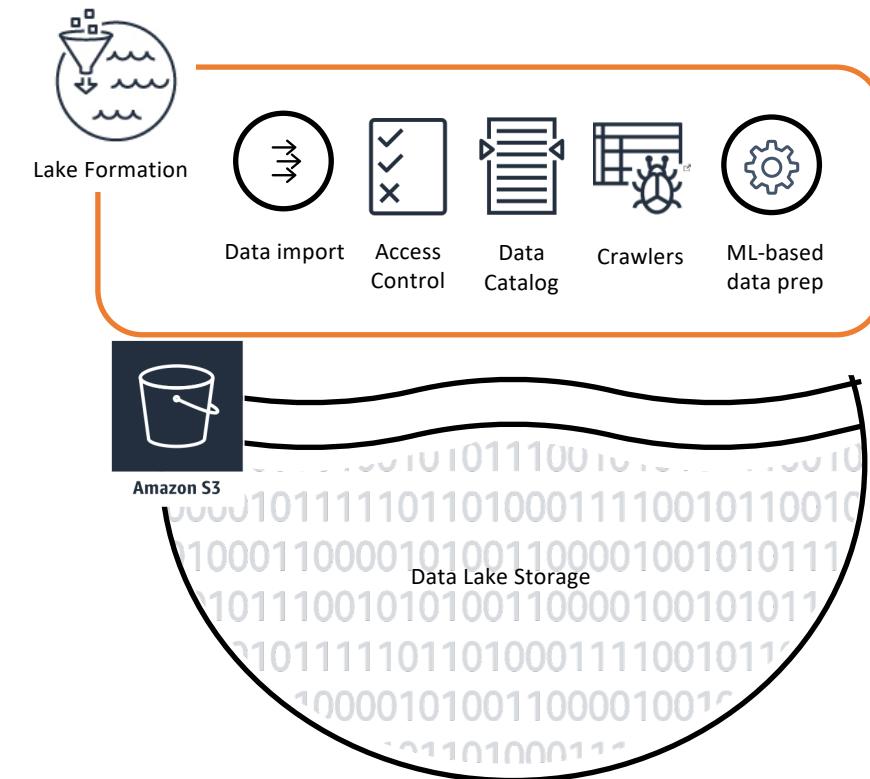
# Register existing data or import new

Amazon S3 forms the storage layer for Lake Formation

Register existing S3 buckets that contain your data

Ask Lake Formation to create required S3 buckets and import data into them

Data is stored in your account. You have direct access to it. No lock-in.



# With blueprints

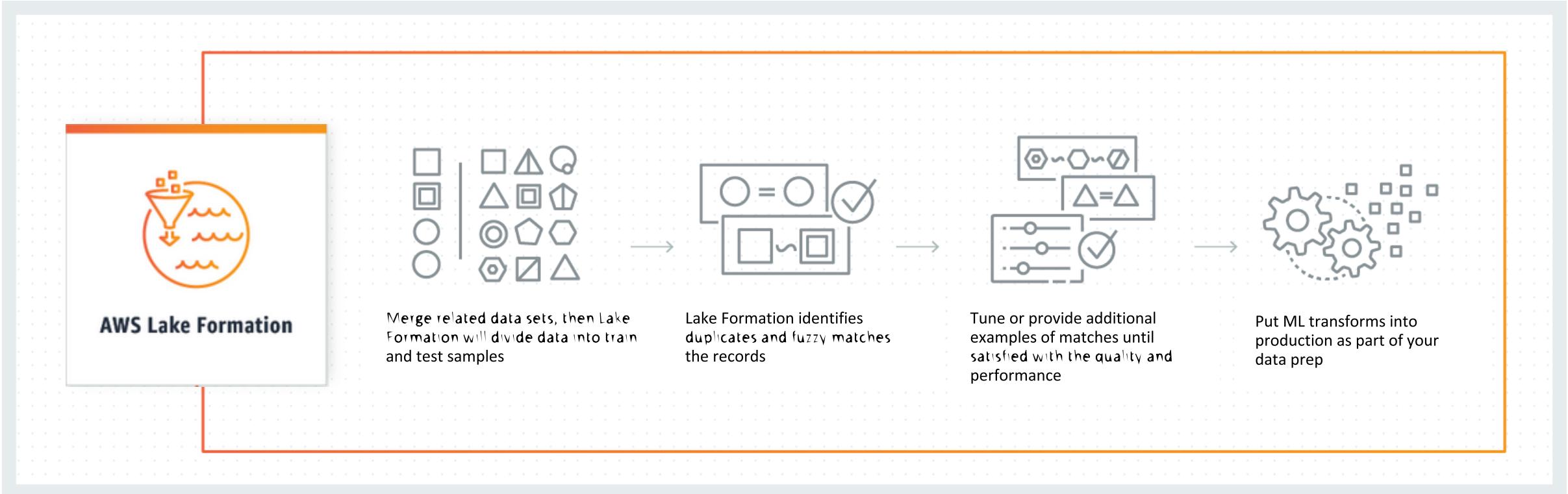
## You

1. Point us to the source
2. Tell us the location to load to in your data lake
3. Specify how often you want to load the data

## Blueprints

1. Discover the source table(s) schema
2. Automatically convert to the target data format
3. Automatically partition the data based on the partitioning schema
4. Keep track of data that was already processed
5. You can customize any of the above

# Easily de-duplicate your data with ML transforms



# Security permissions in Lake Formation

Control data access with simple grant and revoke permissions

Specify permissions on tables and columns rather than on buckets and objects

Easily view policies granted to a particular user

Audit all data access at one place

The screenshot shows the AWS Lake Formation 'Tables' page. A context menu is open over a table named 'reviews'. The menu includes options like 'View data', 'Edit', 'Copy', 'Delete', 'Grant', 'Revoke', and 'Verify permissions'. Below this, a modal window titled 'Grant permissions to table orders' is displayed, allowing users to add IAM users, groups, or roles and select specific permissions to grant.

# What can you do with a Data Lake?

# Query Directly with Amazon Athena

The screenshot shows the Amazon Athena Query Editor interface. At the top, there are tabs for "Athena", "Query Editor" (which is selected), "Saved Queries", "History", and "Catalog Manager". On the right side of the header are links for "Settings", "Tutorial", and "Help".

In the left sidebar, under "DATABASE", the database "sampledb" is selected. Under "TABLES", there is a list of columns from the "elb\_logs" table: timestamp (string), elbname (string), requestip (string), requestport (int), backendip (string), backendport (int), requestprocessingtime (double), backendprocessingtime (double), clientresponsetime (double), elbresponsecode (string), backendresponsecode (string), receivedbytes (bigint), sentbytes (bigint), requestverb (string), url (string), and protocol (string). The "elbname" column is highlighted.

The main area displays a query titled "ELB Select Query" with the description "Sample query to view peak load ELBs during a particular timeframe". The query code is:

```
1 SELECT elbname, count(1) as num
2 FROM sampledb.elb_logs
3 Where elbresponsecode = '200'
4 GROUP BY elbname
5 ORDER BY num DESC limit 10;
```

Below the query, there are buttons for "Run Query", "Save As", "Format Query", and "New Query". A status message indicates "Run time: 1.9 seconds, Data scanned: 826.54KB".

The results section shows a table with one row:

	elbname	num
1	lb-demo	4108

At the bottom right of the results table are download and copy icons.

# 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 emr-4.2.0

<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

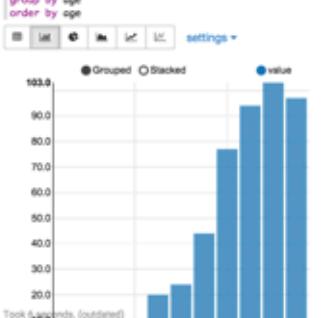
```
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
bankText: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[32] at parallelize at <console>:65
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string, marital: string, education: string, balance: int]
Took 5 seconds. (outdated)
```

FINISHED ▶ ⏪ ⏹ ⏷

### Max Age

```
maxAge
```

settings ▾



Grouped Stacked

Age	Count
18	1
19	1
20	1
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	1
33	1
34	1

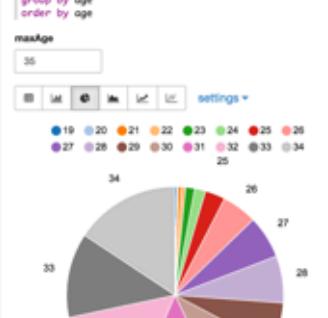
Took 5 seconds. (outdated)

### Marital Status

```
marital
```

single

settings ▾



Marital Status	Count
single	100
divorced	10
married	10
widowed	10

Took 1 seconds. (outdated)

### Value

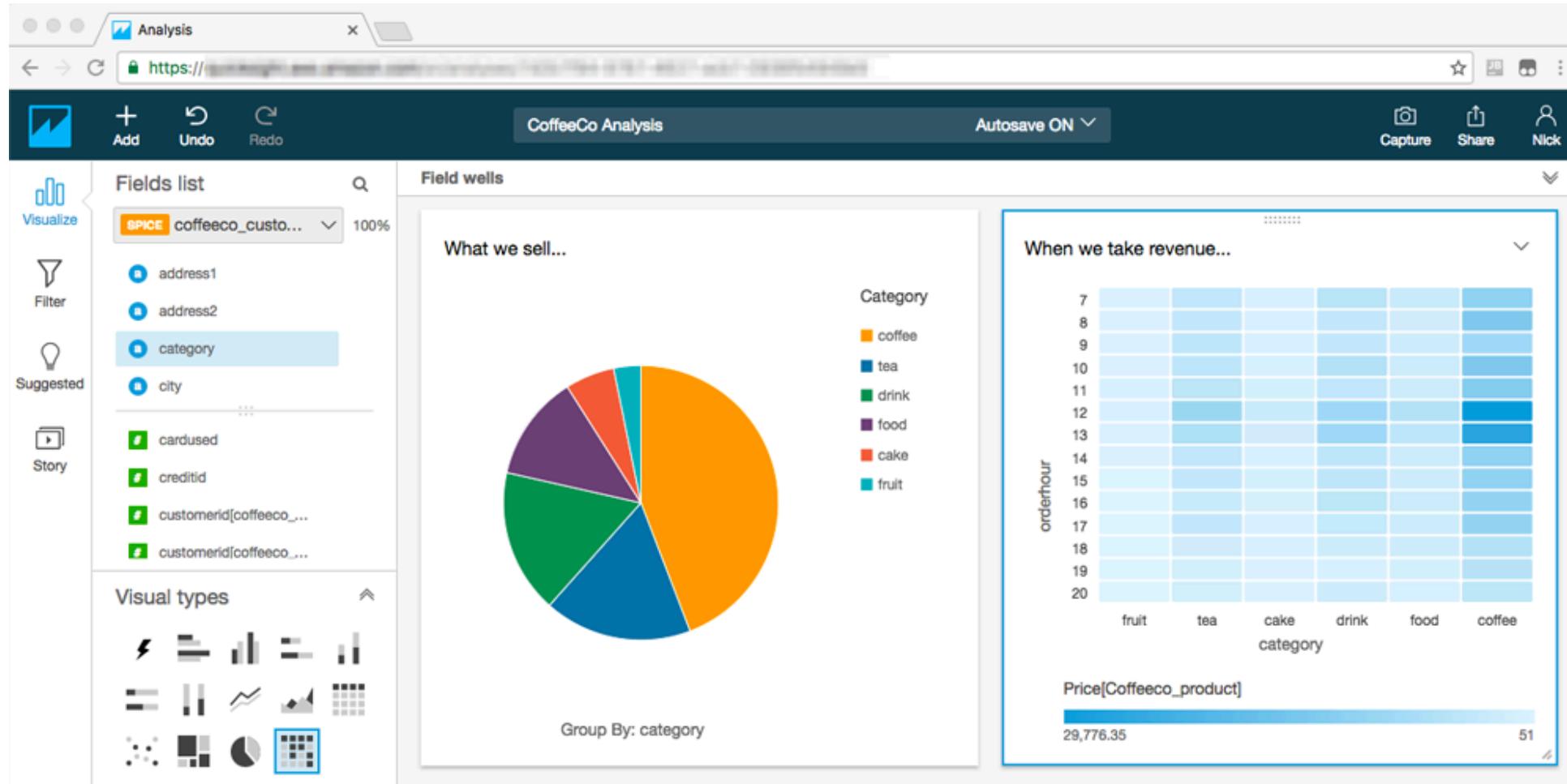
```
value
```

settings ▾

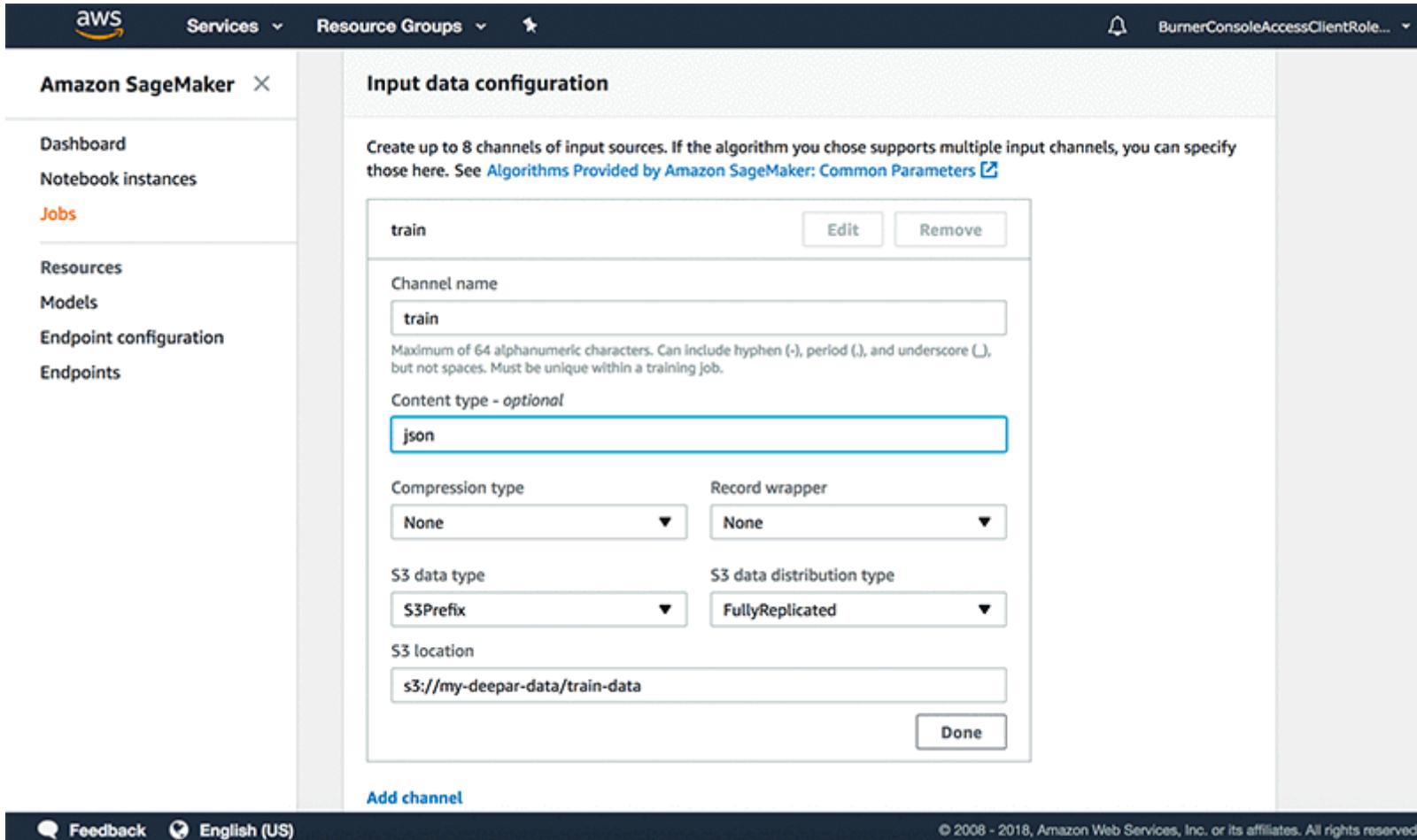


Took 1 seconds. (outdated)

# Create Visualizations with Amazon QuickSight

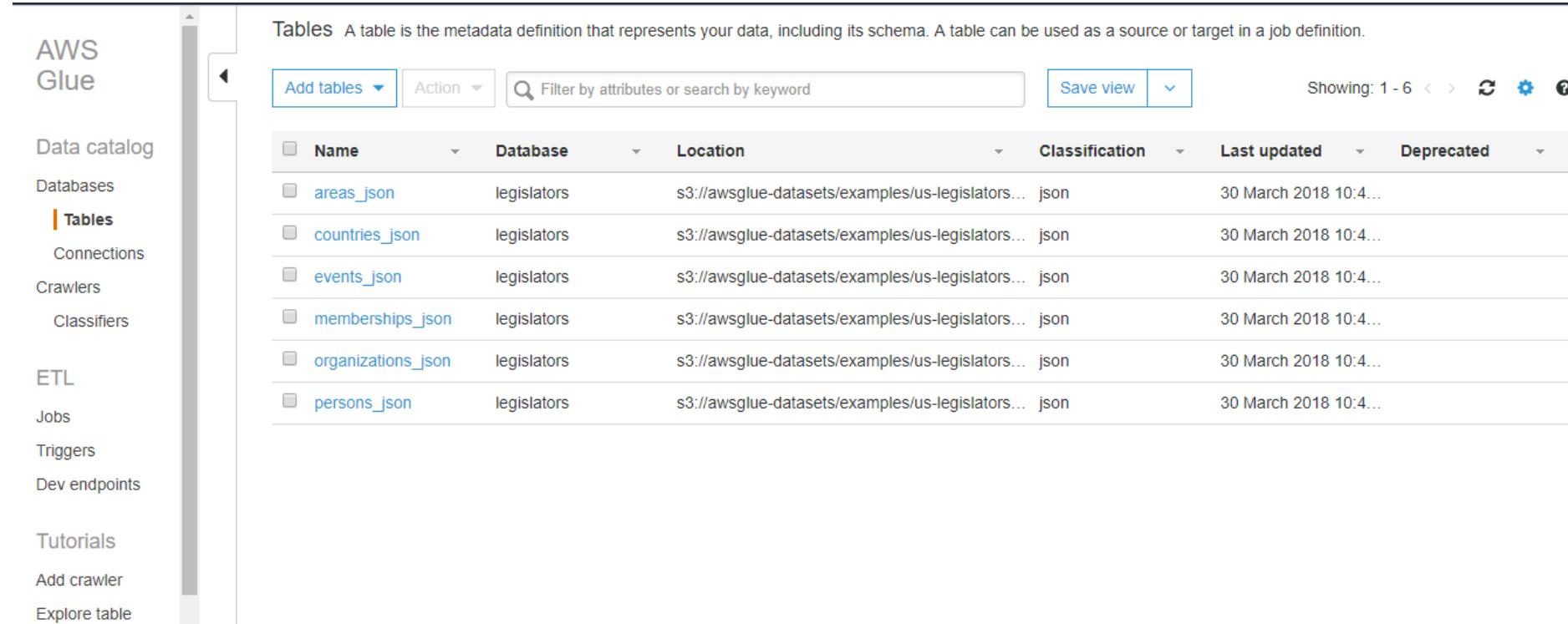


# Train ML Models with Amazon SageMaker



The screenshot shows the 'Input data configuration' page in the Amazon SageMaker console. On the left, a sidebar lists 'Amazon SageMaker' (selected), 'Dashboard', 'Notebook instances', 'Jobs' (selected), 'Resources', 'Models', 'Endpoint configuration', and 'Endpoints'. The main area is titled 'Input data configuration' and contains instructions: 'Create up to 8 channels of input sources. If the algorithm you chose supports multiple input channels, you can specify those here. See [Algorithms Provided by Amazon SageMaker: Common Parameters](#)'. A 'train' channel is selected, with 'Edit' and 'Remove' buttons. The 'Channel name' is set to 'train'. The 'Content type - optional' field is set to 'json'. Under 'Compression type', 'None' is selected. Under 'Record wrapper', 'None' is selected. Under 'S3 data type', 'S3Prefix' is selected. Under 'S3 data distribution type', 'FullyReplicated' is selected. The 'S3 location' is set to 's3://my-deepar-data/train-data'. A 'Done' button is at the bottom right. At the bottom of the page, there are 'Add channel' and 'Feedback' links, and a footer with 'English (US)' and copyright information: '© 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.'

# Create a Central Data Catalog with AWS Glue



The screenshot shows the AWS Glue Data Catalog interface. On the left, a sidebar lists various categories: Data catalog, Databases, Tables (which is selected and highlighted in orange), Connections, Crawlers, Classifiers, ETL, Jobs, Triggers, Dev endpoints, Tutorials, Add crawler, and Explore table. The main area is titled "Tables" with a sub-instruction: "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." Below this are buttons for "Add tables", "Action", "Save view", and navigation controls. A search bar says "Filter by attributes or search by keyword". The table itself has columns: Name, Database, Location, Classification, Last updated, and Deprecated. Six rows are listed, all named with ".json" suffixes and belong to the "legislators" database:

Name	Database	Location	Classification	Last updated	Deprecated
areas_json	legislators	s3://awsglue-datasets/examples/us-legislators... json		30 March 2018 10:4...	
countries_json	legislators	s3://awsglue-datasets/examples/us-legislators... json		30 March 2018 10:4...	
events_json	legislators	s3://awsglue-datasets/examples/us-legislators... json		30 March 2018 10:4...	
memberships_json	legislators	s3://awsglue-datasets/examples/us-legislators... json		30 March 2018 10:4...	
organizations_json	legislators	s3://awsglue-datasets/examples/us-legislators... json		30 March 2018 10:4...	
persons_json	legislators	s3://awsglue-datasets/examples/us-legislators... json		30 March 2018 10:4...	

# AWS databases and analytics

## Broad and deep portfolio, built for builders

Business Intelligence & Machine Learning								AWS Marketplace 250+ solutions	
 Amazon QuickSight	 Amazon SageMaker	 Amazon Comprehend	 Amazon Rekognition	 Amazon Lex	 Amazon Transcribe	 AWS DeepLens			
Databases	Analytics				Blockchain				
 QLDB Ledger Database <span style="background-color: #8B9BBE; color: white; padding: 2px 5px;">NEW</span>	 Neptune Graph	 Amazon Redshift Data warehousing	 Athena Interactive analytics	 Managed Blockchain <span style="background-color: #8B9BBE; color: white; padding: 2px 5px;">NEW</span>	730+	Database solutions			
 ElastiCache Redis, Memcached	 DynamoDB Key value, Document	 Amazon EMR Hadoop + Spark	 Kinesis Analytics Real-time	 Blockchain Templates	600+	Analytics solutions			
 Aurora MySQL, PostgreSQL	 Timestream <span style="background-color: #8B9BBE; color: white; padding: 2px 5px;">NEW</span> Time Series	 Amazon Elasticsearch service Operational Analytics			25+	Blockchain solutions			
 RDS MySQL, PostgreSQL, MariaDB, Oracle, SQL Server	 RDS on VMWare <span style="background-color: #8B9BBE; color: white; padding: 2px 5px;">NEW</span>								
S3/Amazon Glacier	Lake Formation Data Lakes <span style="background-color: #8B9BBE; color: white; padding: 2px 5px;">NEW</span>	Data Lake		AWS Glue ETL & Data Catalog					
				 AWS Glue ETL & Data Catalog					
20+									
Data Movement	Database Migration Service   Snowball   Snowmobile   Kinesis Data Firehose   Kinesis Data Streams   Data Pipeline   Direct Connect							30+	solutions

# Kansas City, Missouri, Spurs Growth with Smart City Analytics on AWS

## Challenge

With IoT devices generating data around the clock—plus more than 4,200 existing datasets—Kansas City, Missouri needed a solution to turn data into tangible outcomes.

## Solution

The city engaged Xaqt, which provides an analytics and intelligence platform on AWS that helps policymakers better innovate how cities function.

## Benefits

- Achieved expected road-maintenance cost savings of up to 50%
- Identified vacant buildings with 85% accuracy
- Ingests one million events per day and growing

“

The Xaqt solution built on AWS enables us to perform preventive roadwork. We expect to **save up to 50%** compared to emergency repair.

**Bob Bennett**, Chief Innovation Officer, City of Kansas City, Missouri

”



Company:	City of Kansas City, Missouri
Industry:	Government & Public Services
Country:	United States
Employees:	4,800
Website:	<a href="http://www.kcmo.gov">www.kcmo.gov</a>

## About Kansas City, Missouri

With a population of 2.1 million, Kansas City, Missouri, is a fast-growing city located in the heart of the United States. In the downtown area, the city has seen more than \$2.1 billion in development over the past 10 years.



# Thank You!

adedan@amazon.com

*In God we trust, every other must bring data....*