

Building a Data Lake on AWS

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

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

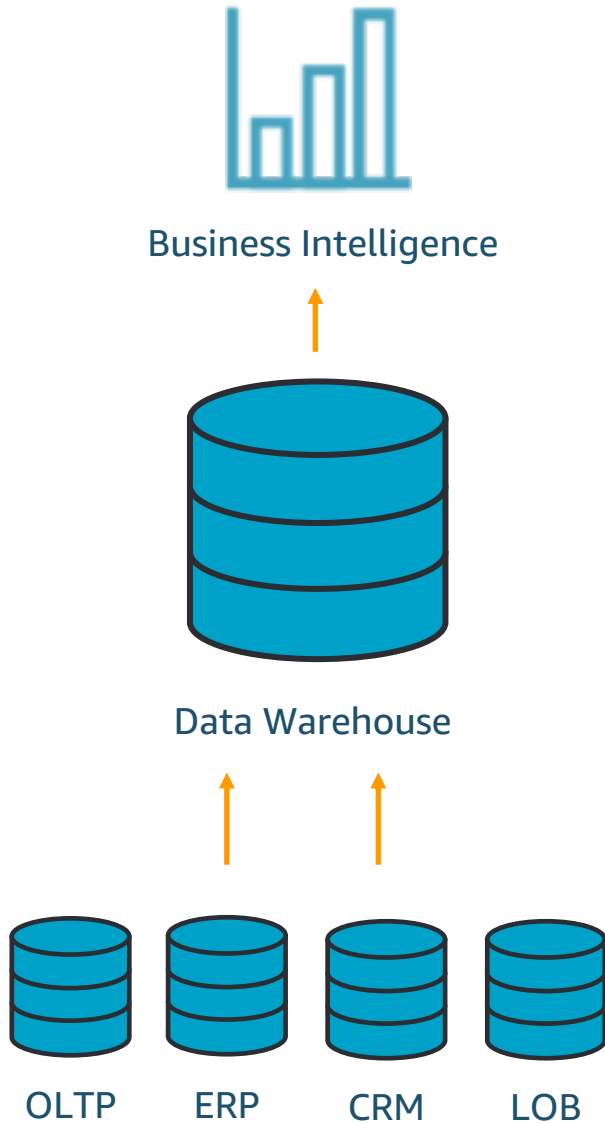


Why a Data Lake?

- 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

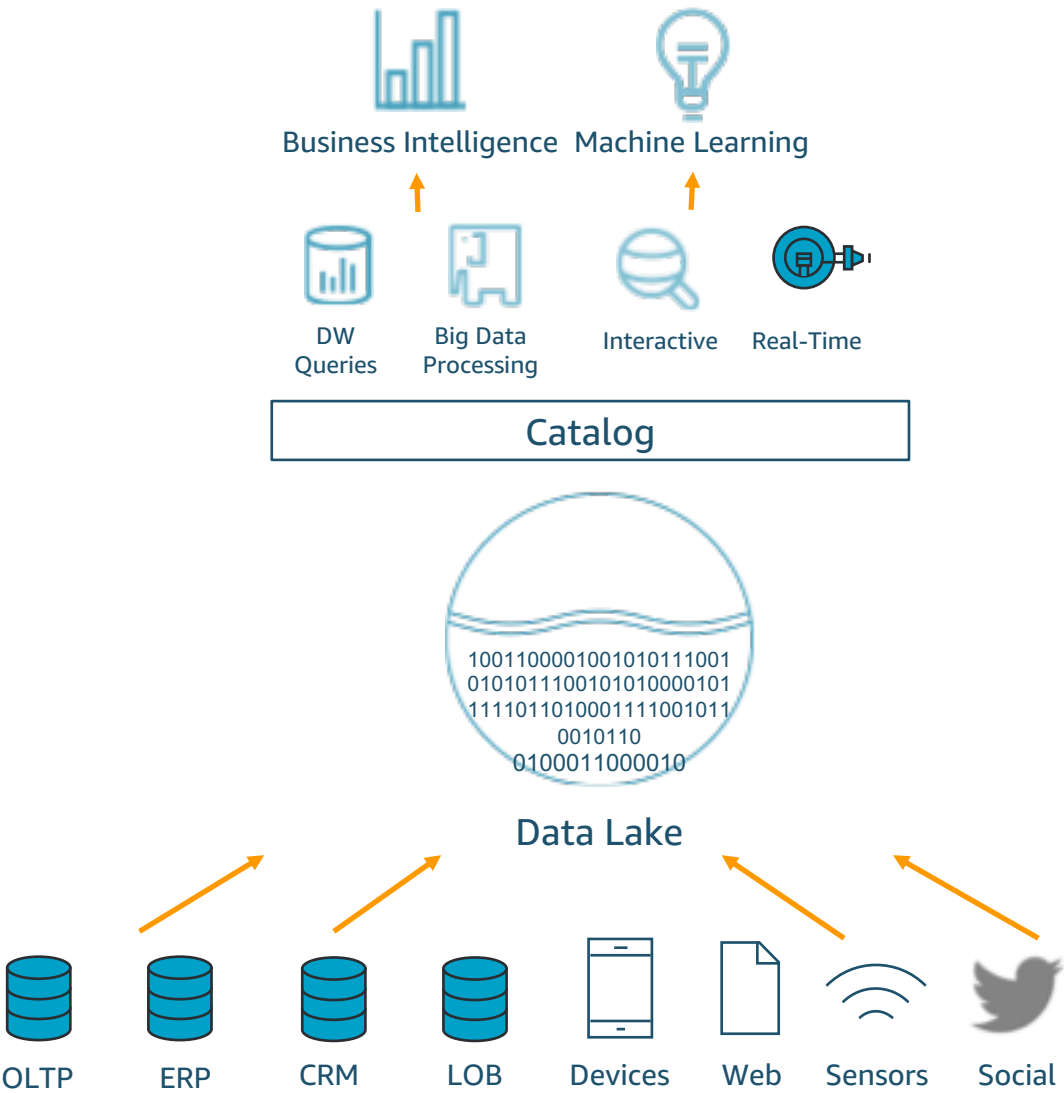
TBs-PBs Scale

Schema Defined Prior to Data Load

Operational and Ad Hoc Reporting

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

Data Lakes Extend the Traditional Approach



Terabyte-Exabyte 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?”

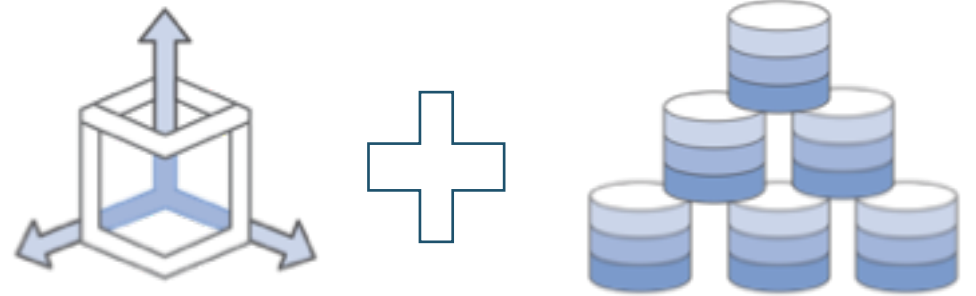


Quickly ingest data without needing to force it into a pre-defined schema.

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?”



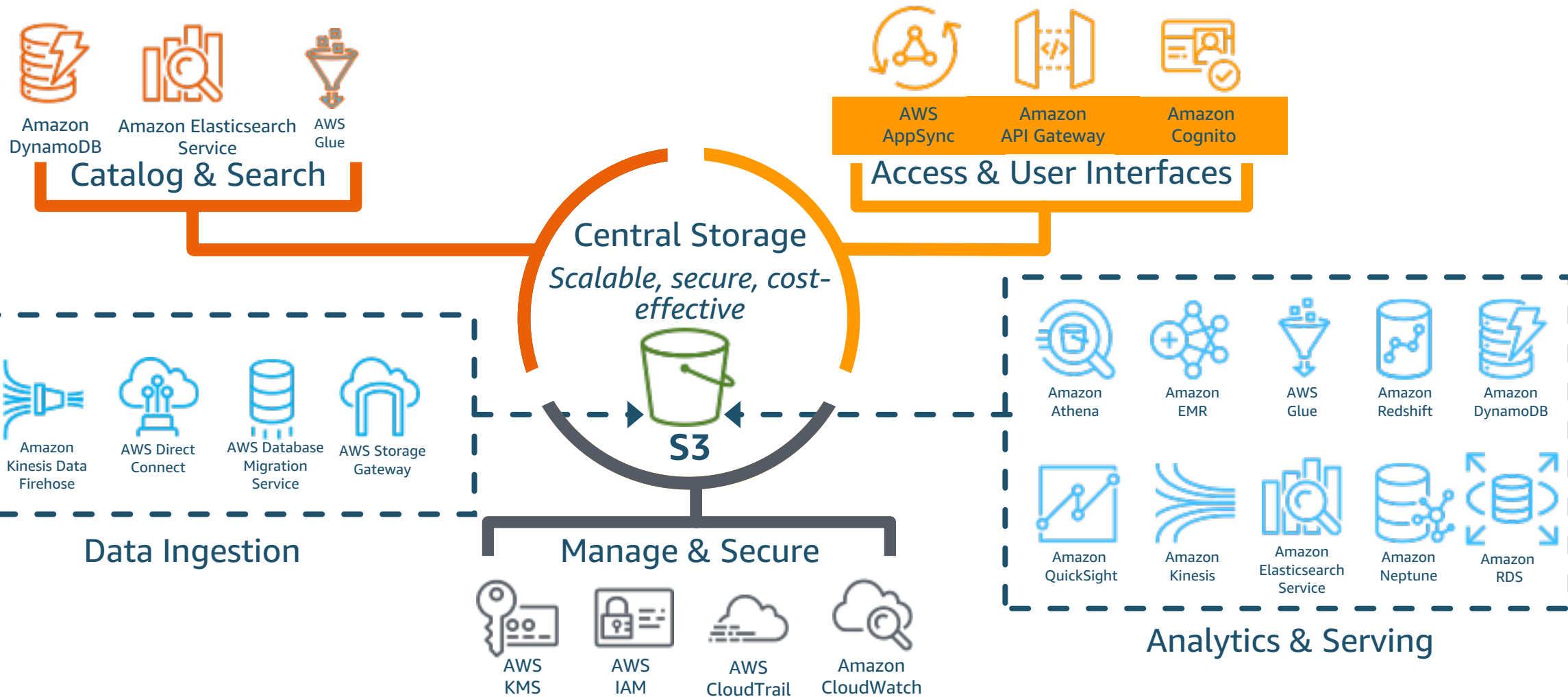
A Data Lake enables ad-hoc analysis by applying schemas on read, not write.

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**.

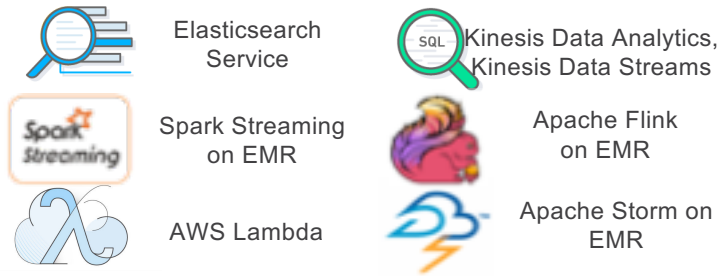
Data Lake on AWS



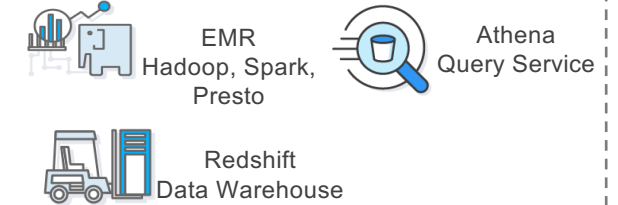


Processing & Analytics

Real-time



Analytics



AI & Predictive



Transactional & RDBMS



BI & Data Visualization



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

What can you do with a Data Lake?

Query Directly with Amazon Athena

The screenshot displays the Amazon Athena Query Editor interface. The top navigation bar includes 'Athena', 'Query Editor' (highlighted), 'Saved Queries', 'History', 'Catalog Manager', 'Settings', 'Tutorial', and 'Help'. On the left, the 'DATABASE' dropdown is set to 'sampledb'. Below it, the 'TABLES' section shows a list of tables with a search filter. The 'Add table...' section lists various columns from the 'elb_logs' table, such as '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 main query editor area is titled 'ELB Select Query' with a subtitle 'Sample query to view peak load ELBs during a particular timeframe'. It contains the following SQL query:

```
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 editor, 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 at the bottom shows a table with two columns: 'elbname' and 'num'. The first row of results is:

	elbname	num
1	lb-demo	4108

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

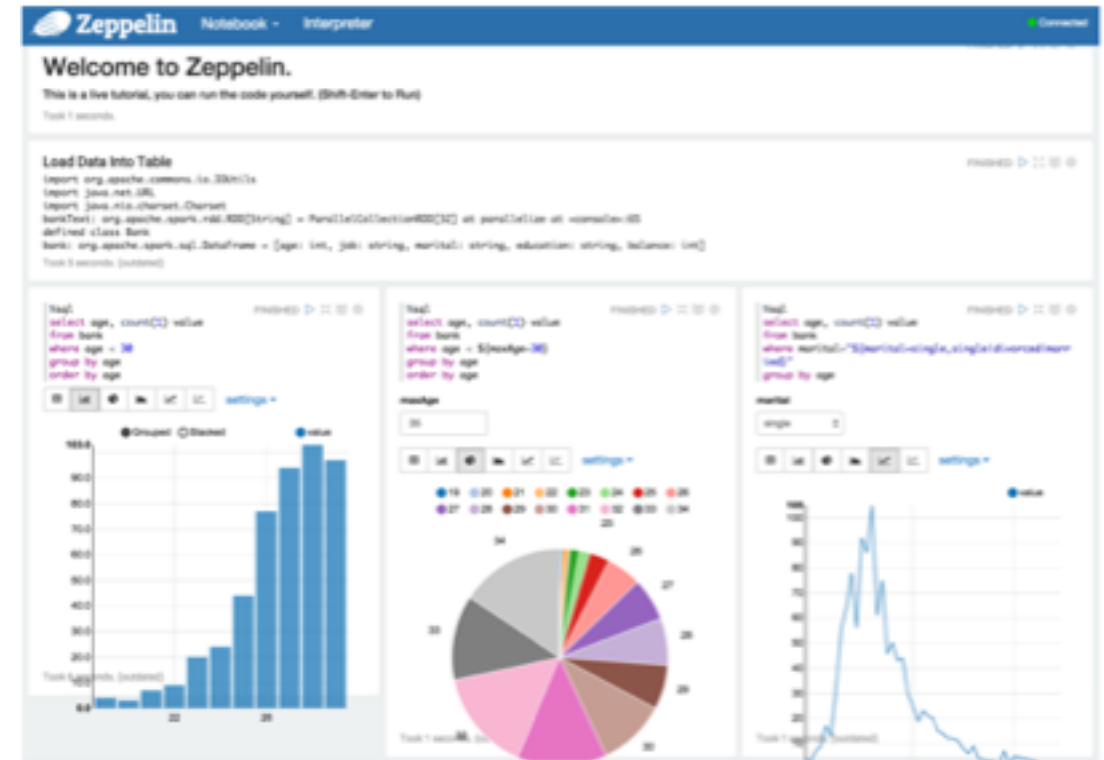
- | | | |
|--|--|---|
| <input checked="" type="checkbox"/> Hadoop 2.6.0 | <input checked="" type="checkbox"/> Hive 1.0.0 | <input type="checkbox"/> Mahout 0.11.0 |
| <input checked="" type="checkbox"/> Zeppelin-Sandbox 0.5.5 | <input type="checkbox"/> Hue 3.7.1 | <input checked="" type="checkbox"/> 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

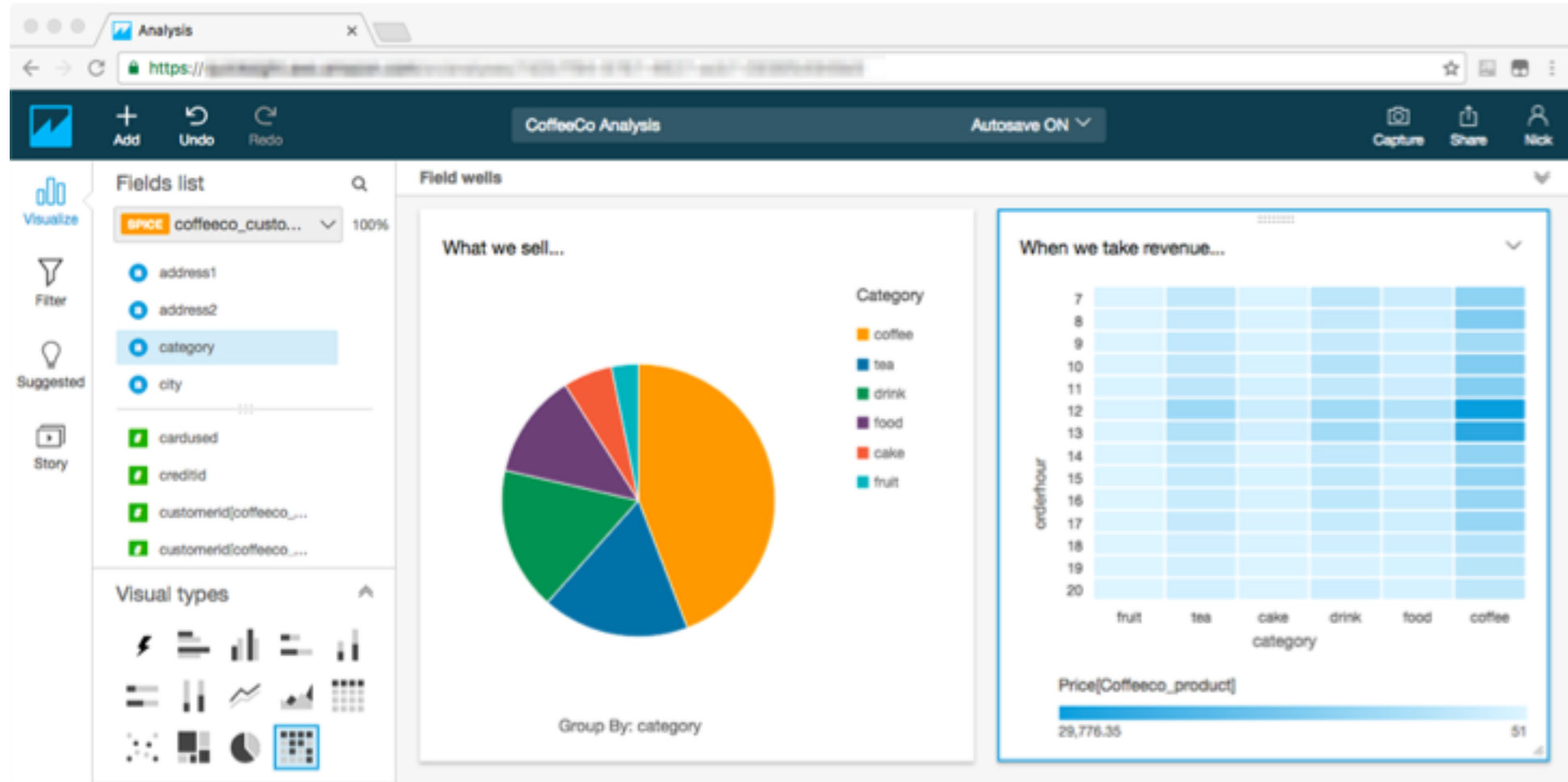
```
val movieLensHomeDir = "s3://emr.examples/movieLens/"

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))
}
```



Create Visualizations with Amazon QuickSight



Train ML Models with Amazon SageMaker

Amazon SageMaker X

Dashboard
Notebook instances
Jobs
Resources
Models
Endpoint configuration
Endpoints

Input data configuration

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](#)

train

Edit Remove

Channel name

train

Maximum of 64 alphanumeric characters. Can include hyphen [-], period [.] and underscore [_], but not spaces. Must be unique within a training job.

Content type - optional

json

Compression type

None

Record wrapper

None

S3 data type

S3Prefix

S3 data distribution type

FullyReplicated

S3 location

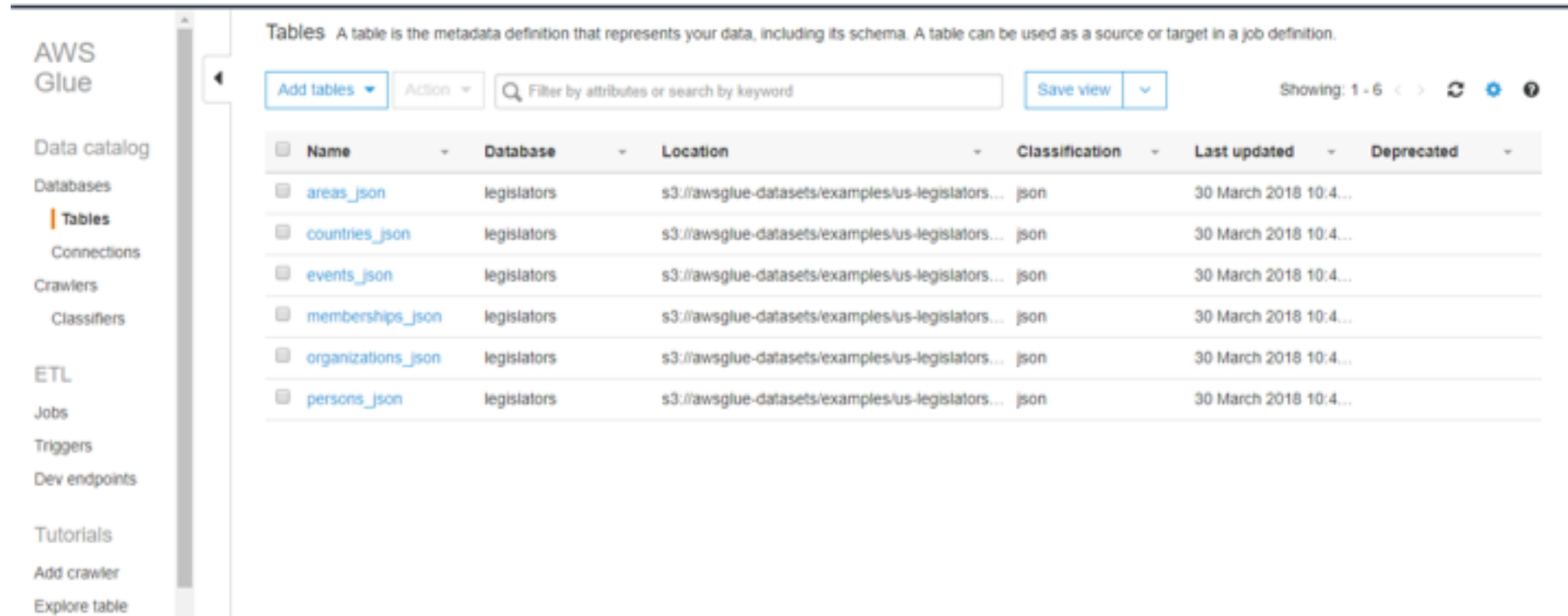
s3://my-deepar-data/train-data

Done

[Add channel](#)

Feedback English (US) © 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 console interface. On the left is a navigation sidebar with links to Data catalog, Databases, Tables (highlighted), Connections, Crawlers, Classifiers, ETL, Jobs, Triggers, Dev endpoints, Tutorials, Add crawler, and Explore table. The main content area is titled 'Tables' and includes a description: '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 is a toolbar with 'Add tables', 'Action', a search filter, and 'Save view'. The table below lists six tables, all in the 'legislators' database, with JSON classification and a last updated date of 30 March 2018.

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

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

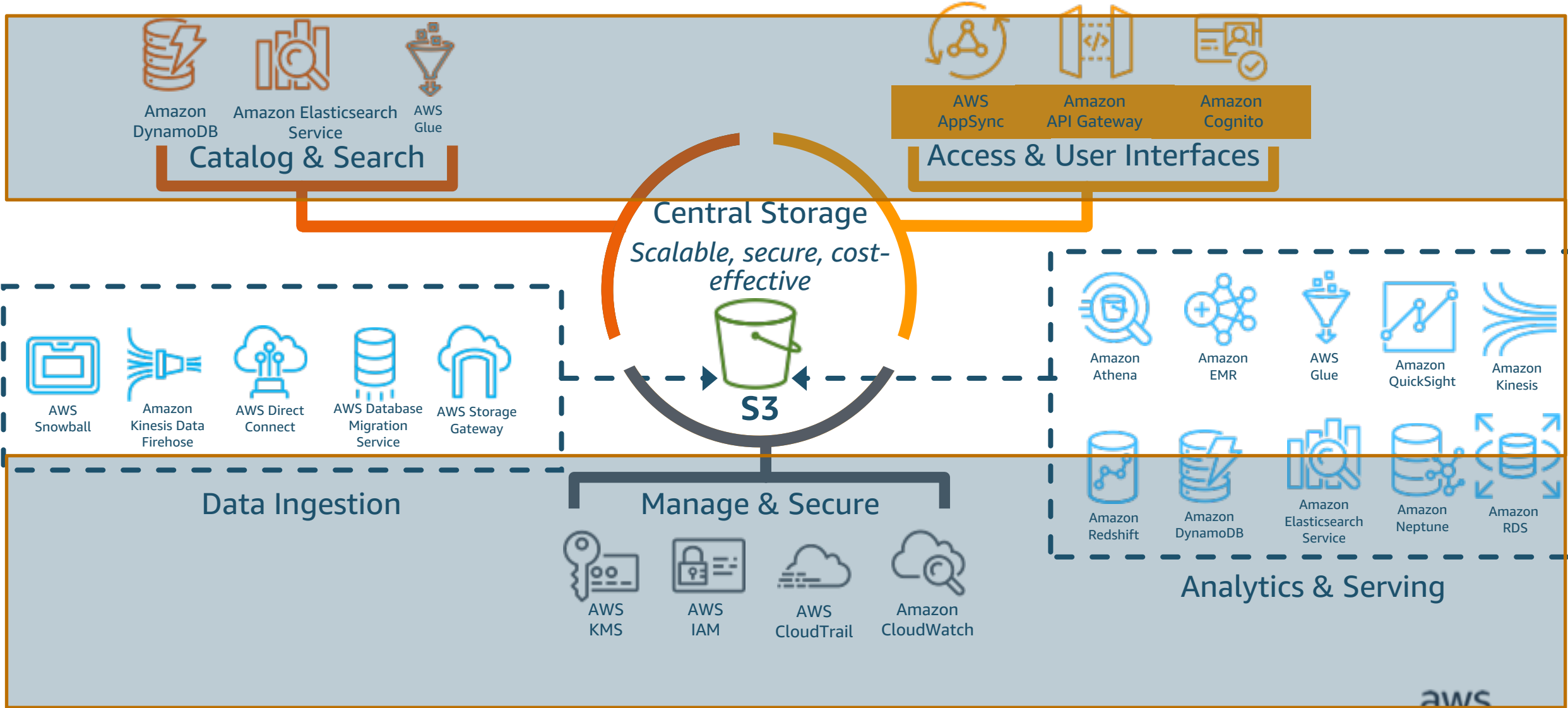
A NoSQL database service that delivers consistent, single-digit millisecond latency at any scale.



Amazon Elasticsearch

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

Data Lake on AWS



Thank You