



AWS PARTNER CERTIFICATION READINESS

Content Review Session

Week 3 – Domain 2

Data Store Management



OPTIONAL AWS Skill Builder Subscription

The Skill Builder subscription provides access to official AWS Certification practice exams, self-paced digital training content including open-ended challenges, self-paced labs, and game-based learning. **Please note, the Skill Builder subscription is not required for this Accelerator program.**



Free digital training

[LINK HERE](#)

Special features include:

- 600+ digital courses
- Learning plans
- 10 Practice Question Sets
- *AWS Cloud Quest (Foundational)*



Individual subscription

[LINK HERE](#)

Everything in free digital training, plus:

- AWS Cloud Quest (Intermediate - Advanced)
- AWS Certification Official Practice Exams
- Enhanced Exam Prep Courses
- Unlimited access to 1000+ hands-on labs
- AWS Jam Journeys (lab-based challenges)
- AWS Digital Classroom (Annual only)

Access **65**
Data Engineer - Associate Practice Exam Questions
with feedback on
your answer choices

Individual subscriptions are priced at **\$29 USD per month** (*Flexibility to cancel anytime*) or **\$449 USD per year**.

Today's Learning Outcomes



During this session, we will cover:

- Choosing a data store
- Understand data cataloging systems
- Manage the lifecycle of data
- Design data models and schema evolution





AWS PARTNER CERTIFICATION READINESS

Domain 2: Data Store Management

Choosing a data store



Choosing a data store

Knowledge of:

- Storage platforms and their characteristics
- Storage services and configurations for specific performance demands
- Data storage formats
- How to align data storage with data migration requirements
- How to determine the appropriate storage solution for specific access patterns
- How to manage locks to prevent access to data

Skills in:

- Implementing the appropriate storage services for specific cost and performance requirements
- Configuring the appropriate storage services for specific access patterns and requirements
- Applying storage services to appropriate use cases
- Integrating migration tools into data processing systems
- Implementing data migration or remote access methods

Managing Data Within AWS



Data Storage



EFS



FSx

File



EBS

Block



S3



S3 Glacier

Object

What is File Data?

A hierarchical storage system that provides shared access to file data.

Files also contain metadata like file name, size, and timestamps

What is Block Data?

Works by dividing data into fixed-sized blocks and stores them as individual units.

Includes most data types

What is Object Data?

Stores and Manages data as Objects

Examples include data like documents, images, or data values

Amazon Simple Storage Service (S3)



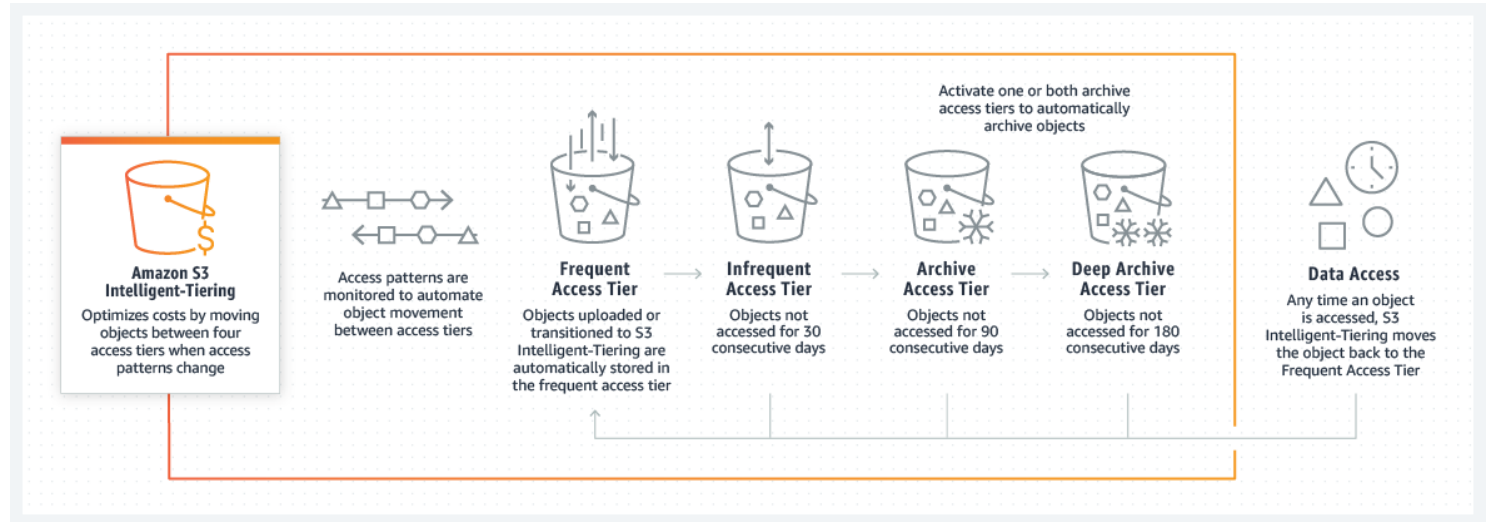
Provides infinitely scalable, highly durable object storage in the AWS Cloud

What does it do?

Stores objects in resources called Buckets, which can be up to 5TB in size, but there are no total limits to the # of objects stored.

Designed to provide 99.999999999% durability and up to 99.99% availability.

Offered at multiple tiers of pricing based on the frequency the objects are needed, and the speed at which they are required to be retrieved.



Amazon Simple Storage Service (S3)



Provides infinitely scalable, highly durable object storage in the AWS Cloud



S3 Intelligent-Tiering



S3 Standard



S3 Standard-IA



S3 Glacier



S3 Glacier Deep Archive



S3 One Zone-IA



S3 Outposts

AWS Region \geq 3 Availability Zones

- Data with changing access patterns
- Opt in for automatic archiving
- Active, frequently accessed data
- Milliseconds access
- Infrequently accessed data
- Milliseconds access
- Retrieval fee per GB
- Minimum storage duration
- Minimum object size
- Archive data
- In minutes and hours
- Retrieval fee per GB
- Minimum storage duration
- Minimum object size
- Long-term archive data
- Select hours
- Retrieval fee per GB
- Minimum storage duration
- Minimum object size

AWS Single AZ

- Re-creatable, less accessed data
- Milliseconds access
- Retrieval fee per GB
- Minimum storage duration
- Minimum object size

AWS Outposts

- On-premises data
- Milliseconds access
- Encrypted with SSE-S3

Amazon Elastic File System



Simple, serverless, set-and-forget, elastic file system

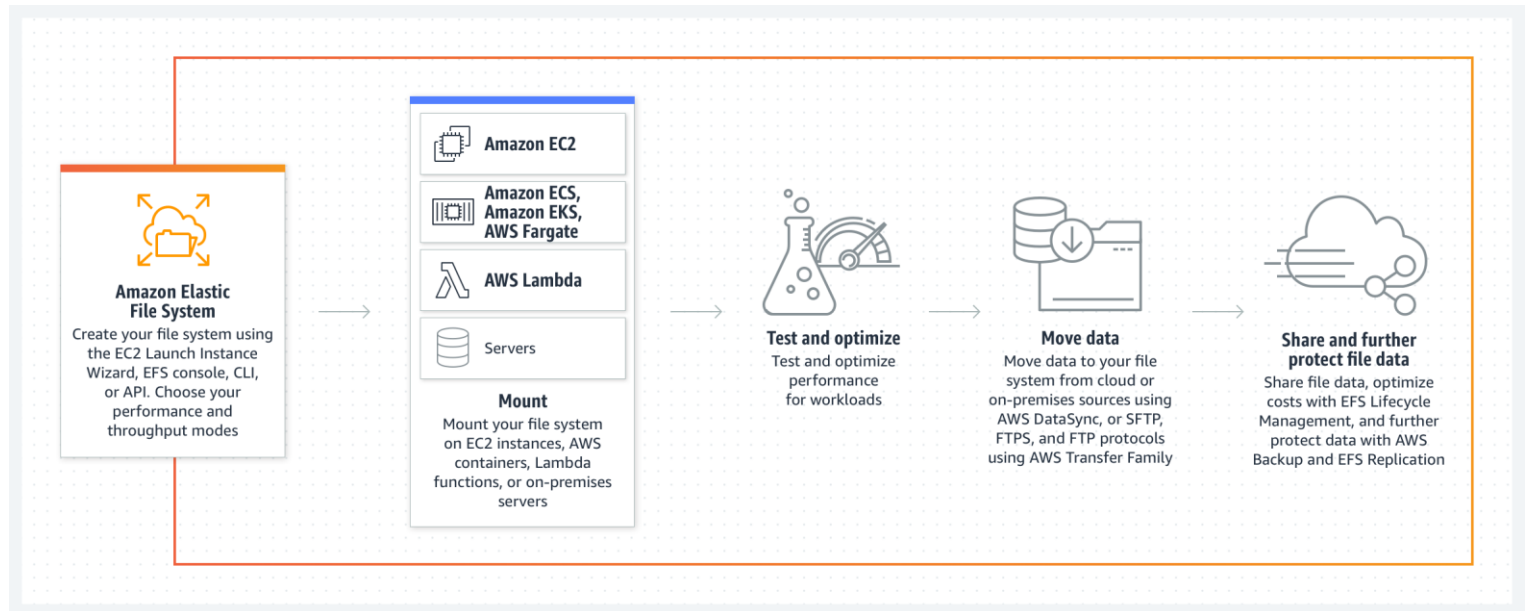
Amazon Elastic File System (EFS) automatically grows and shrinks as you add and remove files with no need for management or provisioning (Automatically spreads data across multiple AZ's for availability)

Simplify

Share code and other files in a secure, organized way to increase DevOps agility and respond faster to customer feedback.

Enhance

Simplify persistent storage for modern content management system (CMS) workloads. Get your products and services to market faster, more reliably, and securely at a lower cost.



Amazon Elastic Block Store (EBS)



Easy to use, high performance block storage at any scale

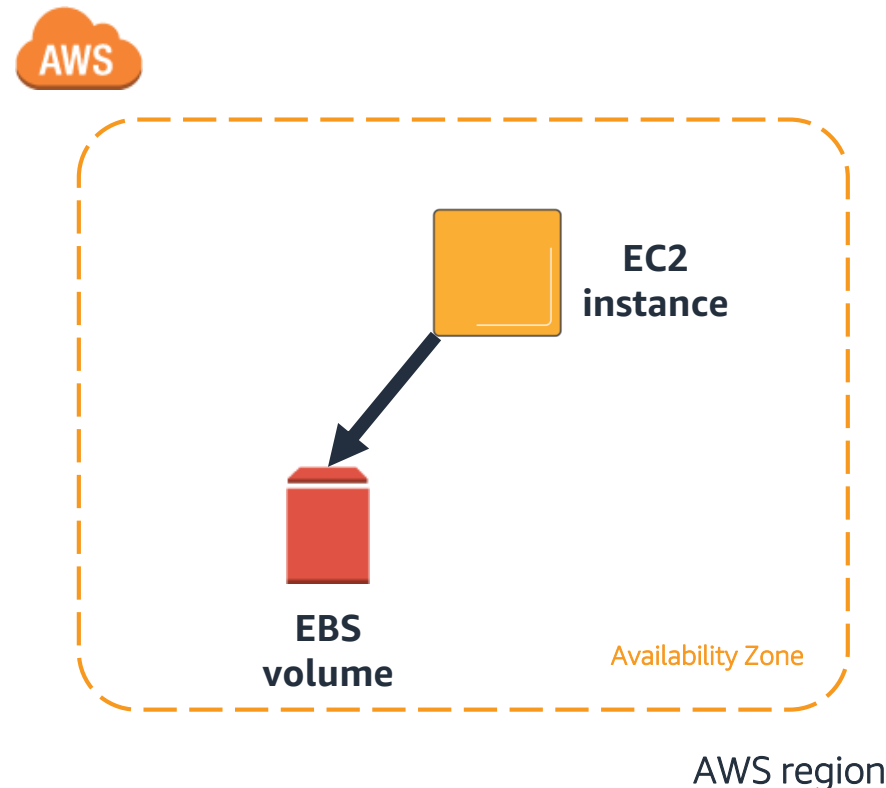
Provides block level storage volumes for use with EC2 instances. Data automatically replicated within the same AZ, but not to another AZ

Improve Performance

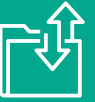
Attach high availability block storage for mission-critical applications and increase volume size without disrupting your users.

Access Quickly

EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage.



AWS Transfer Family



Easily manage and share data with simple, secure, and scalable file transfers

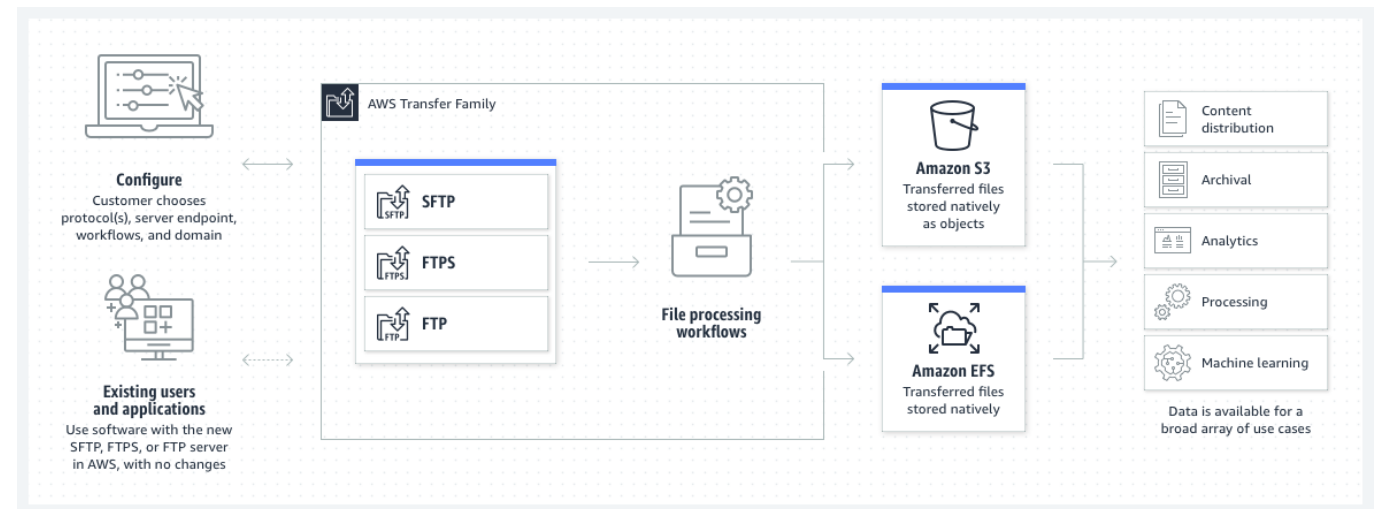
AWS Transfer Family securely scales your recurring business-to-business file transfers to AWS Storage services using SFTP, FTPS, FTP, and AS2 protocols.

Ease of Use

Seamlessly migrate, automate, and monitor your file transfer workflows by maintaining existing client-side configurations for auth, access, and firewalls.

Data lake integration

Use AWS Transfer Family to create secure connections between disparate sources of B2B data and your data lakes. Files transferred in are stored in Amazon S3 or Amazon EFS.



Amazon Redshift



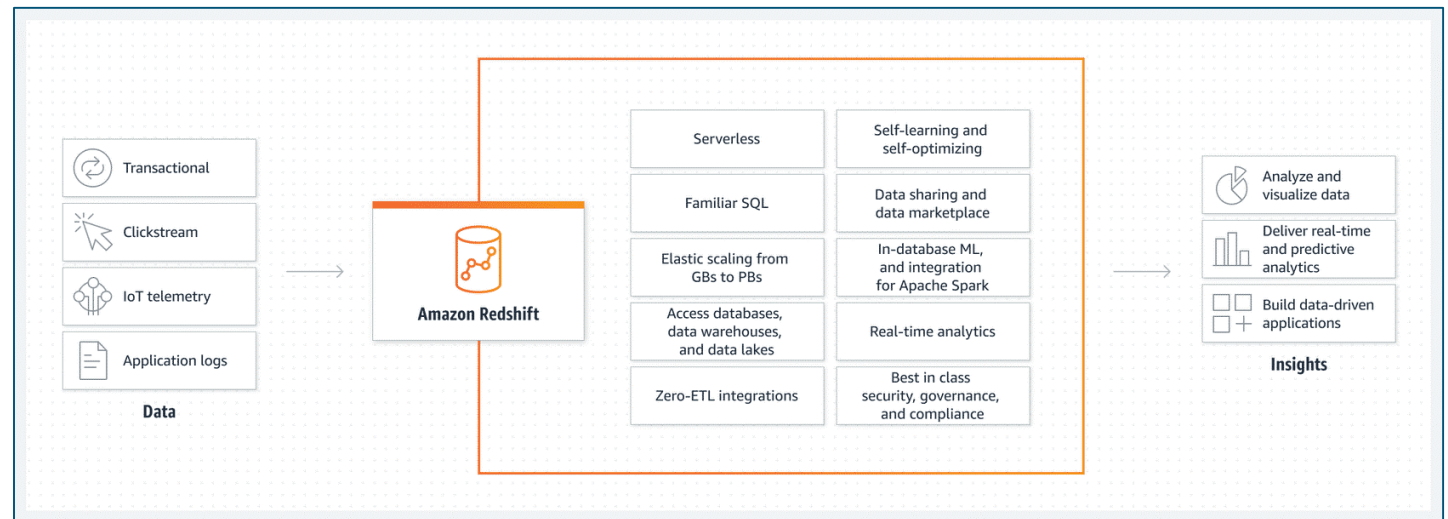
Power data driven decisions with the best price-performance cloud data warehouse

Unify Data

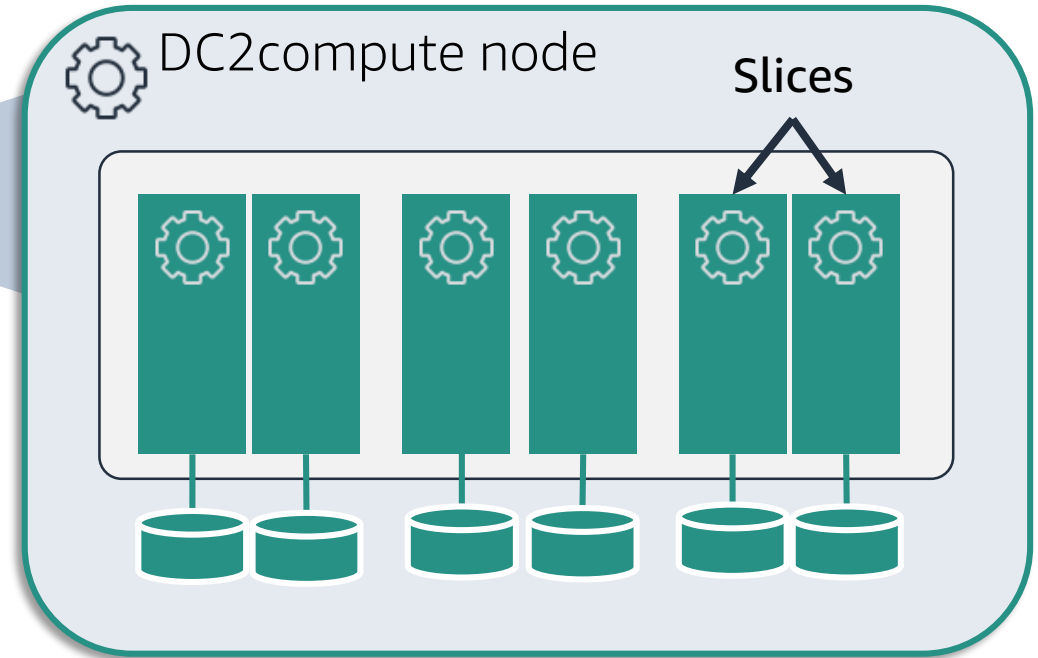
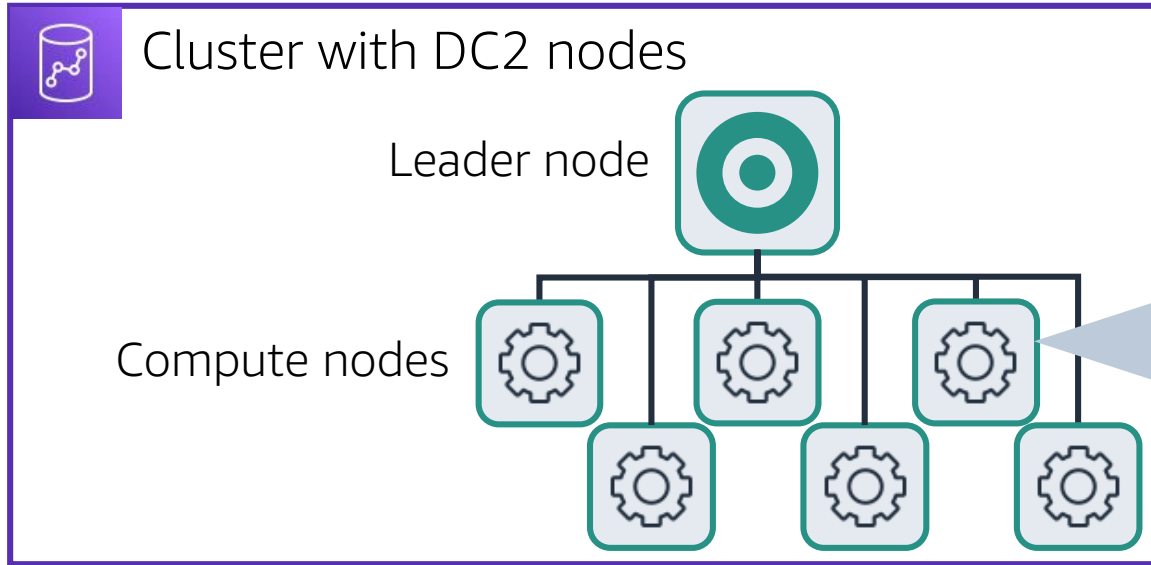
Easily access or ingest data across your data lakes, databases, data warehouses, streaming data - with no code/low code zero-ETL approach for integrated analytics.

Maximize Value

Run SQL queries and open source analytics, power dashboards and visualizations, activate near real-time analytics and AI/ML applications with analytics engines and languages of your choice.

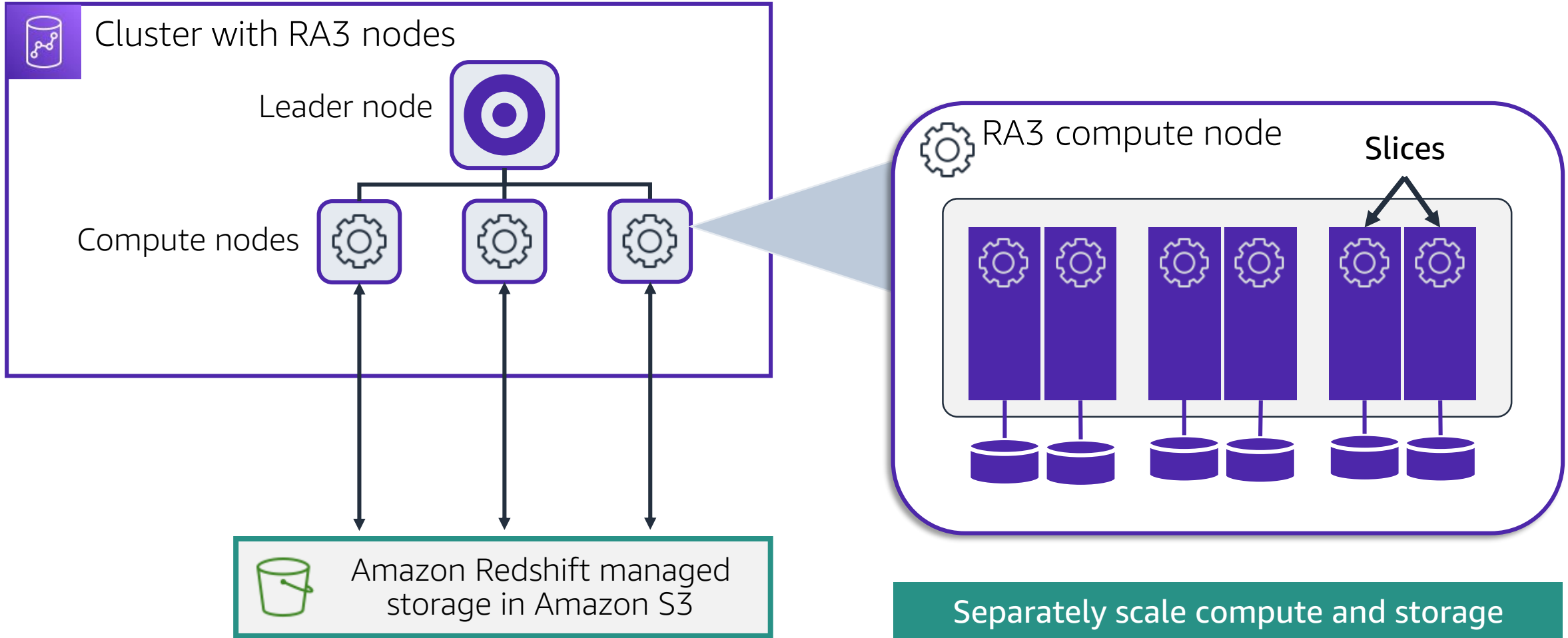


Amazon Redshift cluster architecture – DC2



Compute and storage scale together

Amazon Redshift cluster architecture – RA3



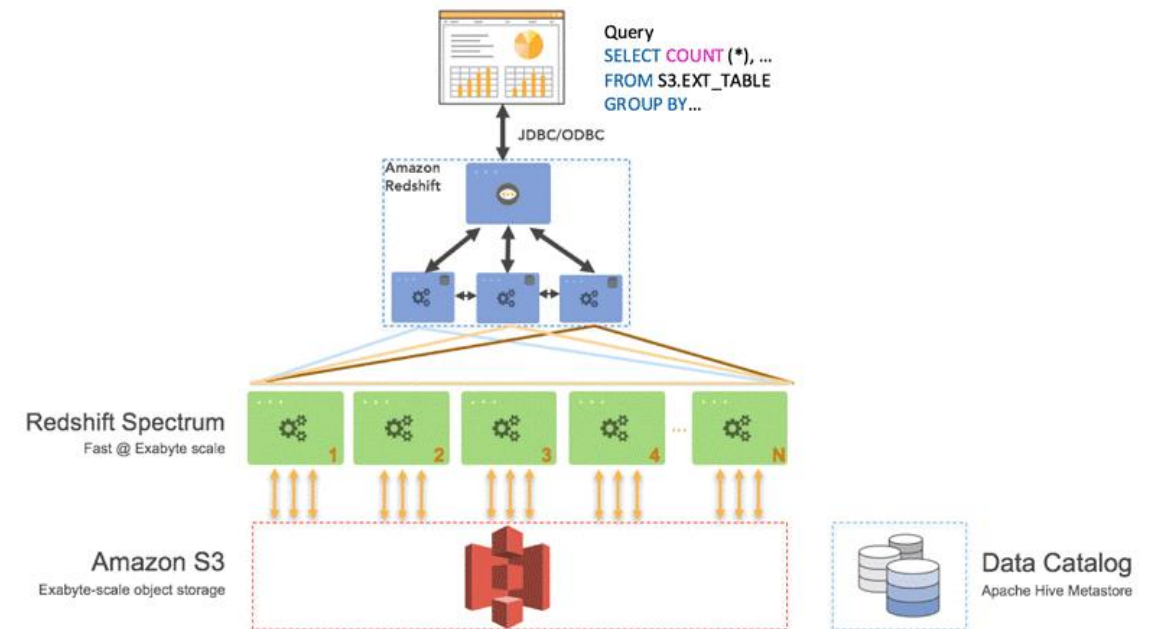
Amazon Redshift Spectrum



Efficiently query and retrieve structured and semistructured data from files in Amazon S3 without having to load the data into Amazon Redshift tables

Overview

Amazon Redshift Spectrum resides on dedicated Amazon Redshift servers independent of your own cluster. Pushes compute-intensive tasks down to the Spectrum layer, thus using less of your cluster's processing capacity. Redshift Spectrum scales intelligently, potentially providing massively parallel processing by using thousands of instances.



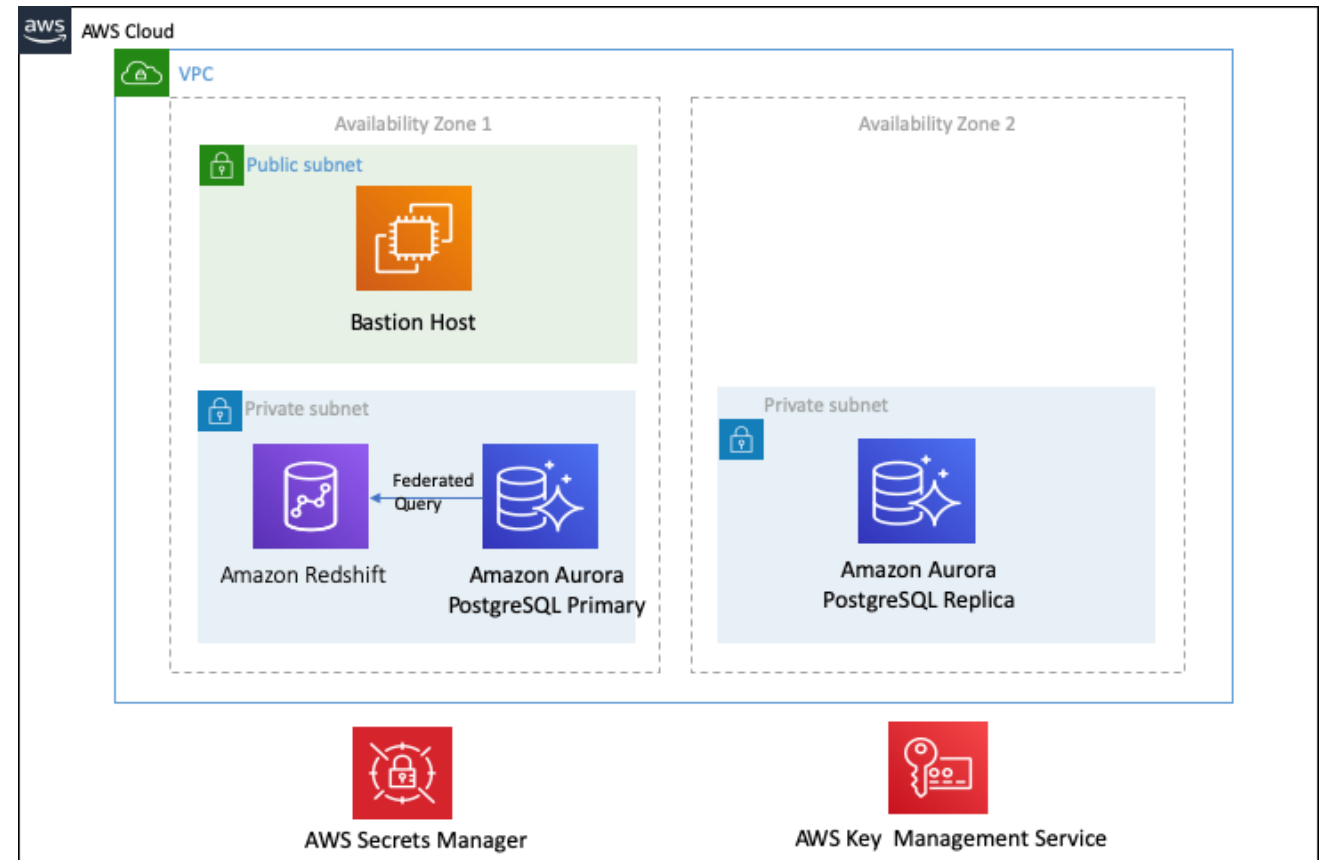
Amazon Redshift – Federated Queries



Query and analyze data across operational databases, data warehouses, and data lakes.

How To

1. Set up connectivity from Amazon Redshift cluster to database instance
2. Set up secrets in AWS Secrets Manager for DB access, reference in AWS IAM access policies and roles
3. Apply IAM role to the Amazon Redshift cluster
4. Connect to your databases with an external schema
5. Run your SQL queries referencing the external schema that references your data bases



Securing federated queries




Amazon RDS AWS Secrets Manager



Secret:
database
credentials

1



Create a secret for Amazon Relational Database Service (Amazon RDS) credentials.



Amazon Redshift

2

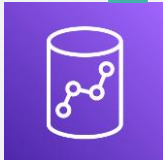

Apply the policy to Amazon Redshift.



Amazon Redshift

3

Create an external schema and tables for Amazon RDS.



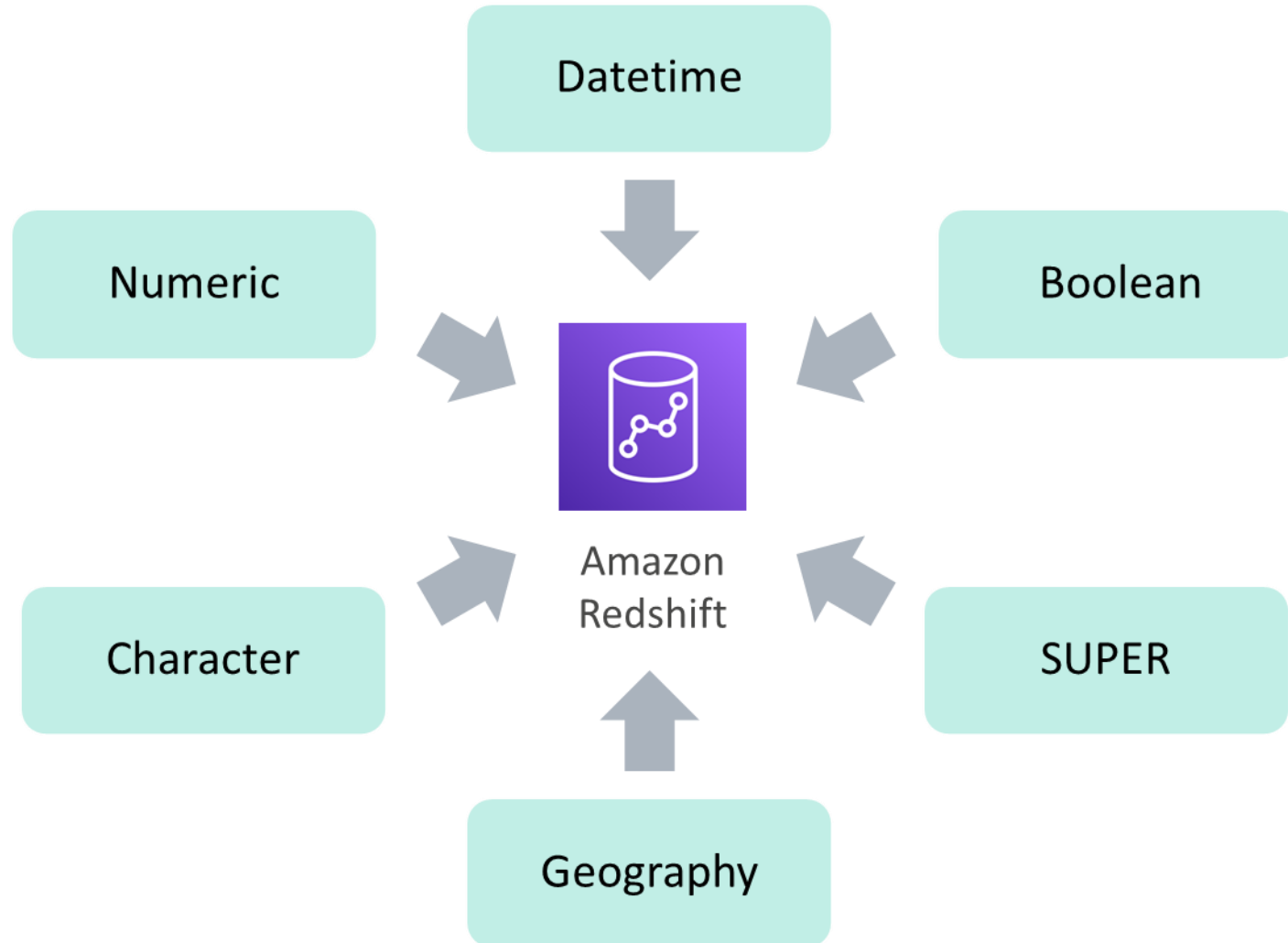
Amazon RDS

Amazon Redshift

4

Query Amazon RDS from Amazon Redshift.

Amazon Redshift supported data types



SUPER data type

Table: customers

| id INTEGER | name SUPER | phones SUPER |
|---------------|---|---|
| 1 | {"given":"Jane", "family":"Doe"} | [{"type":"work", "num":"9255550100"}, {"type":"cell", "num": 6505550101}] |
| 2 | {"given":"Richard", "family":"Roe"}, | [{"type":"work", "num": 5105550102}] |

```
SELECT name.given AS firstname, ph.num  
FROM customers c, c.phones ph  
WHERE ph.type = 'cell'
```

```
firstname | num  
-----+-----  
"Jane"   | 6505550101
```

Columnar Data Storage

Convert CSV based data to optimize analytic query performance

Row Based

Each row contains field values for a single record. Data blocks store values sequentially for each consecutive column making up the entire row. If block size is smaller than the size of a record, storage for an entire record may take more than one block. If block size is larger than the size of a record, storage for an entire record may take less than one block, resulting in an inefficient use of disk space. Examples - **CSV** or **Avro**

| SSN | Name | Age | Addr | City | St |
|-----------|-------|-----|---------------|---------|----|
| 101259797 | SMITH | 88 | 899 FIRST ST | JUNO | AL |
| 892375862 | CHIN | 37 | 16137 MAIN ST | POMONA | CA |
| 318370701 | HANDU | 12 | 42 JUNE ST | CHICAGO | IL |

101259797|SMITH|88|899 FIRST ST|JUNO|AL|892375862|CHIN|37|16137 MAIN ST|POMONA|CA|318370701|HANDU|12|42 JUNE ST|CHICAGO|IL

Block 1

Block 2

Block 3

Column Based

Each data block stores values of a single column for multiple rows. As records enter the system, Amazon Redshift transparently converts the data to columnar storage for each of the columns. Significantly decreases I/O operations and increases storage efficiency.

| SSN | Name | Age | Addr | City | St |
|-----------|-------|-----|---------------|---------|----|
| 101259797 | SMITH | 88 | 899 FIRST ST | JUNO | AL |
| 892375862 | CHIN | 37 | 16137 MAIN ST | POMONA | CA |
| 318370701 | HANDU | 12 | 42 JUNE ST | CHICAGO | IL |

101259797 |892375862| 318370701 |468248180|378568310|231346875|317346551|770336528|277332171|455124598|735885647|387586301

Block 1



Parquet vs ORC

Columnar storage formats that are optimized for fast retrieval of data and used in AWS analytical applications

Characteristics

Compression by column - with compression algorithm selected for the column data type to save storage space in Amazon S3 and reduce disk space and I/O during query processing.

Predicate pushdown in Parquet and ORC enables Athena queries to fetch only the blocks it needs, improving query performance. When an Athena query obtains specific column values from your data, it uses statistics from data block predicates, such as max/min values, to determine whether to read or skip the block.

Splitting of data in Parquet and ORC allows Athena to split the reading of data to multiple readers and increase parallelism during its query processing.

Parquet

Query Performance - Parquet supports a wider range of query types, Parquet might be a better choice if you plan to perform complex queries.

ORC

Complex data types - ORC might be a better choice as it supports a wider range of complex data types.

File size - ORC usually results in smaller files, which can reduce storage costs.



AWS PARTNER CERTIFICATION READINESS

Domain 2: Data Store Management

Understand data cataloging systems



Understand data cataloging systems

Knowledge of:

- How to create a data catalog
- Data classification based on requirements
- Components of metadata and data catalogs

Skills in:

- Using data catalogs to consume data from the data's source
- Building and referencing a data catalog
- Discovering schemas and using AWS Glue crawlers to populate data catalogs
- Synchronizing partitions with a data catalog
- Creating new source or target connections for cataloging

AWS Glue



Discover, prepare, and integrate all your data at any scale

Serverless data integration service that makes it easier to discover, prepare, move, and integrate data from multiple sources for analytics, machine learning (ML), and application development.

Why Use?

Analytics or ML projects begin with preparation of data. AWS Glue provides a serverless data integration service to make data preparation simpler, faster, and cheaper.

- Discover and connect to over 80 diverse data sources
- Manage data in a centralized data catalog
- Visually create, run, and monitor ETL pipelines to load data into your data lakes



AWS Glue Data Catalog



Data Catalog can quickly discover and search multiple AWS datasets without moving the data.

What it is

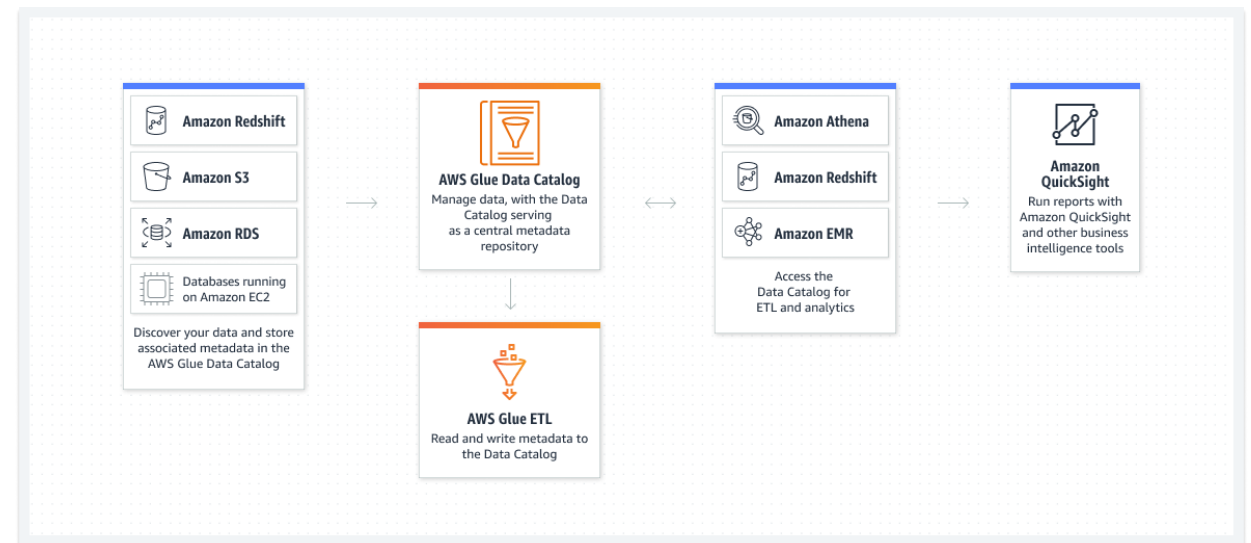
Central repository to store structural and operational metadata for all your data assets. For a given dataset you can store:

- Table definition and physical location
- Business-relevant attributes
- Over time changes to the data

Compatibility & Integrations

Apache Hive Metastore compatible, and a drop-in replacement for the Apache Hive Metastore for Big Data applications running on Amazon EMR.

Built-in integration with Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum.



AWS Glue Schema Registry



Centrally discover, control, and evolve data stream schemas

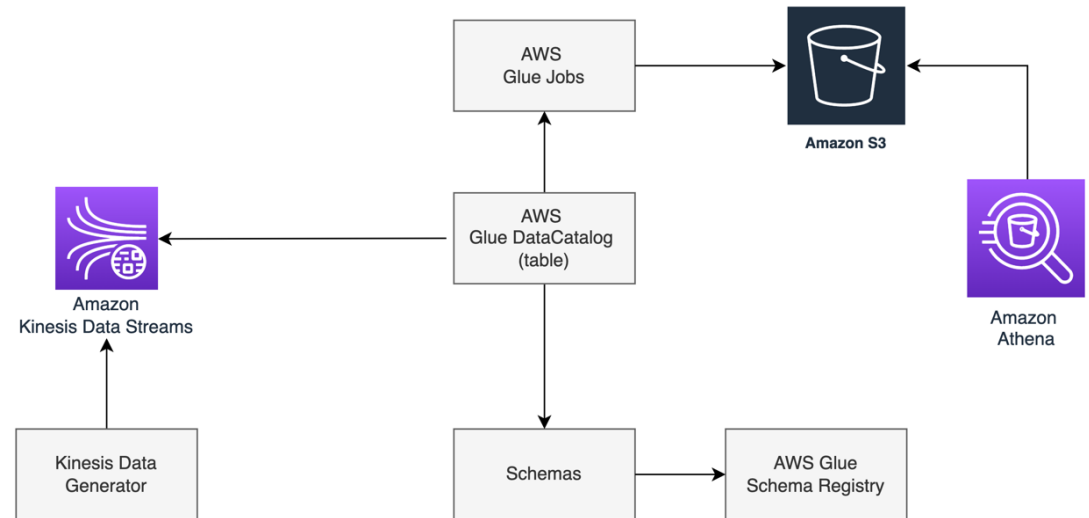
Manage and enforce schemas on your data streaming applications using convenient integrations.

Registry

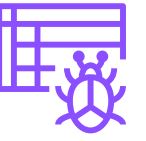
Logical container for schemas. Validate and control the evolution of streaming data using registered schemas. Improve data quality and safeguard against unexpected changes using compatibility checks that govern schema evolution.

Why Use

1. Validate Schemas
2. Safeguard schema evolution
3. Improve data quality
4. Save costs
5. Improve processing efficiency



AWS Glue crawlers



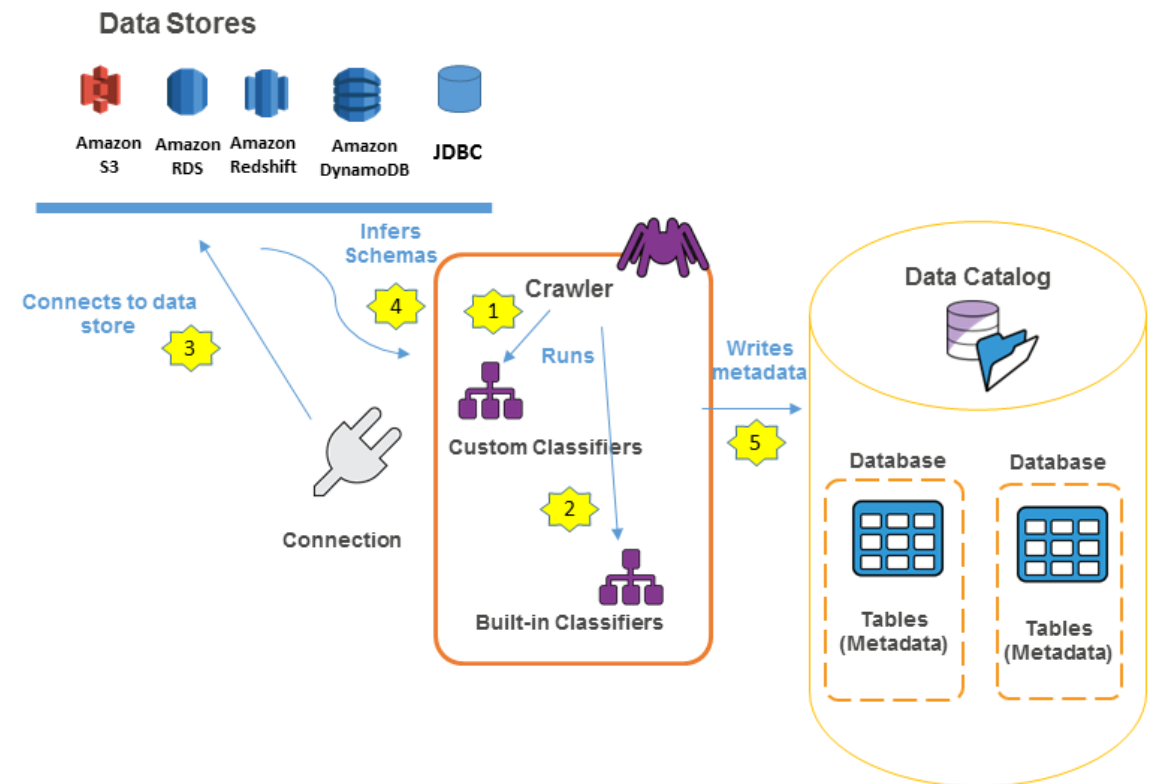
Connects to a data store, progresses through a prioritized list of classifiers to extract the schema of your data, then populates the Data Catalog.

AWS Glue crawlers

Crawlers can run periodically to detect the availability of new data and changes to existing data, including table definition changes. You can customize AWS Glue crawlers to classify your own file types.

General Flow

1. Runs any custom classifiers to infer format and schema of data; built-in classifiers try to recognize data if no matches
2. Crawler connects to data store
3. Inferred schema is created for your data
4. Crawler writes metadata to the Data Catalog



AWS Glue – Table Partitions



AWS Glue table definition of an S3 folder can describe partitioned tables. Allows you to fetch a subset of partitions instead of all partitions.

Constraints

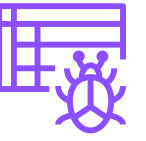
- Schemas of the files are similar
- Data format of the files is the same
- Compression format of the files if the same

Example

Amazon S3 bucket named **my-app-bucket**, where you store both **iOS** and **Android** app sales data. The data is partitioned by **year, month, and day**. The data files for iOS and Android sales have the same schema, data format, and compression format. In the AWS Glue Data Catalog, the AWS Glue crawler creates one table definition with **partitioning keys** for year, month, and day.

```
my-app-bucket/Sales/year=2010/month=feb/day=1/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=1/Android.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/Android.csv
...
my-app-bucket/Sales/year=2017/month=feb/day=4/iOS.csv
my-app-bucket/Sales/year=2017/month=feb/day=4/Android.csv
```

AWS Glue – crawler scheduling



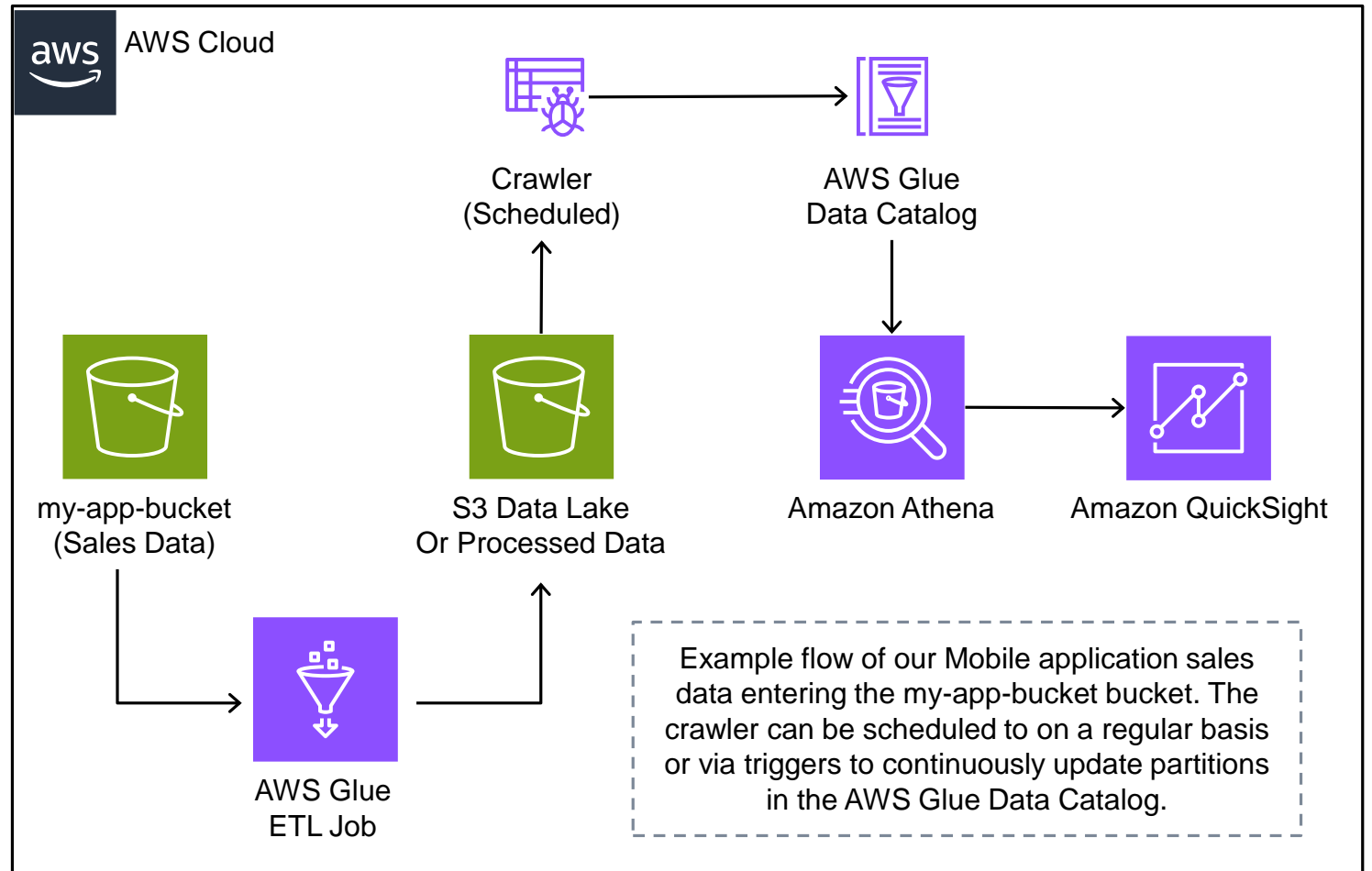
Using AWS Glue crawlers to keep your partitions in data catalog synchronized

AWS Glue Triggers

Scheduled - A time-based trigger based on cron.

Conditional - A trigger that fires when a previous job or crawler or multiple jobs or crawlers satisfy a list of conditions.

On-demand - A trigger that fires when you activate it. On-demand triggers never enter the **ACTIVATED** or **DEACTIVATED** state. They always remain in the **CREATED** state.





AWS PARTNER CERTIFICATION READINESS

Domain 2: Data Store Management

Manage the lifecycle of data



Manage the lifecycle of data

Knowledge of:

- Appropriate storage solutions to address hot and cold data requirements
- How to optimize the cost of storage based on the data lifecycle
- How to delete data to meet business and legal requirements
- Data retention policies and archiving strategies
- How to protect data with appropriate resiliency and availability

Skills in:

- Performing load and unload operations to move data between Amazon S3 and Amazon Redshift
- Managing S3 Lifecycle policies to change the storage tier of S3 data
- Expiring data when it reaches a specific age by using S3 Lifecycle policies
- Managing S3 versioning and DynamoDB TTL

Redshift – Loading data from Amazon S3

Methods of loading data into Amazon Redshift from Amazon S3 buckets

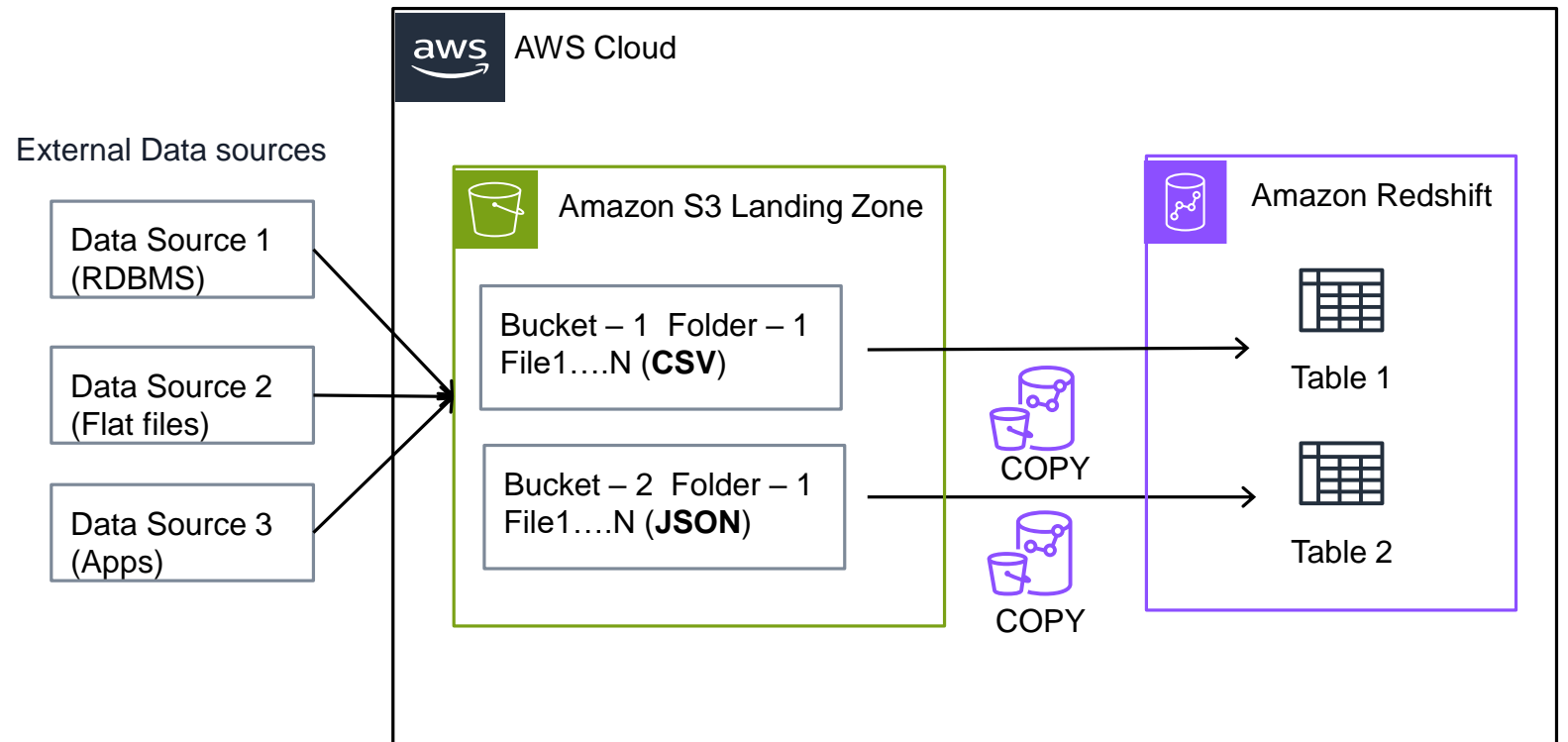
COPY command

Loads data in parallel from:

- Amazon S3
- Amazon EMR
- Amazon DynamoDB
- Other data sources on remote hosts

Loads and stores large amounts of data more efficiently than INSERT statements.

Automated COPY JOB command currently in preview release.



Amazon S3 Lifecycle



Define rules to transition objects from one storage class to another to save on storage costs.

Examples

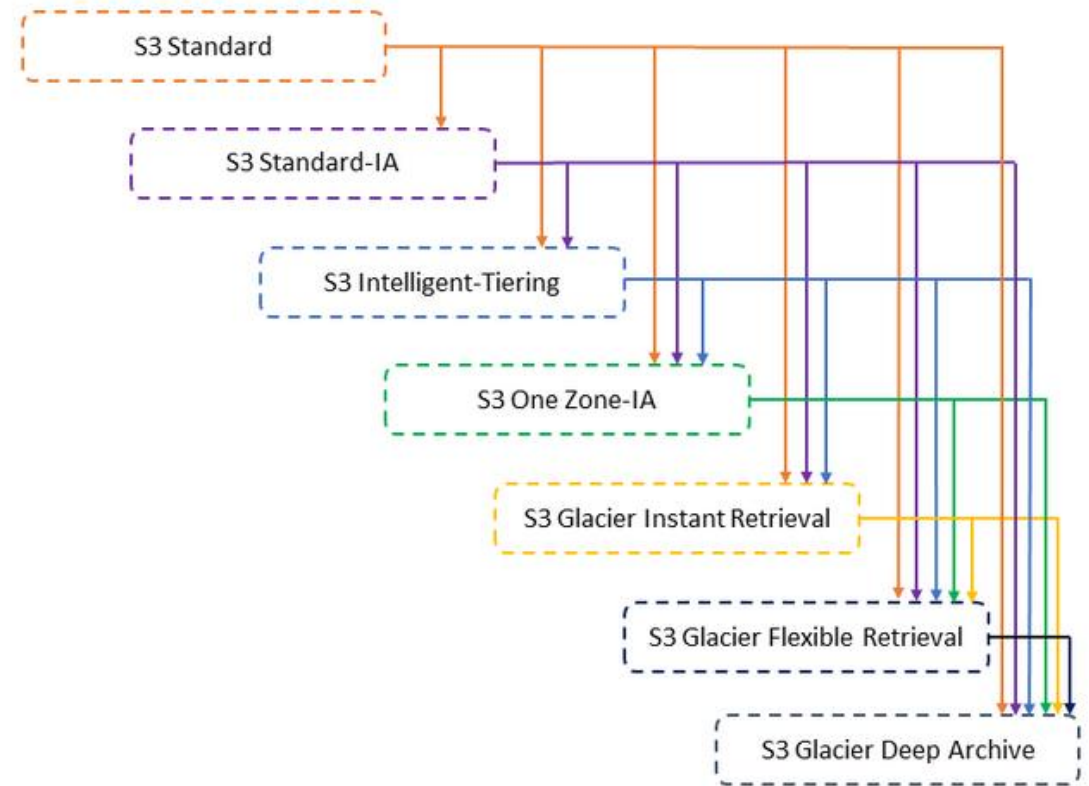
Infrequently Access – Objects that will be infrequently accessed over time may be transitioned to the S3 Standard-IA storage class.

Archival – Objects which doesn't need real-time access may be transitioned to S3 Glacier Flexible Retrieval.

Methods of Management

Rule based – When access patterns are known, configure rules to automatically transition objects to different storage classes.

S3 Intelligent-Tiering – When access patterns are unknown, or changing over time, use this feature to automatically transition objects.



Amazon S3 Lifecycle – Expiring Objects



S3 Lifecycle configurations allows an object to be given a set expiry (deletion)

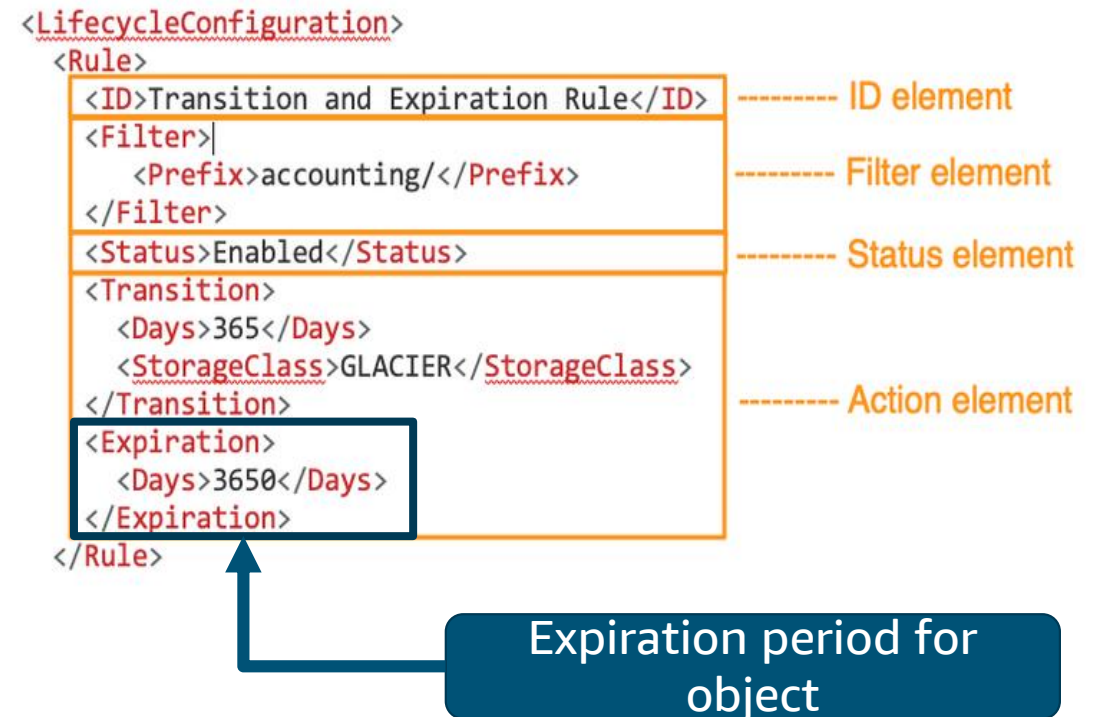
Example Scenarios

Temporary log data– Application log data will be needed and used for a 1-week period, after which it can be deleted. Can be left in S3 Standard storage class, with a 7-day expiration period added.

Data retention policies – Certain objects with business or regulatory requirements for data retention can be transitioned into a S3 Glacier storage class with a 10-year expiration period added.

Minimum Duration Charge

- S3 Standard-IA or S3 One Zone-IA – 30 Days
- S3 Glacier Flexible Retrieval – 90 days
- S3 Glacier Deep Archive – 180 days



S3 Glacier Classes – Cost considerations



Storage Overhead

For each object:

- **8KB** storage allocated for name and other metadata
- **32K** storage allocated for index related metadata

Aggregate smaller objects to reduce.

Days Archived

- Prorated fees charged for objects deleted or overwritten early
- Ensure it is a true archive, and that the object is *ready* to be archived

Transition Requests

- Each object transitioned incurs a transition fee
- Lifecycle policies can filter smaller objects (<128K) to reduce fees

Data Restoration

- Costs incurred for each object restored from archive
- Temporarily restored, must be copied into new S3 bucket for permanent restoration

Amazon S3 Versioning



Keep multiple versions of an object in one bucket so that you can restore objects that are accidentally deleted or overwritten.

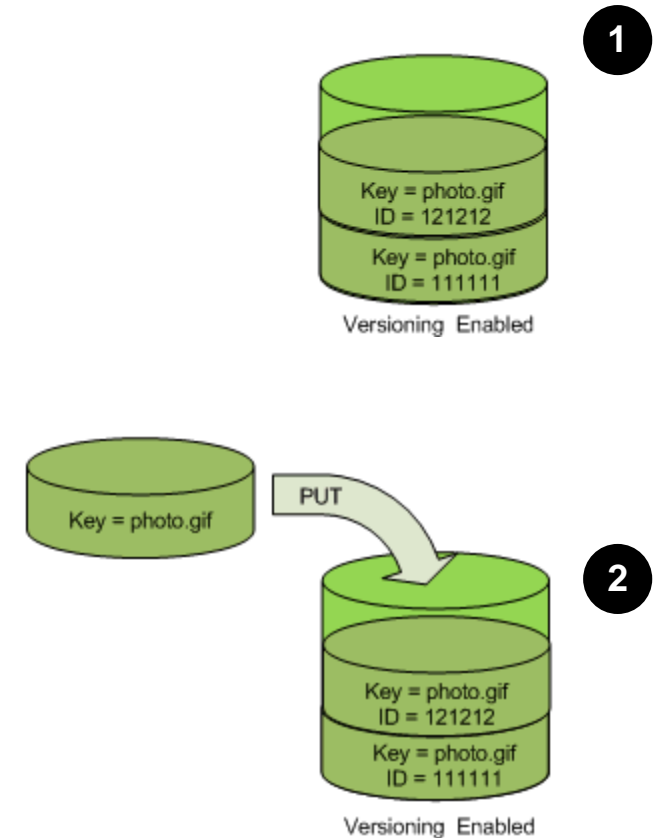
Version IDs

Versioning enabled buckets will automatically generate a unique version ID for the object being stored. Objects with a the same key (object name) will receive different version IDs.

Version ID is set to *null* when versioning is not enabled.

Enabling versioning on an existing bucket will not change the **null** value of existing objects.

Previous versions of an object can be retrieved if the object has been accidentally overwritten or deleted.



Amazon S3 Versioning & Expiration or Deletion



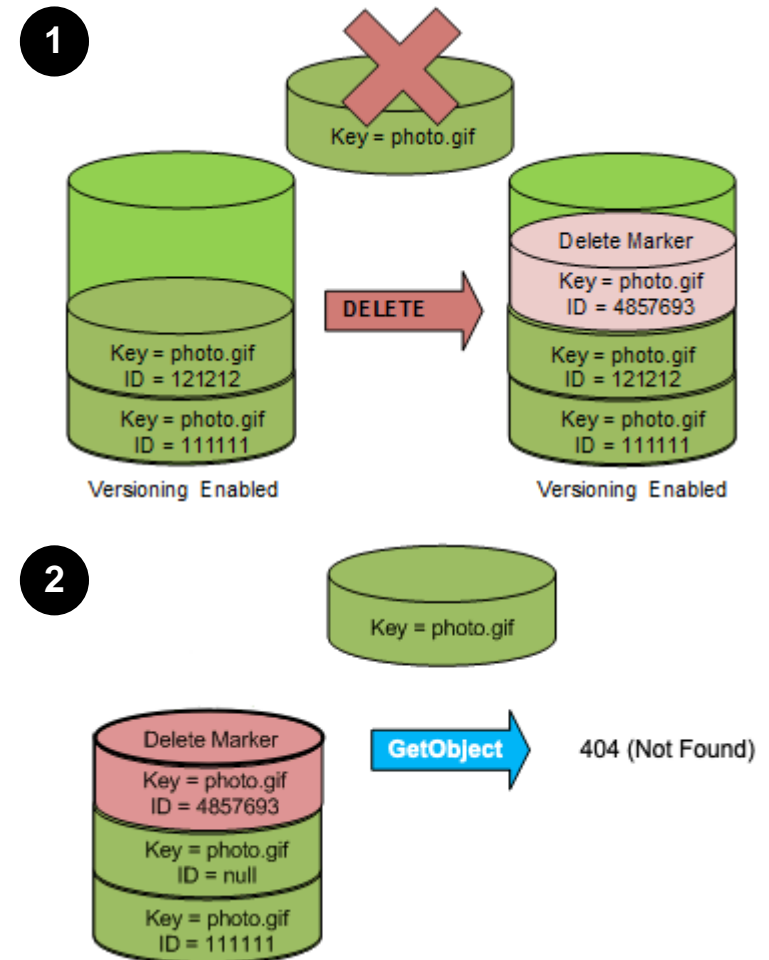
Object expiration or deletion behavior given S3 Versioning

Behavior Scenarios

Nonversioned bucket – Queues the object for removal and removes it asynchronously, permanently removing the object.

Versioning-enabled bucket – If the current object version is not a delete marker, Amazon S3 adds a delete marker with a unique version ID. This makes the current version noncurrent, and the delete marker the current version.

Versioning-suspended bucket – Creates a delete marker with null as the version ID. This delete marker replaces any object version with a null version ID in the version hierarchy, which effectively deletes the object.



Amazon DynamoDB - Time to Live (TTL)



Cost effective method for deleting items that are no longer relevant

How it Works

Define a per-item expiration timestamp that indicates when an item is no longer needed. DynamoDB automatically deletes expired items within a few days of their expiration time, without consuming write throughput.

Configuration

1. Enable TTL on a table in DynamoDB
2. Define an attribute to store the TTL timestamp
3. Timestamp is in the seconds granularity
4. Expiration time can be computed and saved in the attribute when items are created or updated

DynamoDB > Tables > Music > Turn on Time to Live (TTL)

Turn on Time to Live (TTL) [Info](#)

TTL settings

TTL attribute name
The name of the attribute that will be stored in the TTL timestamp.

Between 1 and 255 characters.

Preview
Confirm that your TTL attribute and values are working properly by specifying a date and time, and reviewing a sample of the items that will be deleted by then. Note that preview may show only some of the relevant items.

Simulated date and time
Specify the date and time to simulate which items would be expired.

Epoch time value ▼ September 13, 2023, 15:28:52 (UTC-06:00)

ⓘ Activating TTL can take up to one hour to be applied across all partitions. You will not be able to make additional TTL changes until this update is complete.



AWS PARTNER CERTIFICATION READINESS

Domain 2: Data Store Management

Design data models and schema evolution



Design data models and schema evolution

Knowledge of:

- Data modeling concepts
- How to ensure accuracy and trustworthiness of data by using data lineage
- Best practices for indexing, partitioning strategies, compression, and other data optimization techniques
- How to model structured, semi-structured, and unstructured data
- Schema evolution techniques

Skills in:

- Designing schemas for Amazon Redshift, DynamoDB, and Lake Formation
- Addressing changes to the characteristics of data
- Performing schema conversion (for example, by using the AWS Schema
- Conversion Tool [AWS SCT] and AWS DMS Schema Conversion)
- Establishing data lineage by using AWS tools (for example, Amazon SageMaker ML Lineage Tracking)

Amazon Redshift Schema Design



Sort Keys

- Data is stored in **sorted order** according to the **sort key**
- Query optimizer uses **sort order** when it determines optimal query plans
- **AUTO** can be used to allow Amazon Redshift to choose appropriate sort order

Distribution Style

- Query optimizer **redistributes rows** to compute nodes to perform joins and aggregations
- Locate the data where it **needs to be** prior to query run by distribution style selection

Amazon Redshift Schema Design



Automatic Compression

- **ENCODE AUTO** - Amazon Redshift automatically manages compression encoding for all columns in the table.
- **COPY** command analyzes your data and applies compression encodings to an empty table **automatically** as part of the load operation.

Primary / Foreign Keys

- Define primary key and foreign key **constraints** between tables wherever appropriate.
- The query optimizer uses those constraints to generate **more efficient** query plans

AWS Database Migration Service



Trusted by customers to securely migrate 1M+ databases w/ minimal downtime

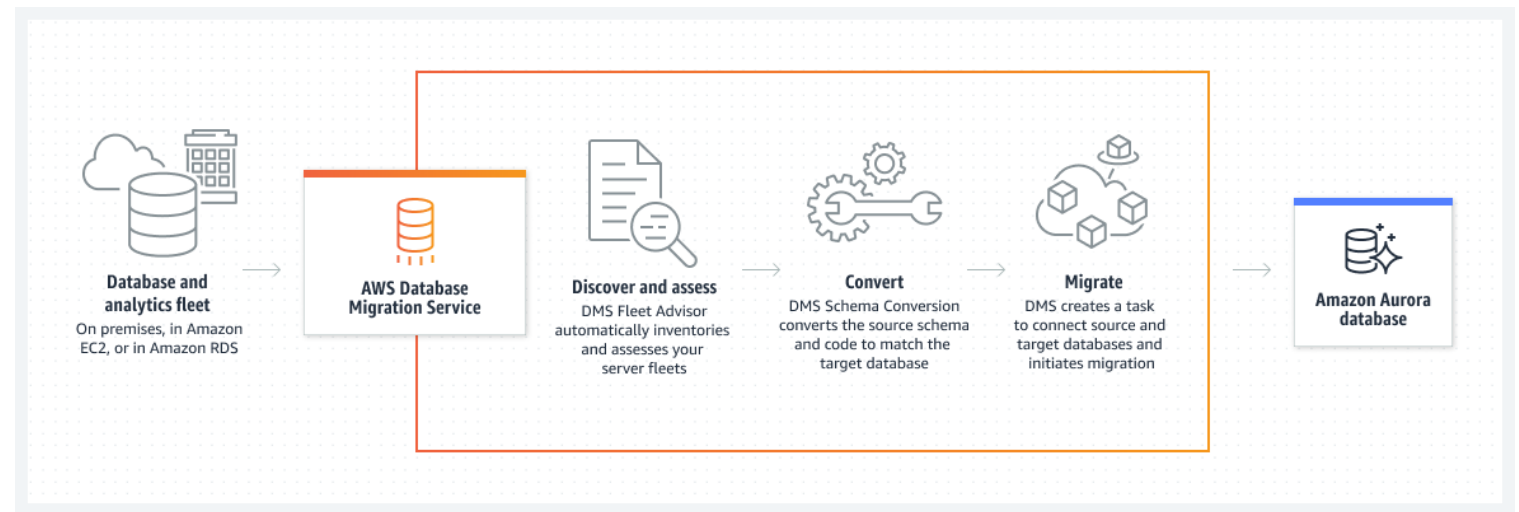
Managed migration and replication service that helps move your database and analytics workloads to AWS quickly, securely, and with minimal downtime and zero data loss

Schema Conversion

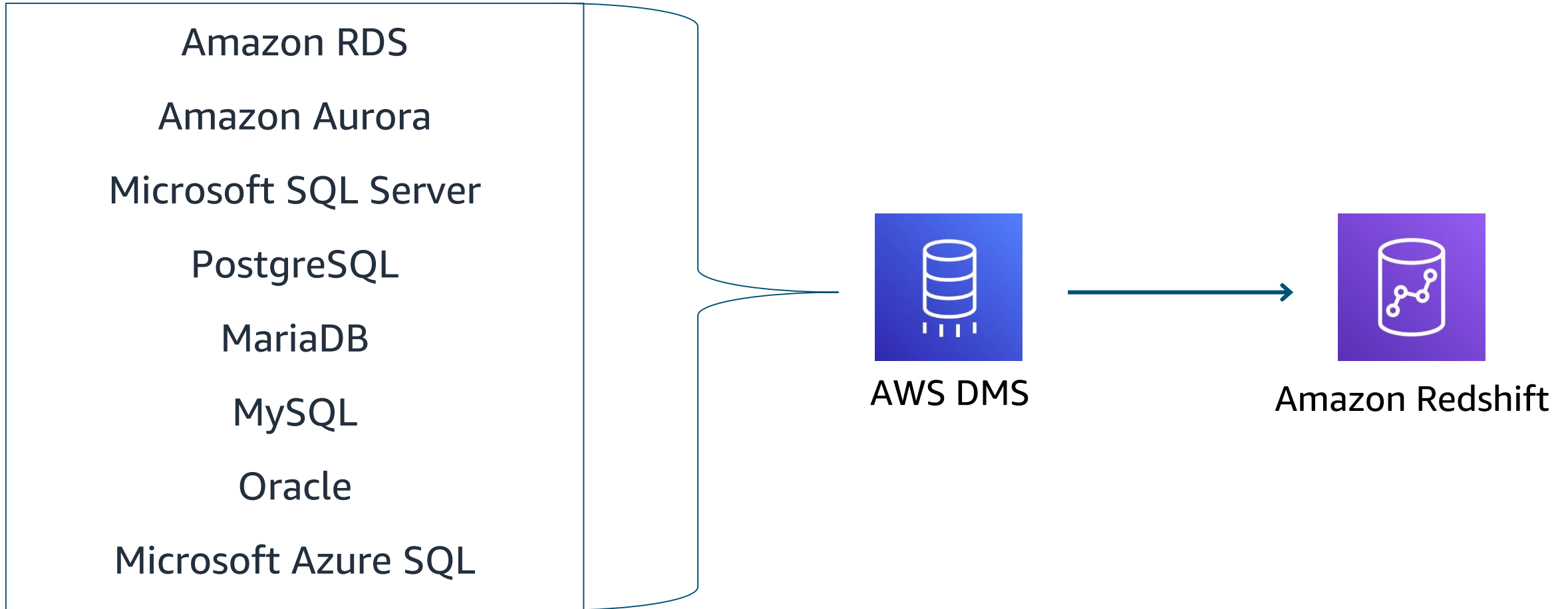
Free-to-use feature allowing for fast, secure, simple schema assessment and conversion at scale, under one fully managed service.

Data lake integration

Build data lakes and perform real-time processing on change data from your data stores. Change data can be streamed to Amazon Kinesis Data Streams using AWS DMS.



Migrating data to Amazon Redshift with AWS DMS



AWS DMS Schema Conversion



Use DMS Schema Conversion to assess the complexity of your migration for your source data provider, and to convert database schemas and code objects.

Components

Instance profiles – specifies network and security settings

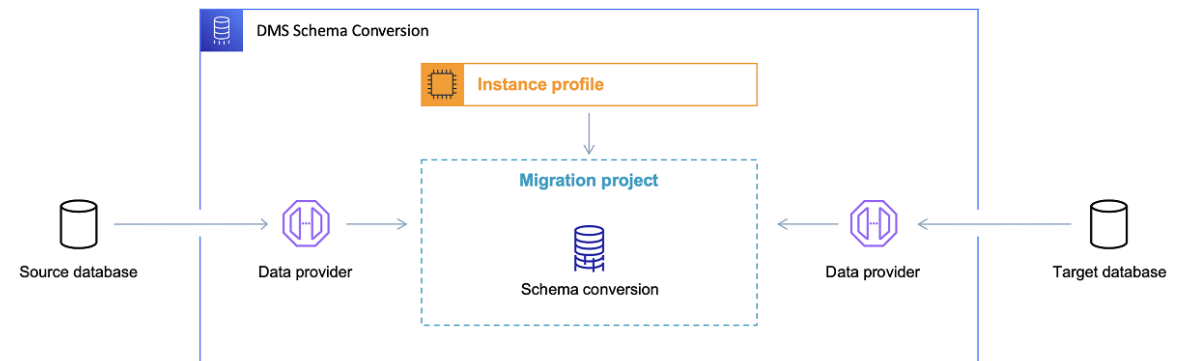
Data providers – stores database connection credentials

Migration project – contains data providers, instance profile, and migration rules

Compatibility

Primarily support conversion to these target databases:

1. Aurora MySQL / PostgreSQL
2. MySQL / PostgreSQL



AWS Schema Conversion Tool (SCT)



Downloadable software to automatically assess and convert source database or data warehouse schema to format compatible with target

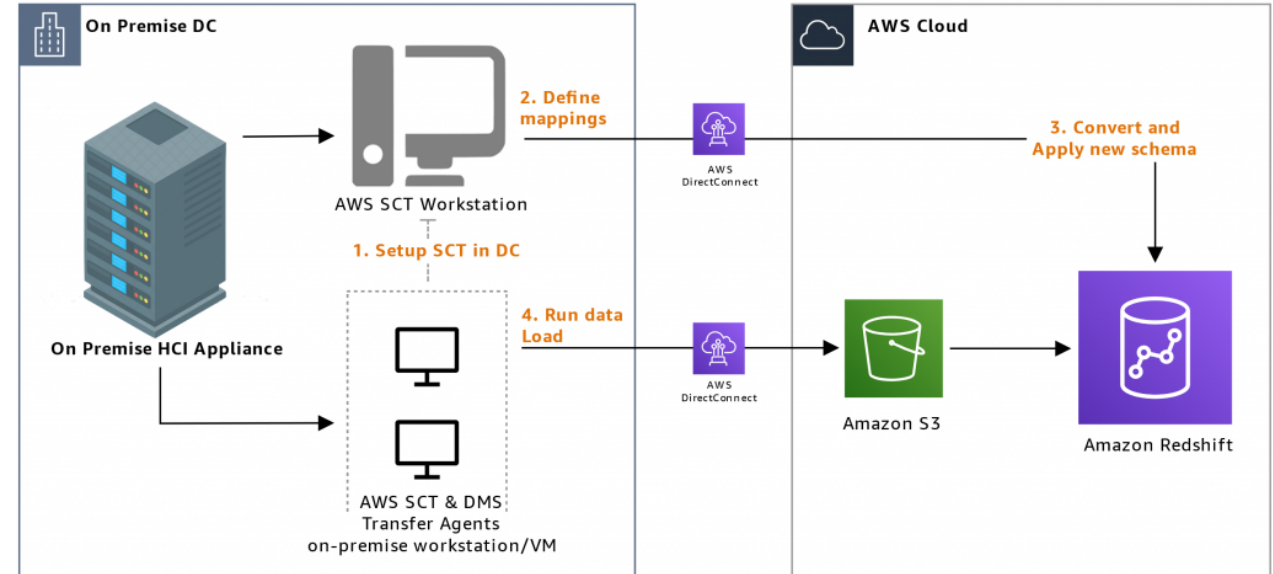
Benefits

Extended compatibility– supports relational OLTP schemas or data warehouse schemas.

Industry standards and compliance – FIPS, FedRAMP

Compatibility

1. Aurora MySQL / PostgreSQL
2. MySQL, PostgreSQL, MariaDB
3. Amazon Redshift
4. Apache Cassandra to DynamoDB
5. ETL Processes





**Thank you for attending
this session**