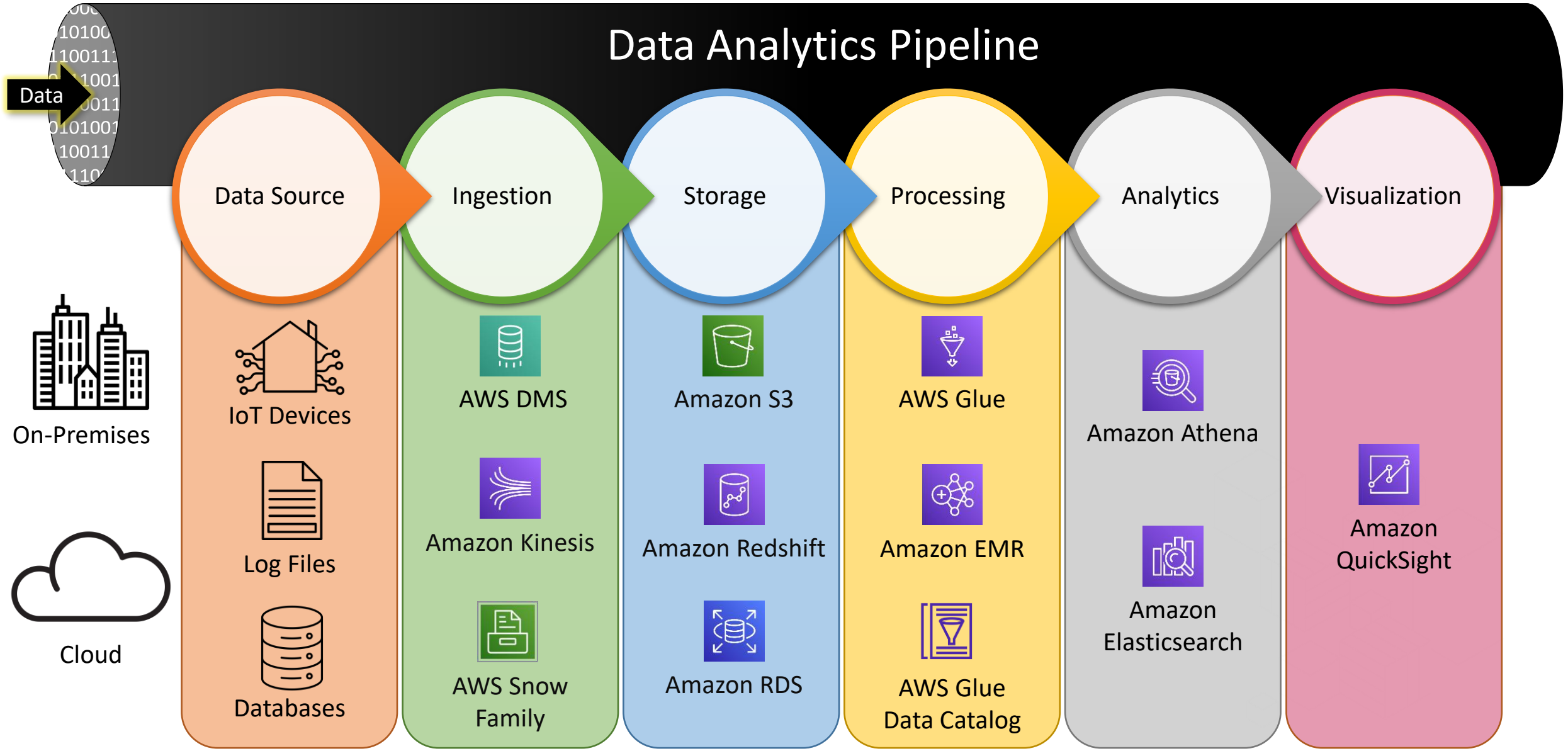
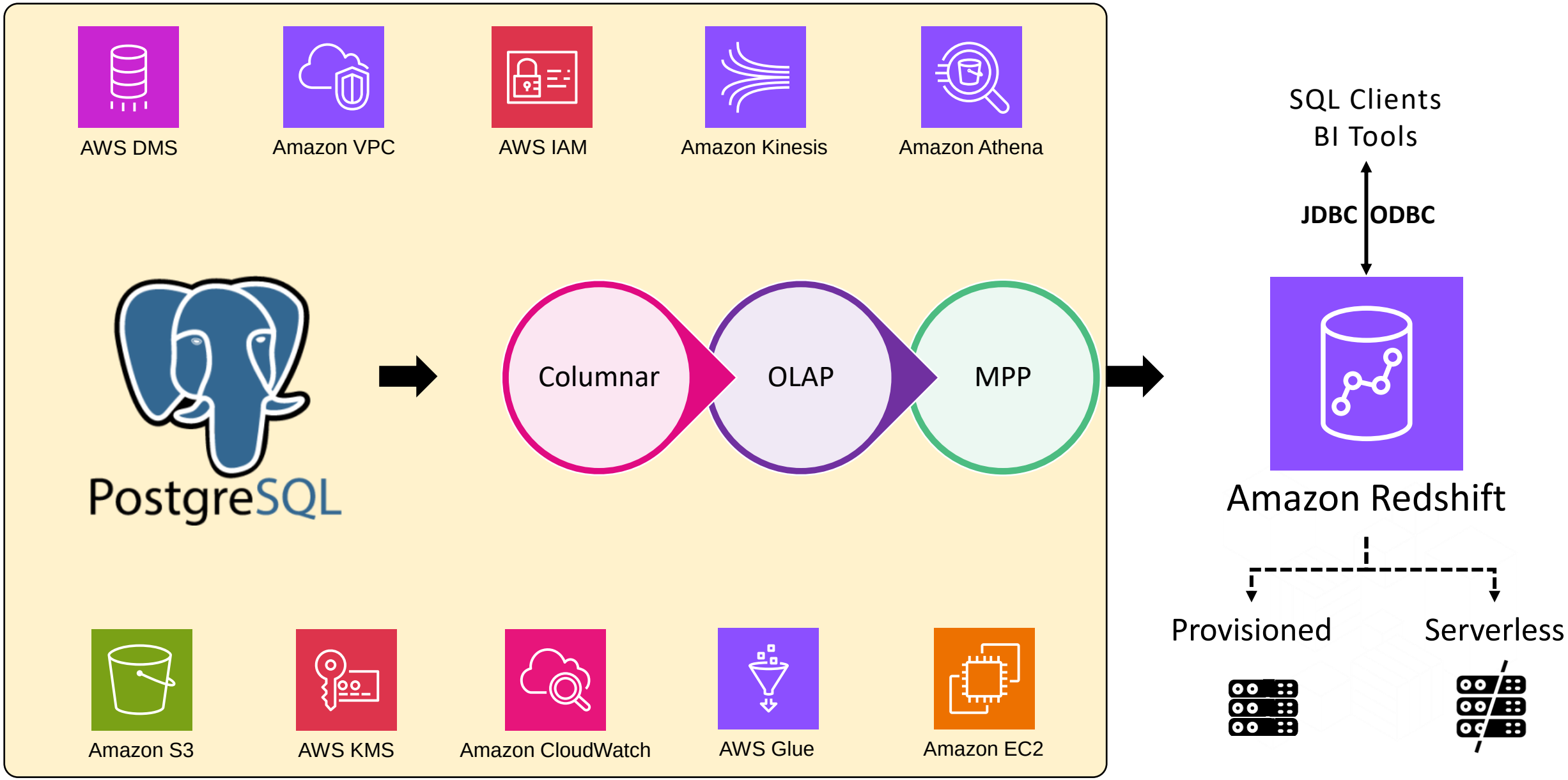
Amazon Redshift

Data Analytics on AWS



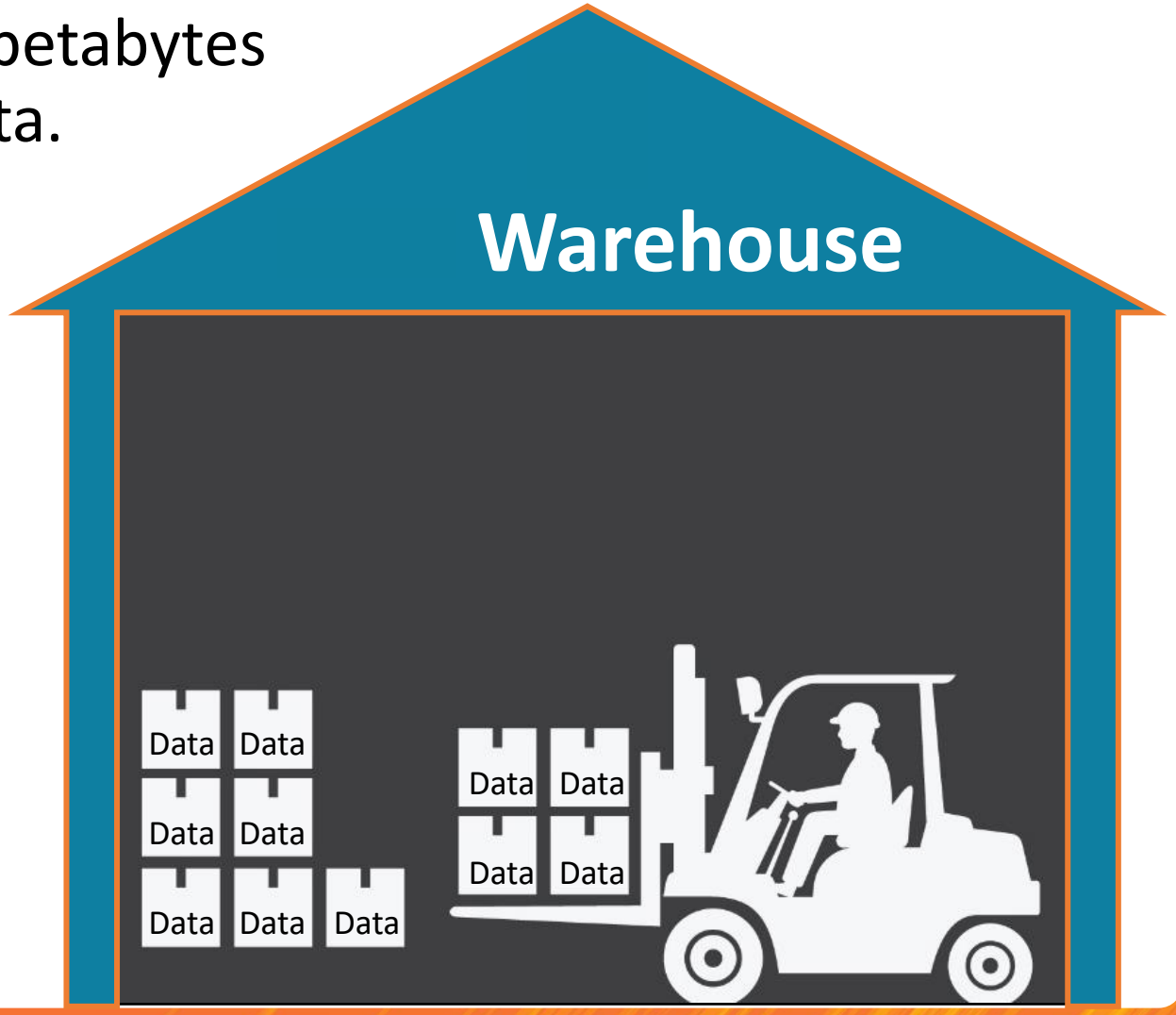
Storage in Kitchen vs. Warehouse

Amazon Redshift



Amazon Redshift

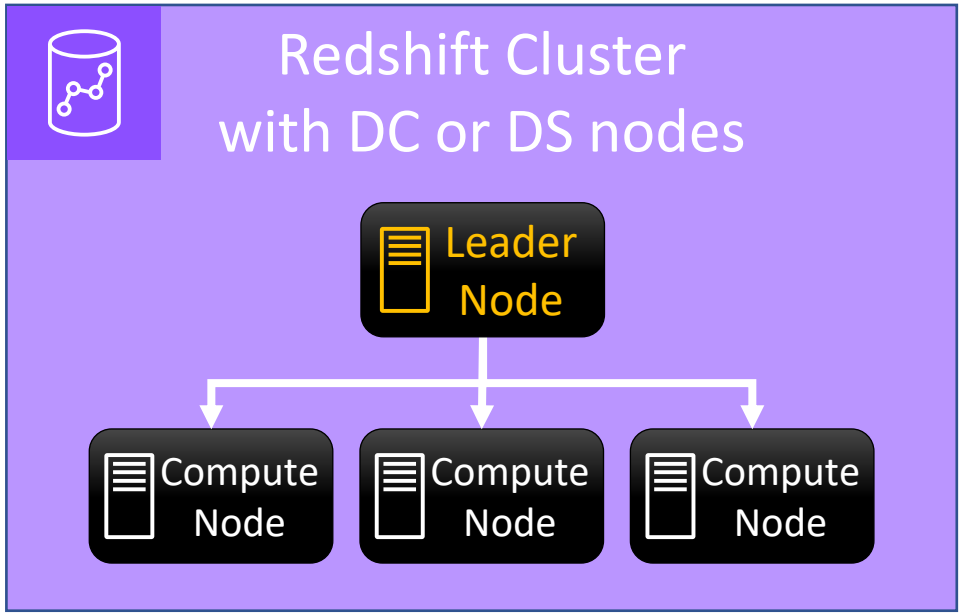
- Amazon Redshift is a purpose-built, fully managed data warehouse.
- It runs high-performance queries on petabytes of structured and semi-structured data.
- Allows you to analyze all the data using standard SQL and existing business intelligence (BI) tools.
- Fine tuned for analytics using features like columnar data storage, data compression, and massive parallel processing.



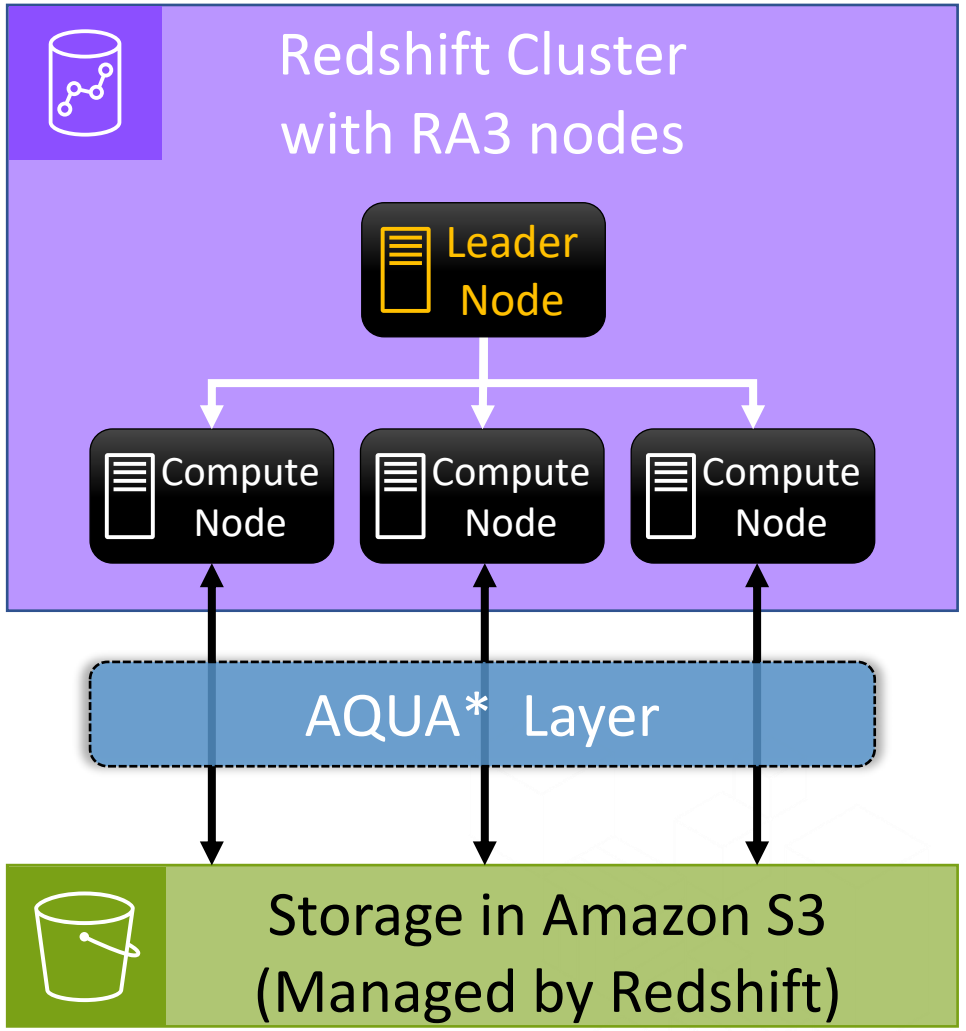
Redshift Architecture

A typical warehouse

Amazon Redshift Architecture



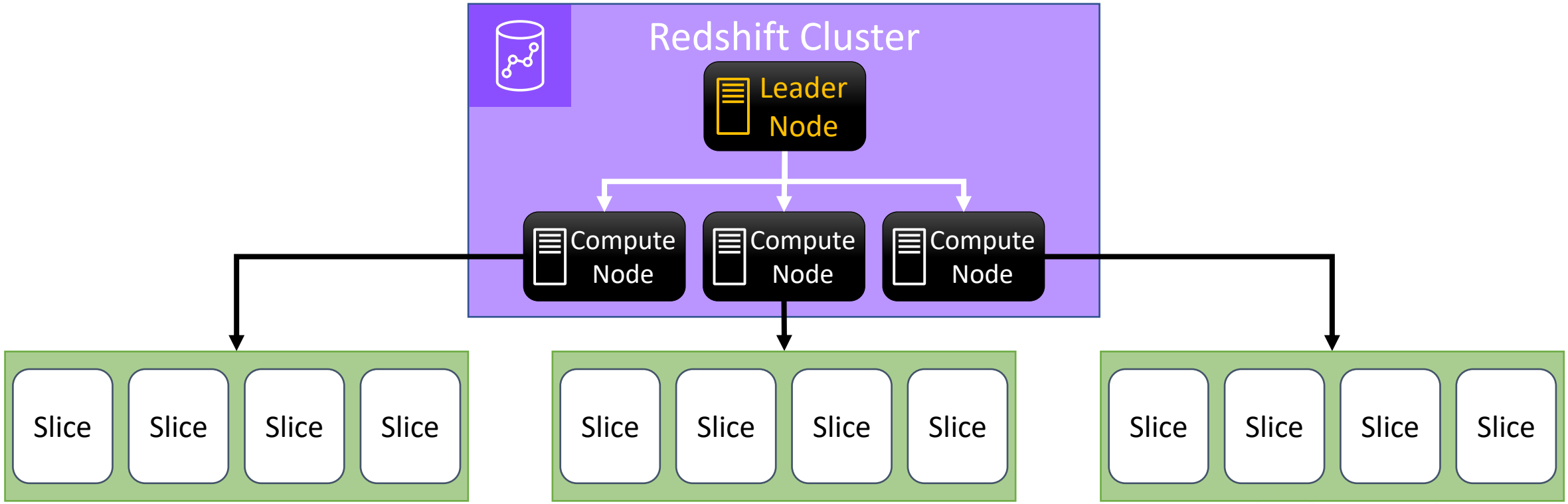
Scale compute and storage together



Scale compute and storage independently

*AQUA (Advanced Query Accelerator)
A Speed Boost for Your Amazon Redshift Queries

Redshift architecture



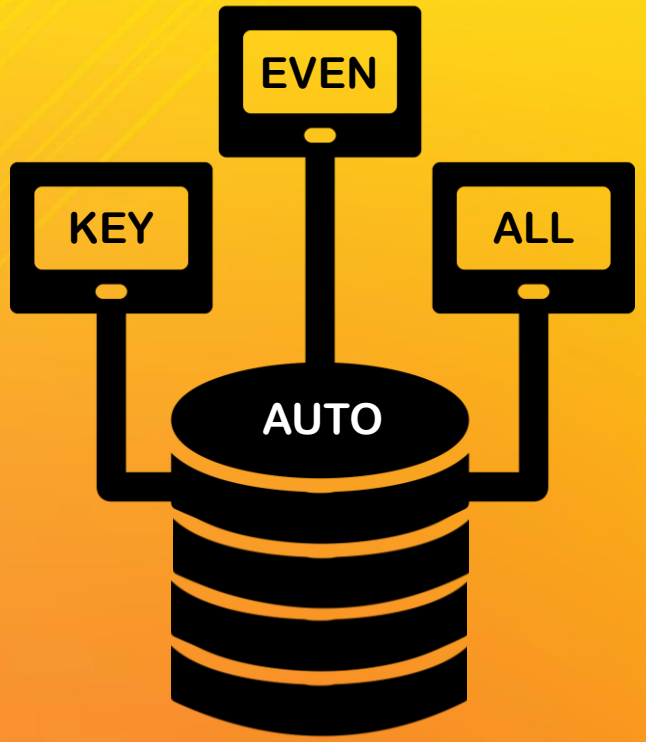
Node type	vCPU	RAM	Slices per node
ra3.xlplus	4	32	2
ra3.4xlarge	12	96	4
ra3.16xlarge	48	384	16

- Each slice is allocated a portion of the node's memory and disk space
- The number of slices per node is determined by the node size of the cluster.

Redshift Query Editor v2

Redshift Query Editor v2

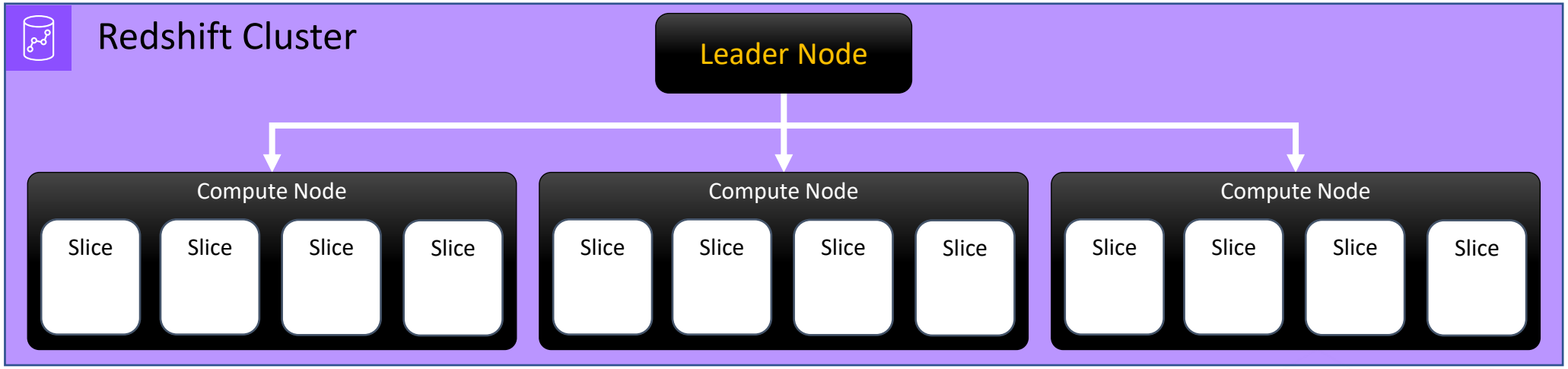
- The query editor v2 is a web-based SQL client application that you use to author and run queries on your Amazon Redshift data warehouse.
- Features:
  - Create databases, schemas, tables, and user-defined functions (UDFs).
  - Create a schedule to run a SQL statement
  - Refresh the materialized views
  - Visualize your results in charts
  - Collaborate with others on your team by sharing your queries
- You can also use SQL notebooks to organize, annotate, and share multiple SQL queries in a single document.



Distribution Style

Distribution Style

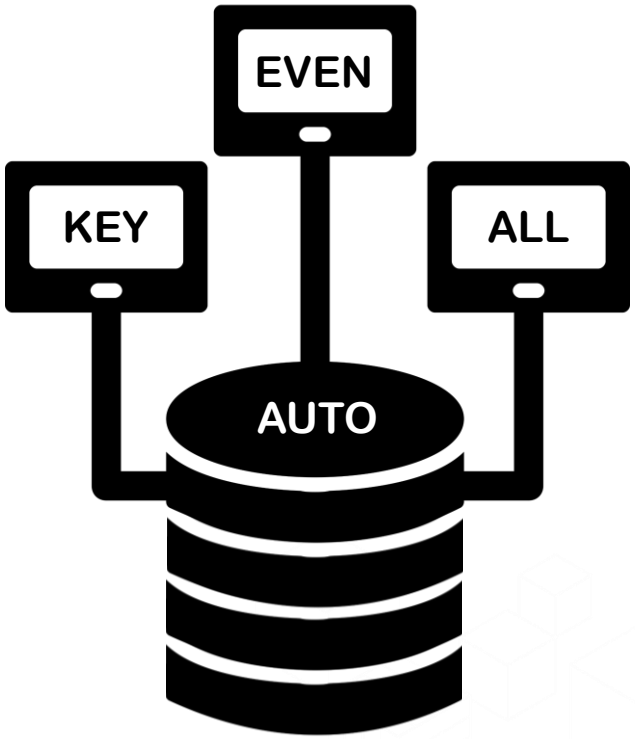
- When you load data into a table, Amazon Redshift distributes the rows of the table to each of the node slices according to the table's distribution style.



- The primary goal in selecting a table’s DISTSTYLE is to evenly distribute the data throughout the cluster for parallel processing.
- The secondary goal in selecting a table’s DISTSTYLE is to minimize the cost of data movement necessary for query processing. To achieve minimization, data should be located where it needs to be before the query is executed.

Distribution Style

- When you create a table, you can designate one of the following distribution styles:
  - AUTO
  - EVEN
  - KEY
  - ALL



- If you don't specify a distribution style, Amazon Redshift uses AUTO distribution.

- In certain scenarios, you can change the distribution style of a table after it is created.

Viewing distribution styles

```
create table public.dist_key (col1 int) diststyle key distkey (col1);
insert into public.dist_key values (1);

create table public.dist_even (col1 int) diststyle even;
insert into public.dist_even values (1);

create table public.dist_all (col1 int) diststyle all;
insert into public.dist_all values (1);

create table public.dist_auto (col1 int);
insert into public.dist_auto values (1);

select "schema", "table", diststyle from SVV_TABLE_INFO where "table" like 'dist%';
```

schema	table	diststyle
public	dist_key	KEY(col1)
public	dist_auto	AUTO(ALL)
public	dist_all	ALL
public	dist_even	EVEN

Redshift distribution styles

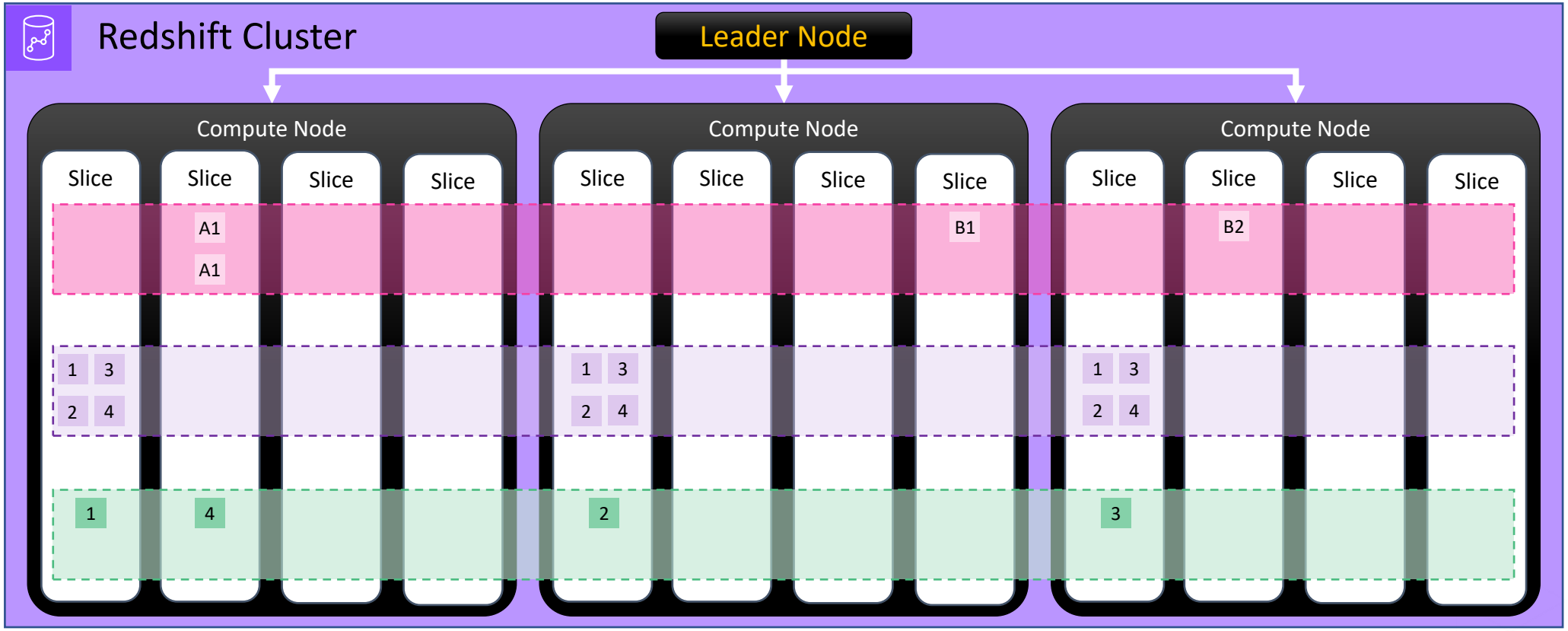


Table 1

1	A1	
2	B1	
3	A1	
4	B2	

KEY

Table 2

1	A1	
2	B1	
3	A1	
4	B2	

ALL

Table 3

1	A1	
2	B1	
3	A1	
4	B2	

EVEN

KEY requires a single column to be defined as a DISTKEY. On ingest, Amazon Redshift hashes each DISTKEY column value, and route hashes to the same slice consistently.

ALL distribution stores a full copy of the table on the first slice of each node.

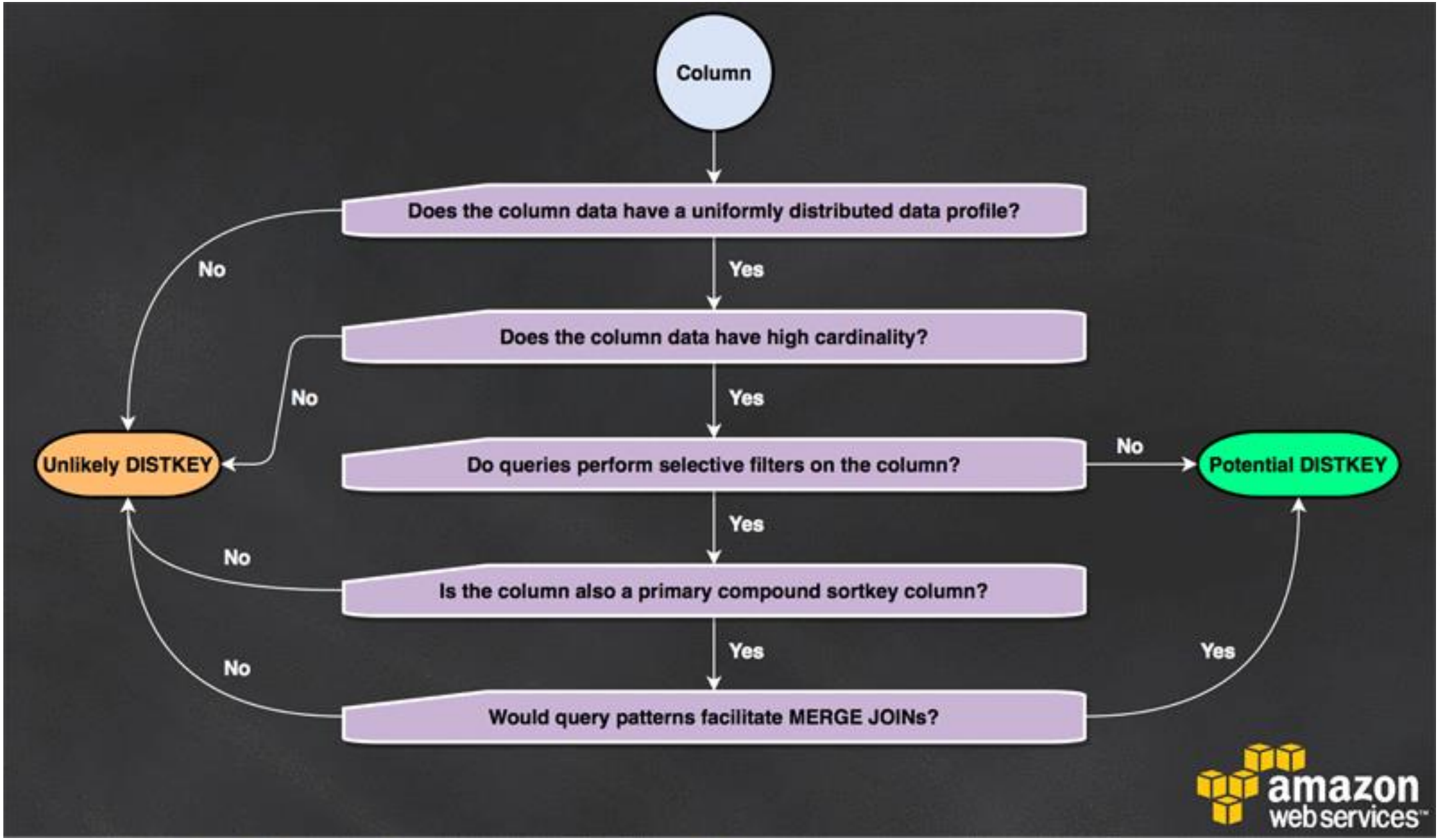
EVEN will do a round-robin distribution of data.

KEY distribution is appropriate for tables that are frequently joined together.

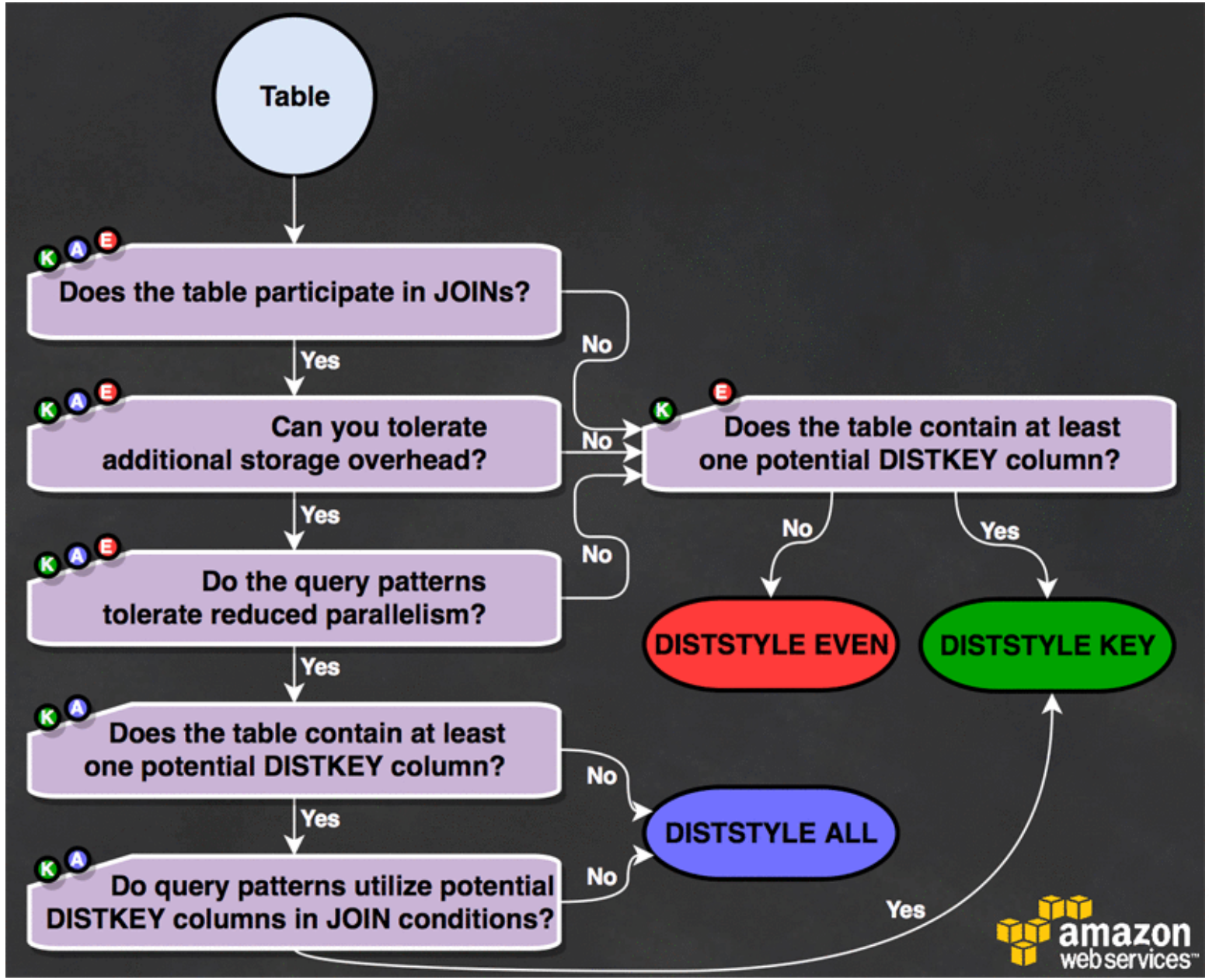
ALL distribution is appropriate only for the tables that are not updated frequently or extensively.

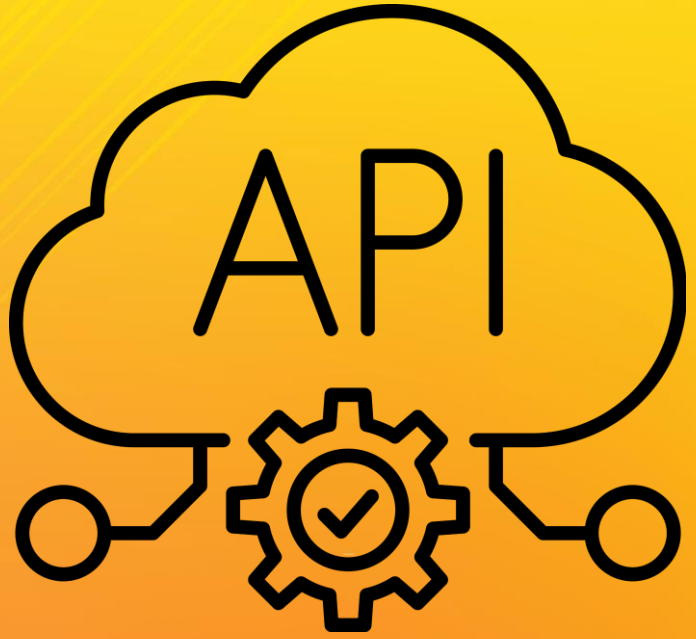
EVEN distribution is appropriate when a table doesn't participate in joins.

Which one to choose? (1)



Which one to choose? (2)

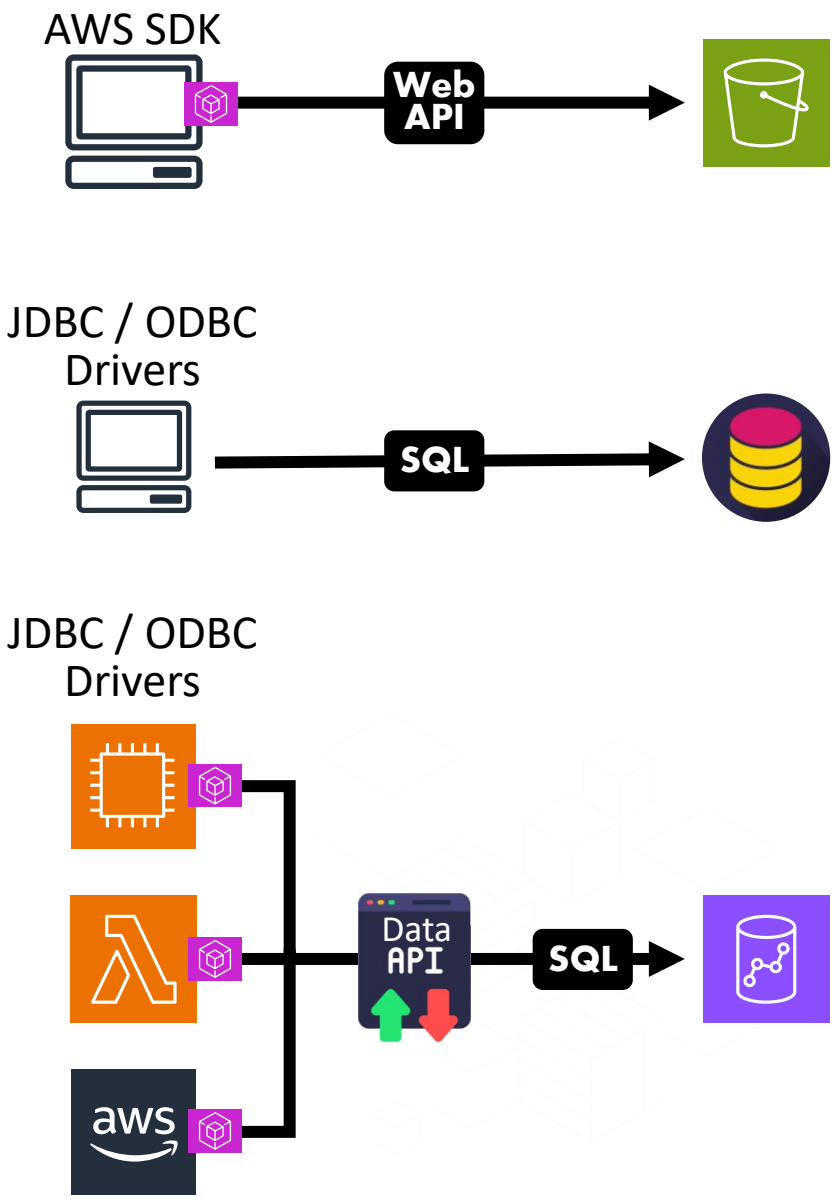




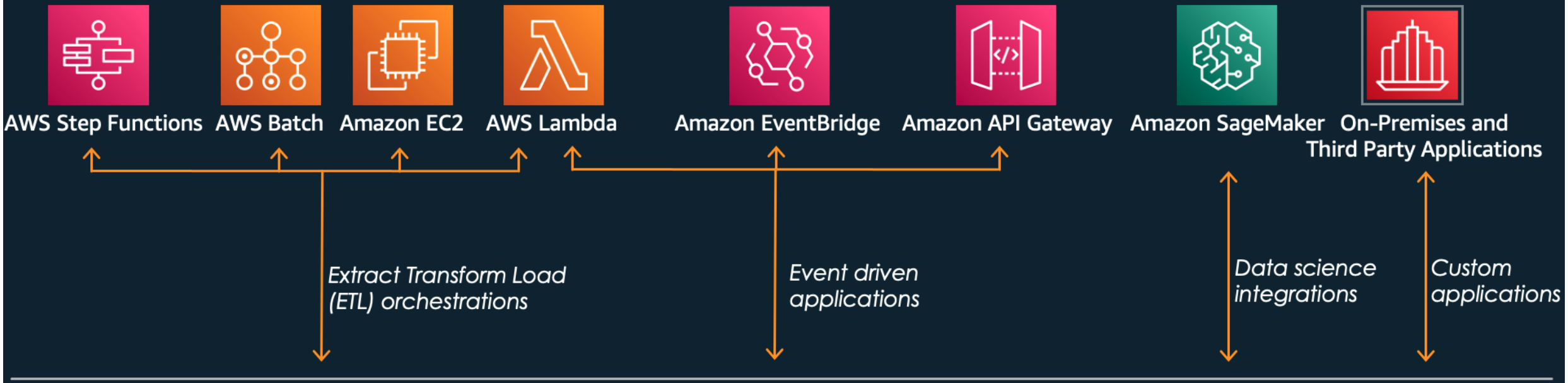
Amazon Redshift Data API

Redshift Data API

- With the Amazon Redshift Data API, you can interact with Redshift Serverless without having to configure JDBC or ODBC.
- The Data API doesn't require a persistent connection to your database. Instead, it provides a secure HTTP endpoint and integration with AWS SDKs.
- You can use the endpoint to run SQL statements without managing connections.
- Calls to the Data API are asynchronous.
- It uses either credentials stored in AWS Secrets Manager or temporary database credentials.



Use Cases



Amazon Redshift Data API

Amazon Redshift

- Federate the credentials from SSO/LDAP provider such as Active Directory / Okta
- Enables event-driven pipelines and scheduled query execution
- Results are stored for 24 hours

- Supports Provisioned clusters and Serverless
- No need to configure drivers, connection pools
- Query, Load and Unload data from CLI/SDK

Examples

- Using AWS CLI

```
aws redshift-data execute-statement
--database [DATABASE]
--query [QUERY]
--secret-arn [CREDENTIALS_ARN]
```

```
aws redshift-data execute-statement
--cluster-identifier my-redshift-cluster
--database dev
--db-user awsuser
--sql 'select data from redshift_table’;
```

- Using SDKs

```
new AWS.RedshiftData().executeStatement({
        ClusterIdentifier: 'my-redshift-cluster',
        Database: 'dev',
        DbUser: 'awsuser',
        Sql: 'select data from redshift_table'
}).promise();
```

JavaScript

```
boto3.client("redshift-data").execute_statement(
ClusterIdentifier = ‘my-redshift-cluster’,
Database = ‘dev’,
DbUser = ‘awsuser’,
Sql = 'select data from redshift_table’)
```

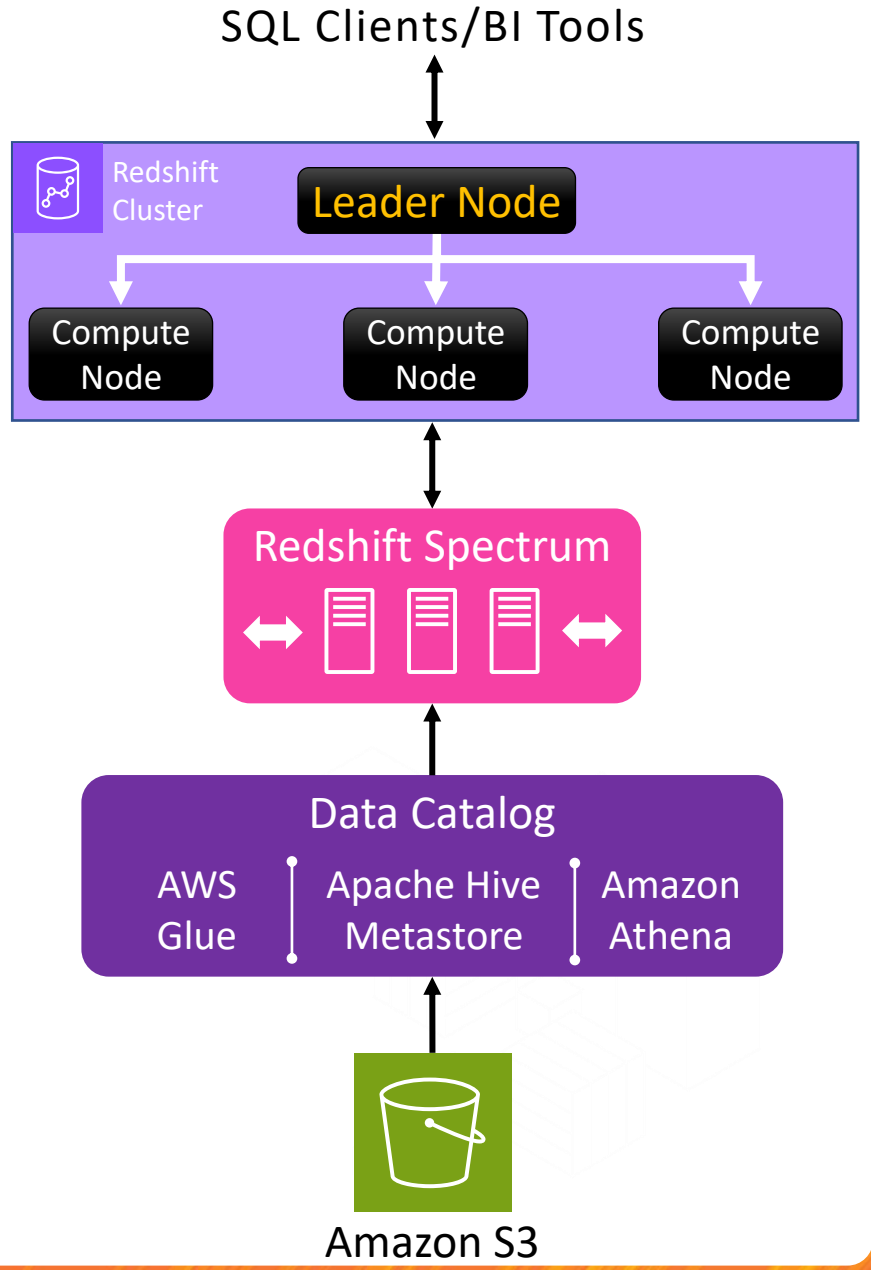
Python

```
aws redshift-data execute-statement
   --region us-west-2
   --secret arn:aws:secretsmanager:us-west-2:123456789012:secret:myuser-secret-hKgPWn
   --cluster-identifier mycluster-test
   --sql "select * from stl_query limit 1"
   --database dev
```

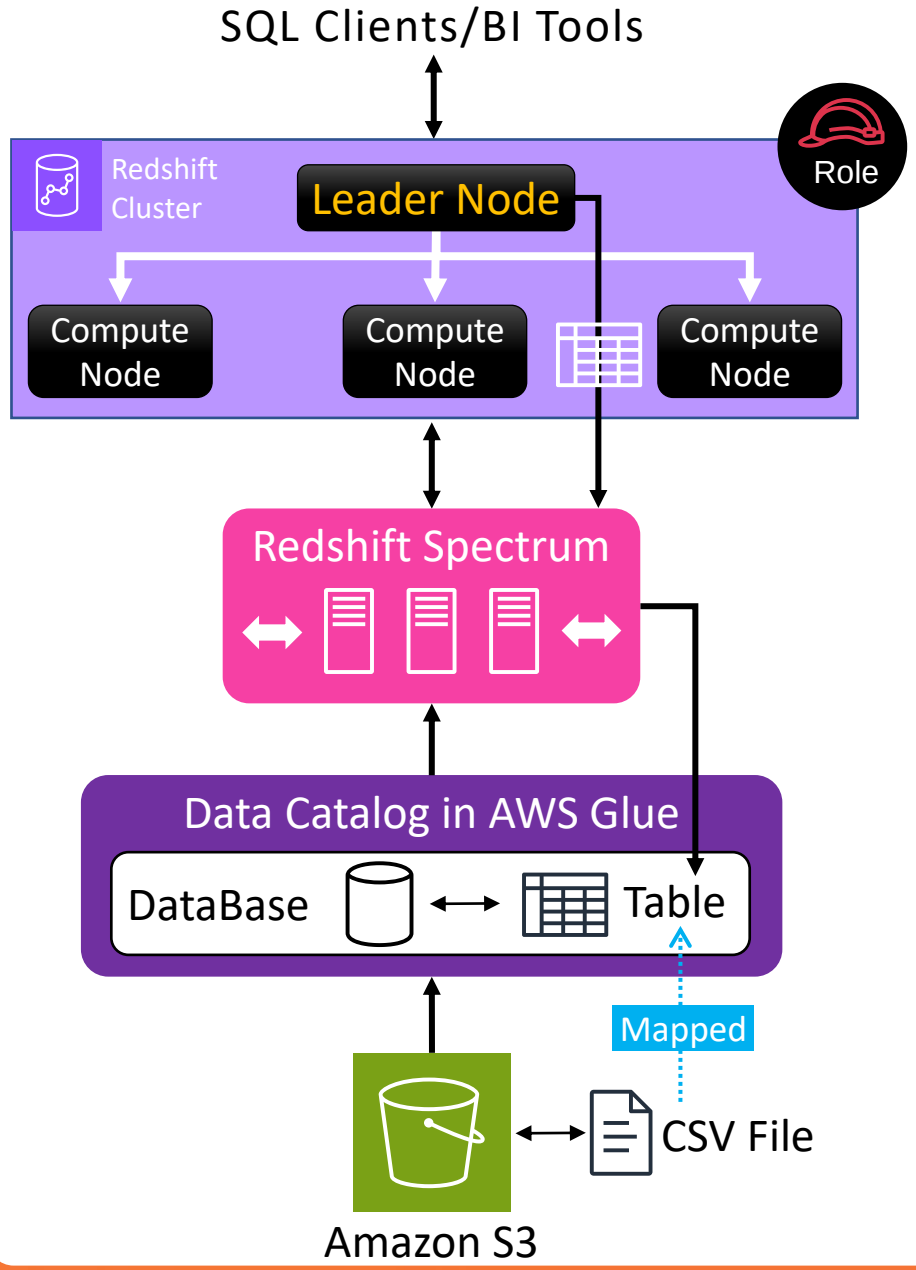
Redshift Spectrum

Amazon Redshift Spectrum

- Using Amazon Redshift Spectrum, you can efficiently query and retrieve structured and semi-structured data from files in Amazon S3 without having to load the data into Amazon Redshift tables.
- Amazon Redshift Spectrum resides on dedicated Amazon Redshift servers that are independent of your cluster.
- You create Redshift Spectrum tables by defining the structure for your files and registering them as tables in an external data catalog.
- The external data catalog can be AWS Glue, the data catalog that comes with Amazon Athena, or your own Apache Hive metastore.



Amazon Redshift Spectrum – Using AWS Glue as External Metastore



1. Create IAM Role with required permissions and assign it to Redshift cluster.
2. Create an external schema in AWS Glue.

```
create external schema s3dataschema
from data catalog database 'dev'
iam_role '<Redshift-Role-ARN>'
create external database if not exists;
```

3. Create a table in external schema based on schema of file you want to query.

```
create external table s3dataschema.countrybusinessindex( country
nvarchar(100), businessindex int ) row format delimited fields
terminated by ',' stored as textfile location 's3://<bucket>';
```

4. Run Query on the data

```
select * from s3dataschema.countrybusinessindex
```

Redshift Serverless

100% Fresh
MILK
Dairy cow milk production
LIGHT
100% Fresh
MILK
Dairy cow milk production
100% Fresh
MILK
Dairy cow milk production
1lt
Fresh Milk
PASTEURIZED
1lt
Fresh Milk
PASTEURIZED

The supply chain

MILK
MILK
MILK

Why serverless?

YOU focus on insights

- Get insights from data in seconds
- Experience consistently high performance
- Get started with no modifications
- Pay for what you use

Amazon Redshift Serverless

aws takes care of the rest

- Automatic scaling
- Compute provisioning
- Automated patching and failover
- Advanced monitoring
- Backup, recovery
- Smaller RPU configurations
- Routine maintenance
- Table level restore from snapshots
- Query level scheduling

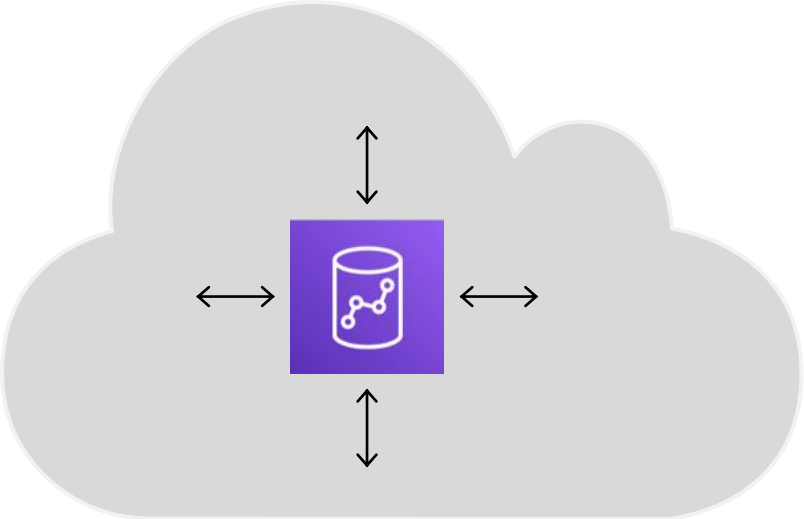
Workgroup and Namespaces

To isolate workloads and manage different resources in Amazon Redshift Serverless, you can create namespaces and workgroups and manage storage and compute resources separately.

- Workgroup: Collection of compute resources
  - RPUs, VPC subnet groups, security groups, and an endpoint.
  - Access and usage limits.
- Namespace: Collection of storage object
  - Database objects - schemas, tables, users, permissions encryption key
  - Other resources - datashares, recovery points, and usage limits.

Redshift Processing Units (RPUs)

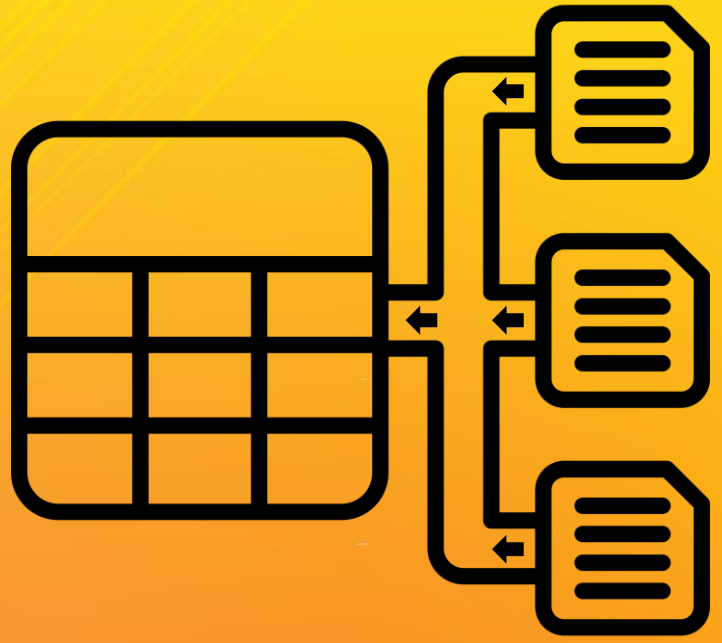
- Amazon Redshift Serverless measures data warehouse capacity in Redshift Processing Units (RPUs). RPUs are resources used to handle workloads.
- You can set a base capacity in Redshift Processing Units (RPUs). One RPU provides 16 GB of memory.
- The default base capacity for Amazon Redshift Serverless is 128 RPUs. You can adjust the Base capacity setting from 8 RPUs to 512 RPUs in units of 8 (8,16,24...512)
- You pay for workloads in RPU-hours per second (with a minimum charge of 60 seconds)



Amazon Redshift – Provisioned vs. Serverless

	Provisioned	Serverless
Use case	• Continuous streaming ingestion • ETL processing • Analytical queries • Reporting	• Continuous streaming ingestion • ETL processing • Analytical queries • Reporting • **Self-service Analytics**
Workload type	• Predictable with High utilization	• Volatile, sporadic with unpredictable utilization

Cost Optimization	• Node selection
	  • On-Demand or Reserved Instances
	  • Compute, Storage capacity
	• Elastic Resize
	• Concurrency Scaling
	• Pause and Resume
		• Redshift Processing Units (RPUs):
		• Base capacity and Max

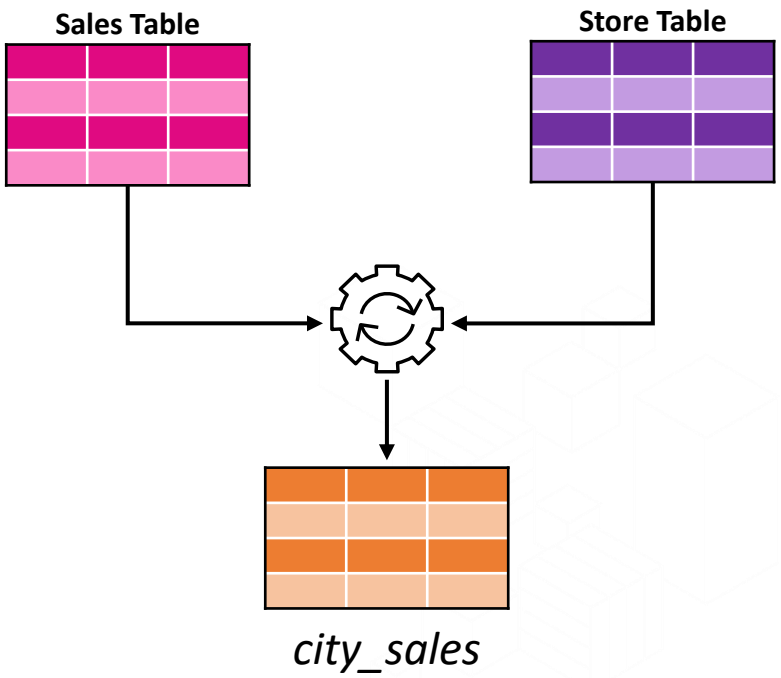


Materialized Views

Materialized views

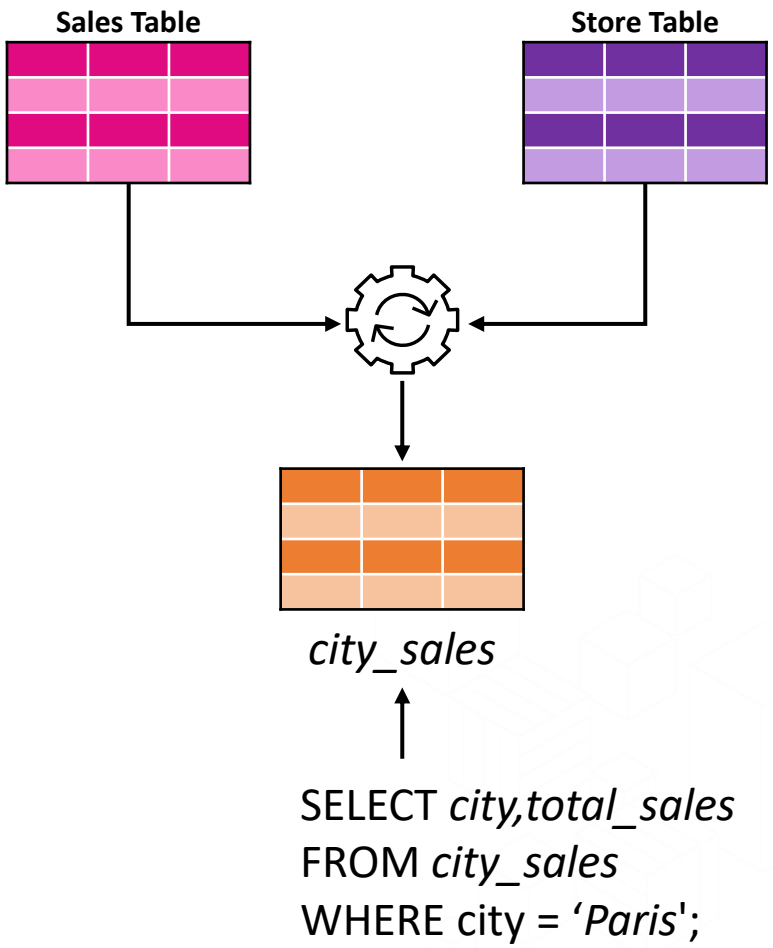
- Consider the scenario where a set of queries is used to populate dashboards, such as Amazon QuickSight. The queries will be predictable and repeated over and over again, which may not be very efficient.
- Materialized views are database object that persists the precomputed result set of an SQL query.
- Materialized views are especially useful for speeding up queries that are predictable and repeated.
- Automatic query rewrite leverages relevant materialized views.

```
CREATE MATERIALIZED VIEW city_sales AS (
  SELECT st.city, SUM(sa.amount) as total_sales
  FROM sales sa, store st
  WHERE sa.store_id = st.id
  GROUP BY st.city
);
```



Materialized views

- You can use a materialized view in any SQL query by referencing the materialized view name as the data source, like a table or standard view.
- Because the query results are precomputed, there's no need to access the underlying tables (category, event, and sales). Amazon Redshift can return the results directly from *city_sales*.
- Amazon Redshift can automatically rewrite SQL queries that don’t explicitly reference existing materialized views that are used for performance improvements.
- Refreshed (automatically or manually) when data in underlying tables changes.



Refreshing Materialized Views – Automated and Manual method

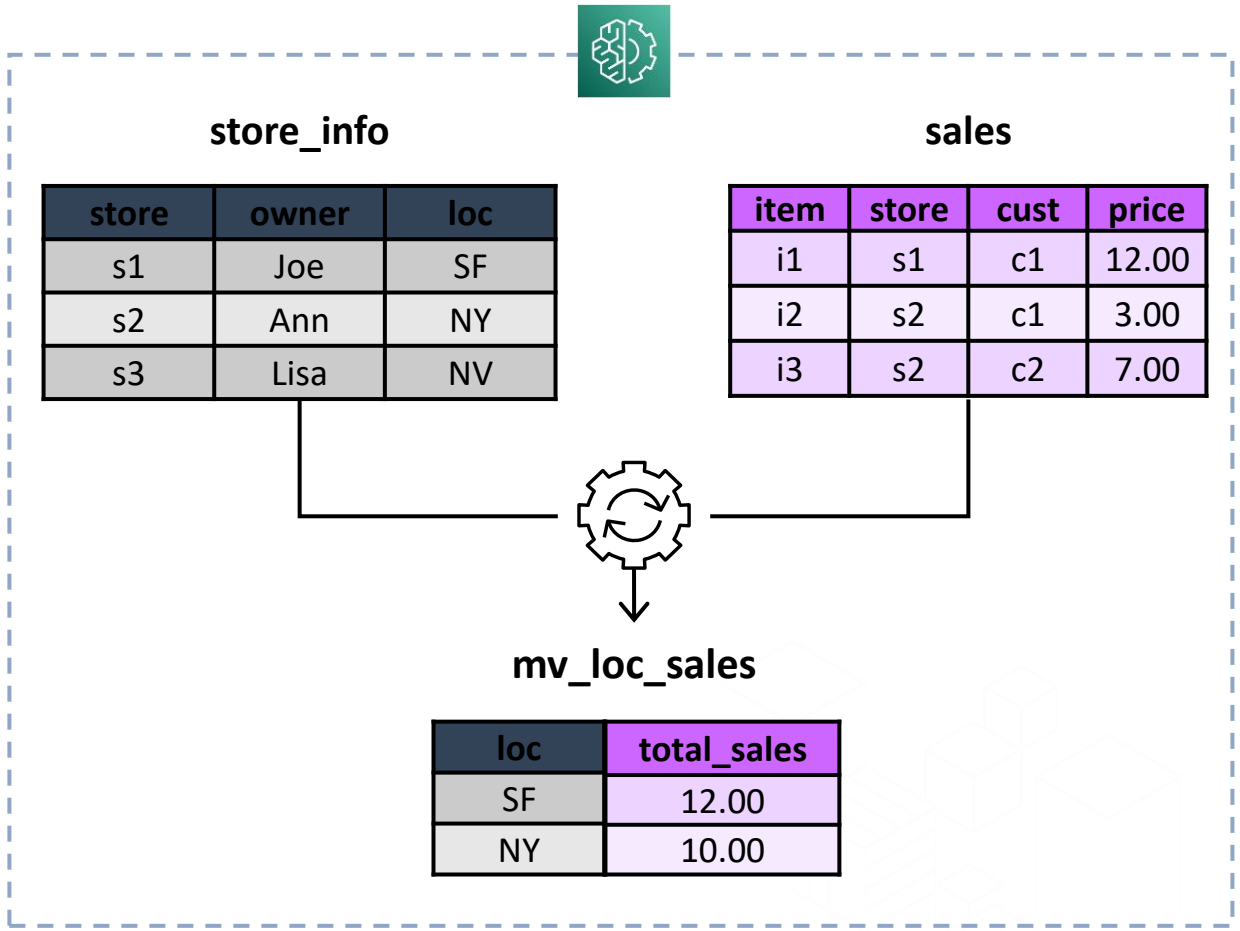
- While creating you can setup auto-refresh option

```
CREATE MATERIALIZED VIEW mv_name
[ BACKUP { YES | NO } ]
[ table_attributes ]
[ AUTO REFRESH { YES | NO } ]
AS query
```

- You can still refresh a materialized view explicitly using REFRESH MATERIALIZED VIEW command even if you haven't enabled autorefresh for the materialized view.
- In many cases, Amazon Redshift can perform an incremental refresh. If an incremental refresh isn't possible, then Amazon Redshift performs a full refresh.

Automated Materialized Views (AutoMV)

- Automated creation and deletion of MVs, with incremental refresh.
- Continual monitoring using ML algorithms to identify queries that will benefit from having an MV.
- The queries are rewritten by Amazon Redshift to read data from the auto materialized views instead of the base tables. This process is completely transparent to the user.
- Automatically deletes MVs that are no longer useful.



store	owner	loc
s1	Joe	SF
s2	Ann	NY
s3	Lisa	NV

item	store	cust	price
i1	s1	c1	12.00
i2	s2	c1	3.00
i3	s2	c2	7.00

loc	total_sales
SF	12.00
NY	10.00

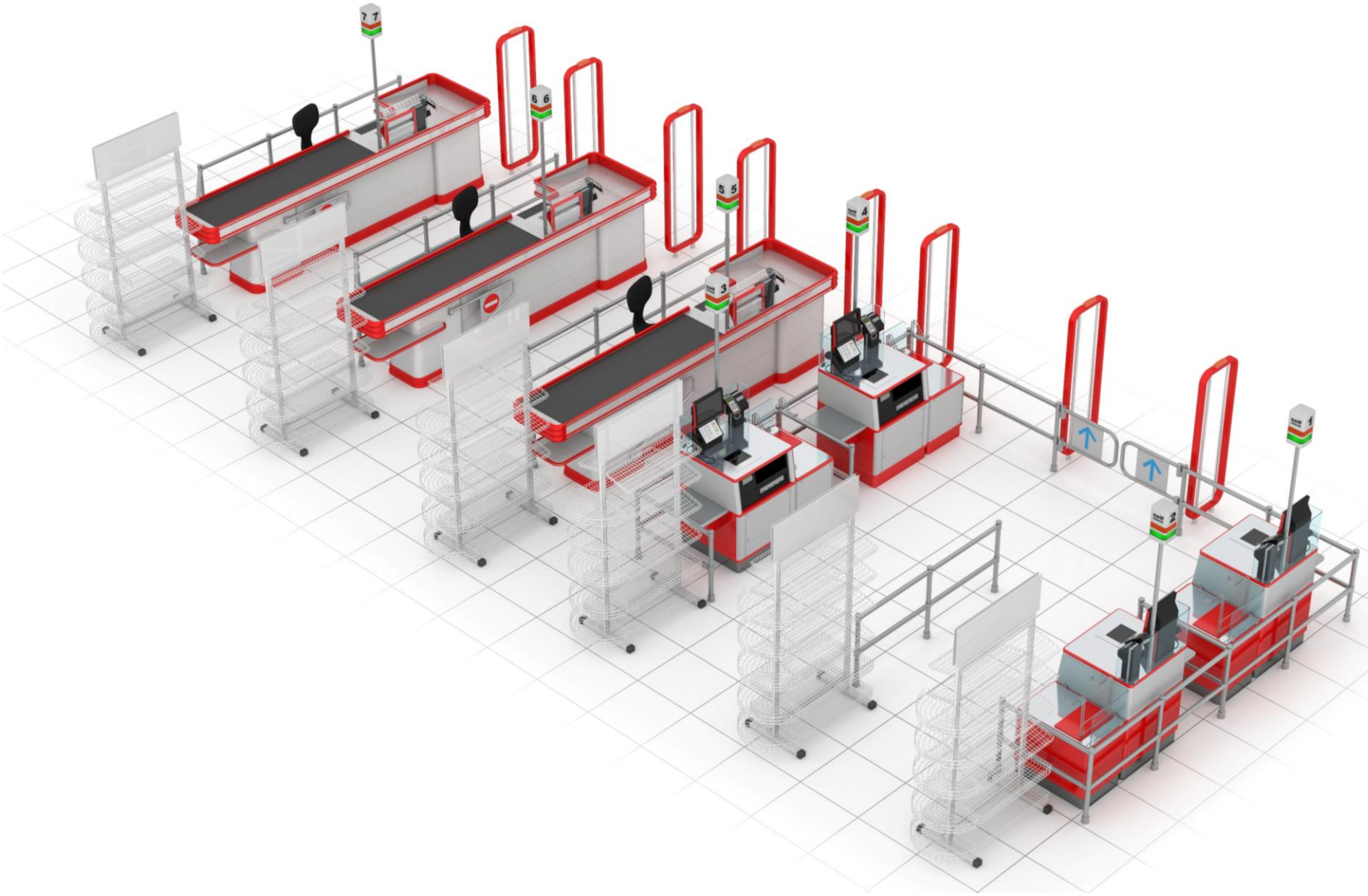
Workload Management

Queries are like shopping baskets

Processing a query is like checking out the shopping cart

How long it takes to check out a shopping cart depends on what is in the shopping cart.

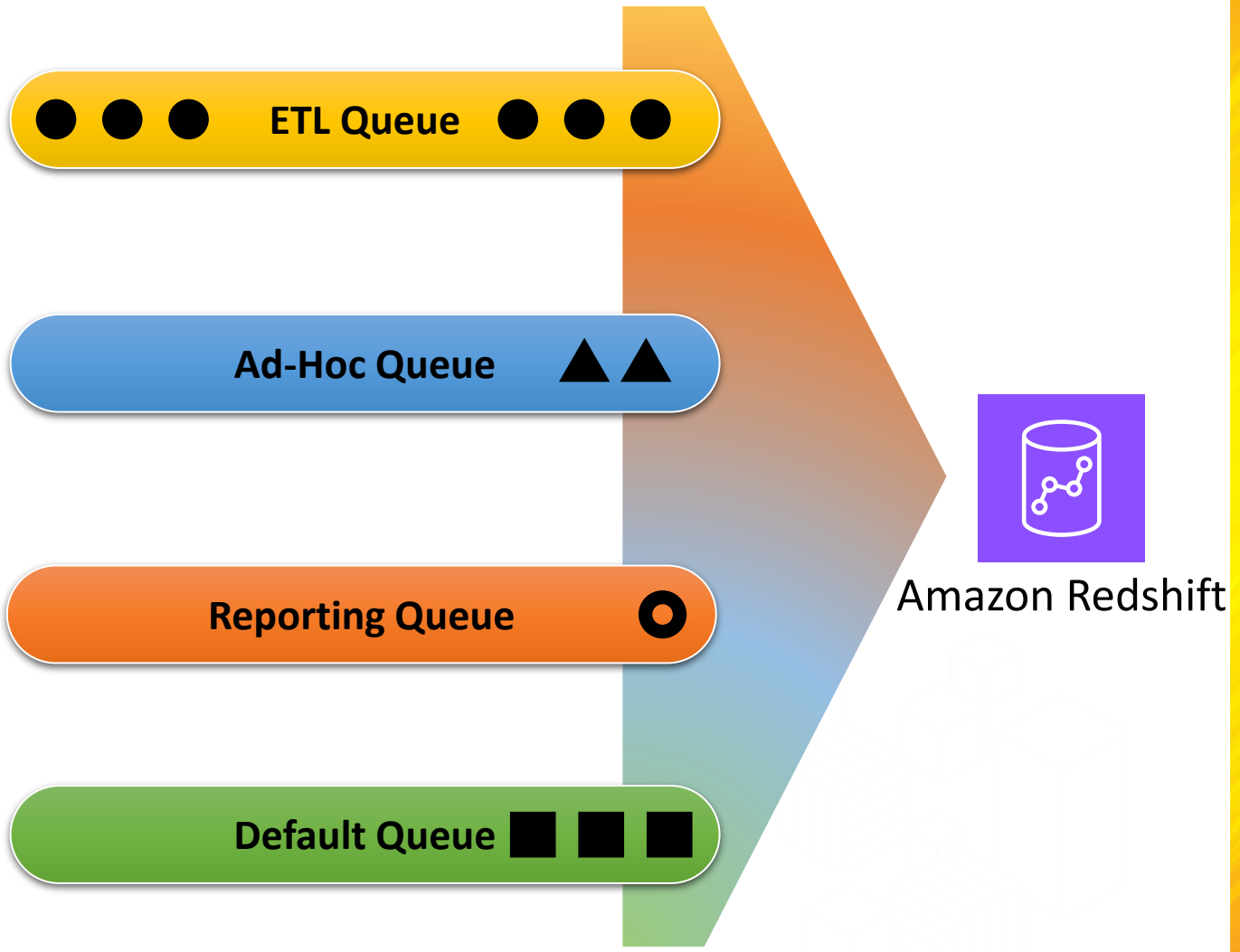
Multiple checkout queues based on different criteria and different priority



- No. of items
- Age
- Families with kids
- Loyalty cards
- Other parameters

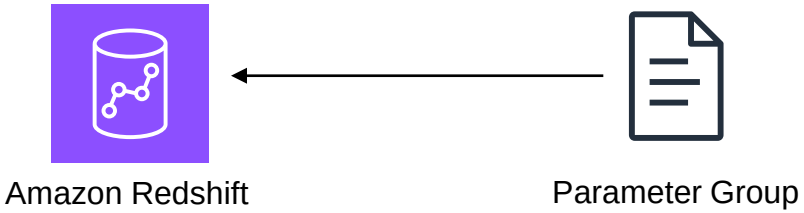
Redshift workload management (WLM)

- WLM is used to define multiple query queues and to route queries to the appropriate queues at runtime.
- WLM optimizes query performance by allocating cluster resources to query queues.
- Each queue will be configured to have its own memory%, query slot count and monitoring rules.
- Default queue will handle queries which doesn't fall under any of the user created queue.



WLM Configuration

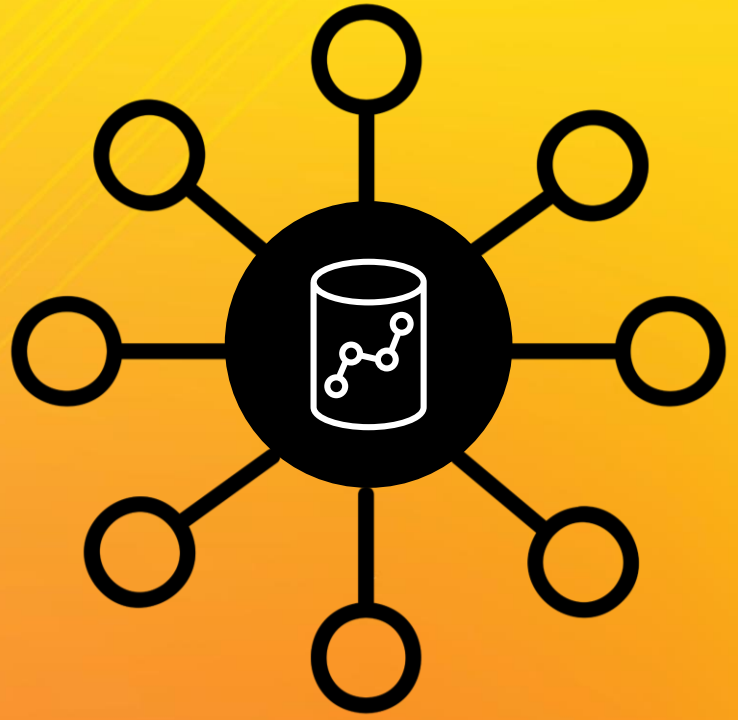
- A parameter group contains a WLM configuration and a set of cluster parameters. You associate a parameter group with each cluster that you create.



- You can switch WLM Mode using parameter group.
- Automatic WLM
  - The system calculates the optimal memory and concurrency for your queues.
- Manual WLM
  - You control the memory and concurrency for your queues.

Concurrency Scaling

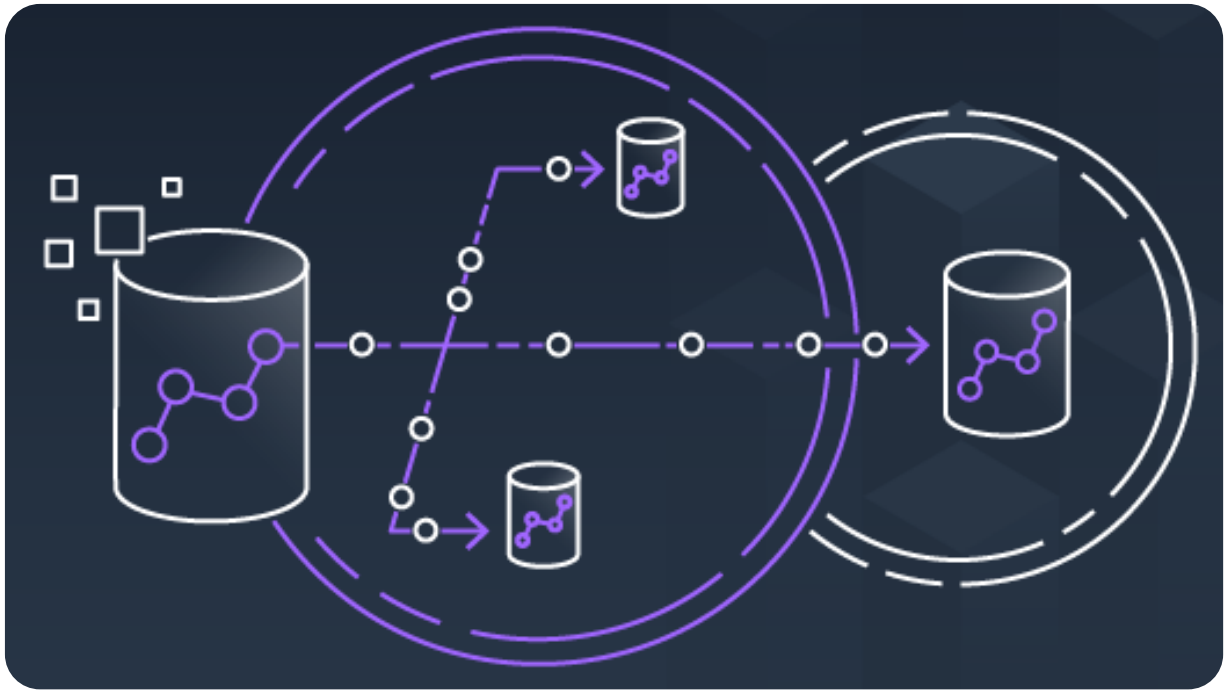
- With the Concurrency Scaling feature, you can support thousands of concurrent users and concurrent queries, with consistently fast query performance.
- When you turn on concurrency scaling, Amazon Redshift automatically adds additional cluster capacity to process an increase in both read (e.g. dashboard queries) and write (e.g. data ingestion and processing) queries.
- To turn on concurrency scaling for a queue, set the Concurrency Scaling mode value to auto.
- When the number of queries routed to a concurrency scaling queue exceeds the queue's configured concurrency, eligible queries are sent to the concurrency scaling cluster.
- You can control the *max_concurrency_scaling_clusters* through parameter groups.



Redshift Data Sharing

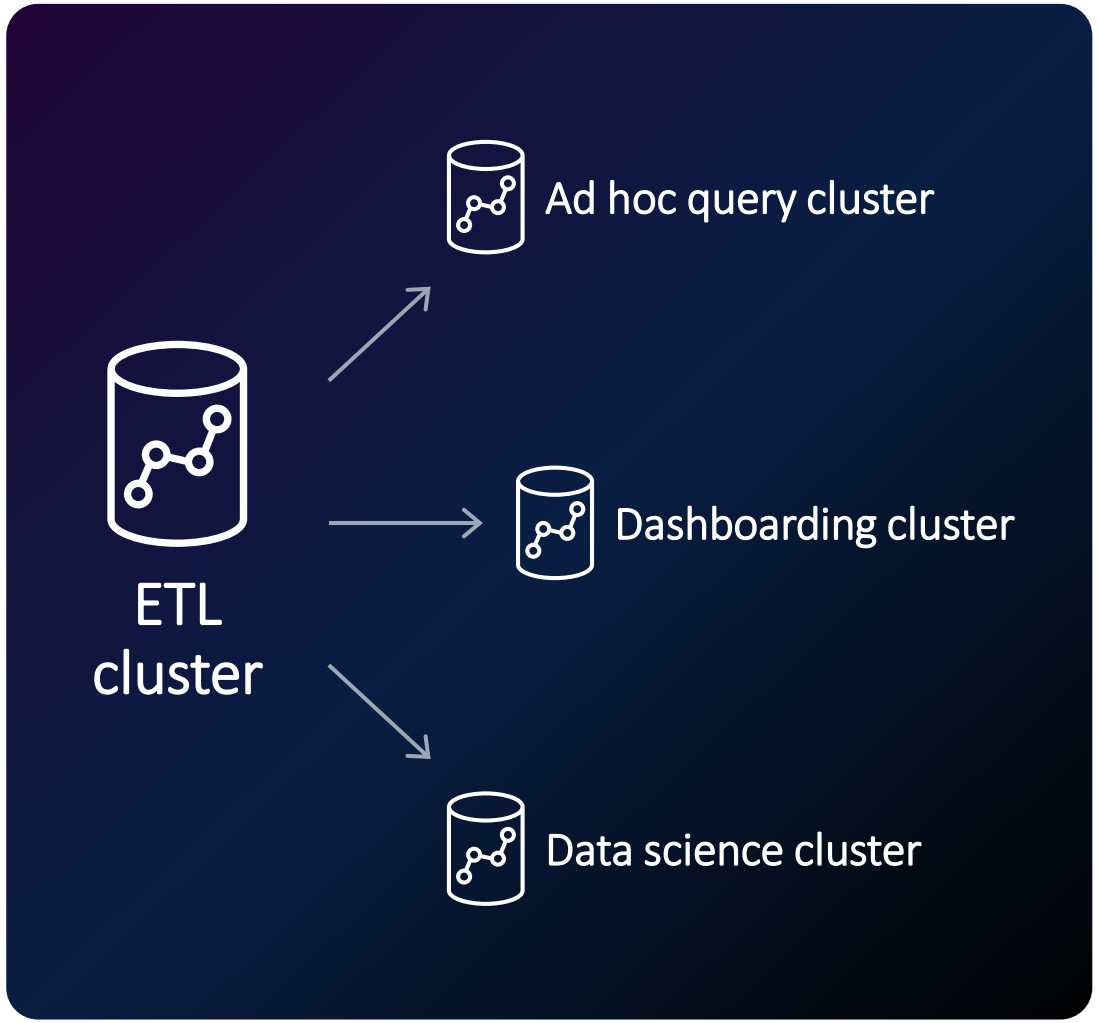
Redshift data sharing

- Redshift data sharing enables users to share live data across Redshift clusters without the need for data movement or duplication.
- This feature allows multiple clusters to access the same data sets, making it easier to collaborate and derive insights from shared data.
- A cluster can be a Producer and Consumer at the same time.
- On Producer
  - Create a datashare on the cluster
  - Set permissions for consumer
- On Consumer
  - Create a database from the datashare
- You can share data:
  - Across accounts
  - Across regions

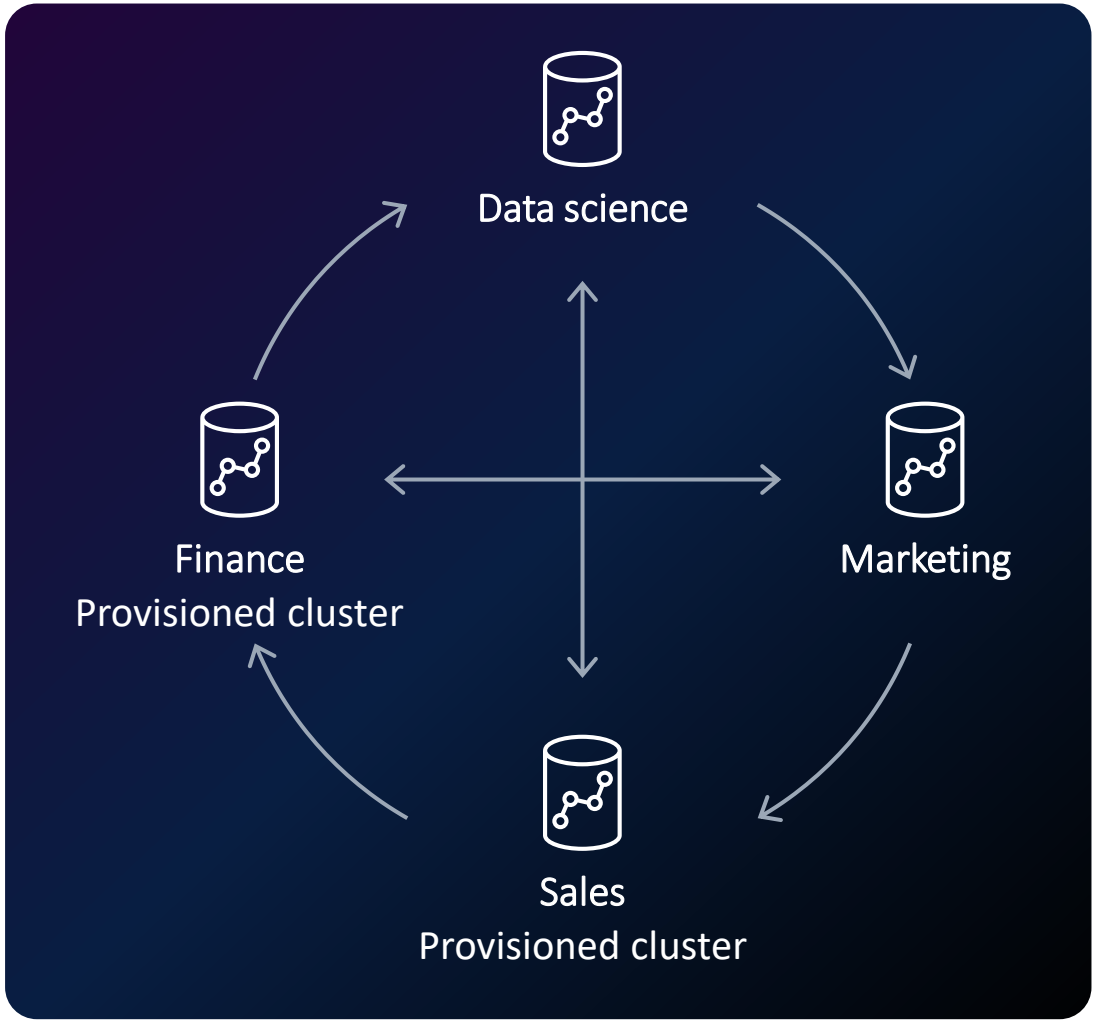


Multi-cluster reference architectures

Hub-and-spoke architecture



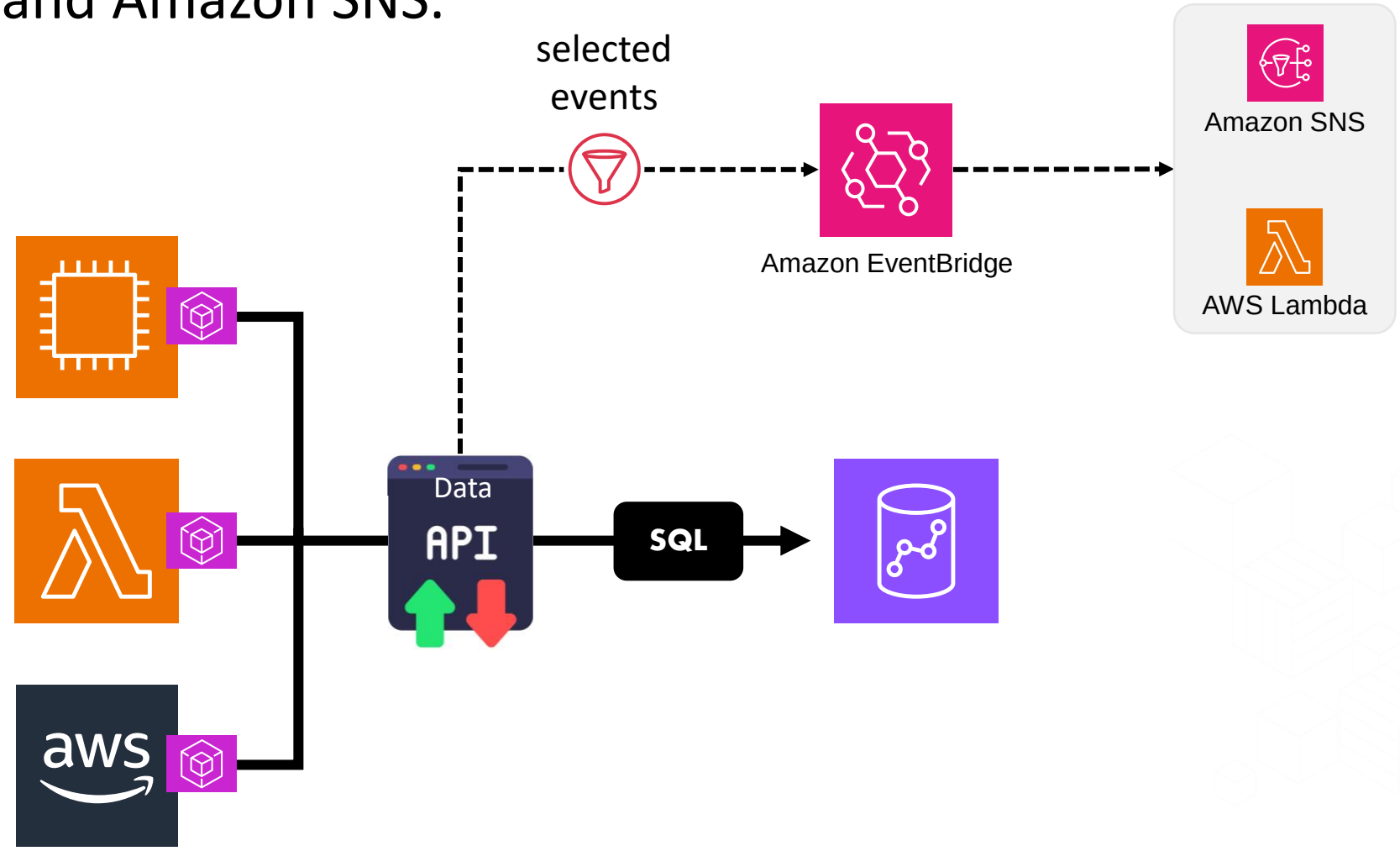
Data mesh architecture



Additional Information

Integration with Amazon EventBridge

- You can monitor Data API events in EventBridge and can create rules that match selected events and route them to targets to take action such as AWS Lambda and Amazon SNS.



Stored Procedure

- A stored procedure is a user-created object to perform a set of SQL queries and logical operations. The procedure is stored in the database and is available to users who have sufficient privileges to run it.
- Stored procedures are commonly used to encapsulate logic for data transformation, data validation, and business-specific logic. By combining multiple SQL steps into a stored procedure, you can create reusable code blocks that can run together as a single transaction or multiple individual transactions.
- You can use the PL/pgSQL procedural language, including looping and conditional expressions, to control logical flow.
- Using Query Editor v2.0 you can schedule running of stored procedures.

Troubleshooting performance issues

- Amazon Redshift has many system tables and views that contain information about how the system is functioning.
- You can query these system tables and views the same way that you would query any other database tables.
- There are several types of system tables and views:
  - SYS views are used to monitor query and workload usage for provisioned clusters and serverless workgroups.
  - STL views are generated from logs that have been persisted to disk to provide a history of the system.
- STL_ALERT_EVENT_LOG
  - Records an alert when the query optimizer identifies conditions that might indicate performance issues. Use details in it to identify opportunities to improve query performance.

VACCUM

- Redshift Vacuum is a maintenance process in Amazon Redshift that helps in reclaiming space and sorting data in tables. It plays a crucial role in improving query performance and reducing storage costs.
- VACCUM command re-sorts rows and reclaims space in either a specified table or all tables in the current database.
- *VACCUM FULL <TableName>*
  - Sorts the specified table (or all tables in the current database) and reclaims disk space occupied by rows that were marked for deletion by previous UPDATE and DELETE operations. VACUUM FULL is the default.
  - A full vacuum doesn't perform a reindex for interleaved tables.
- *VACCUM REINDEX <TableName>*
  - Analyzes and reindex the distribution of the values in interleaved sort key columns, then performs a full VACUUM operation.

Redshift Copy Command

- The COPY command loads data in parallel from Amazon S3, Amazon EMR, Amazon DynamoDB, or multiple data sources on remote hosts. COPY loads large amounts of data much more efficiently than using INSERT statements, and stores the data more effectively as well.

- You can use a manifest file to ensure that your COPY command loads all of the required files, and only the required files, from Amazon S3. You can also use a manifest when you need to load multiple files from different buckets or files that don't share the same prefix.

```
COPY customer
FROM 's3://mybucket/cust.manifest'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
MANIFEST;
```