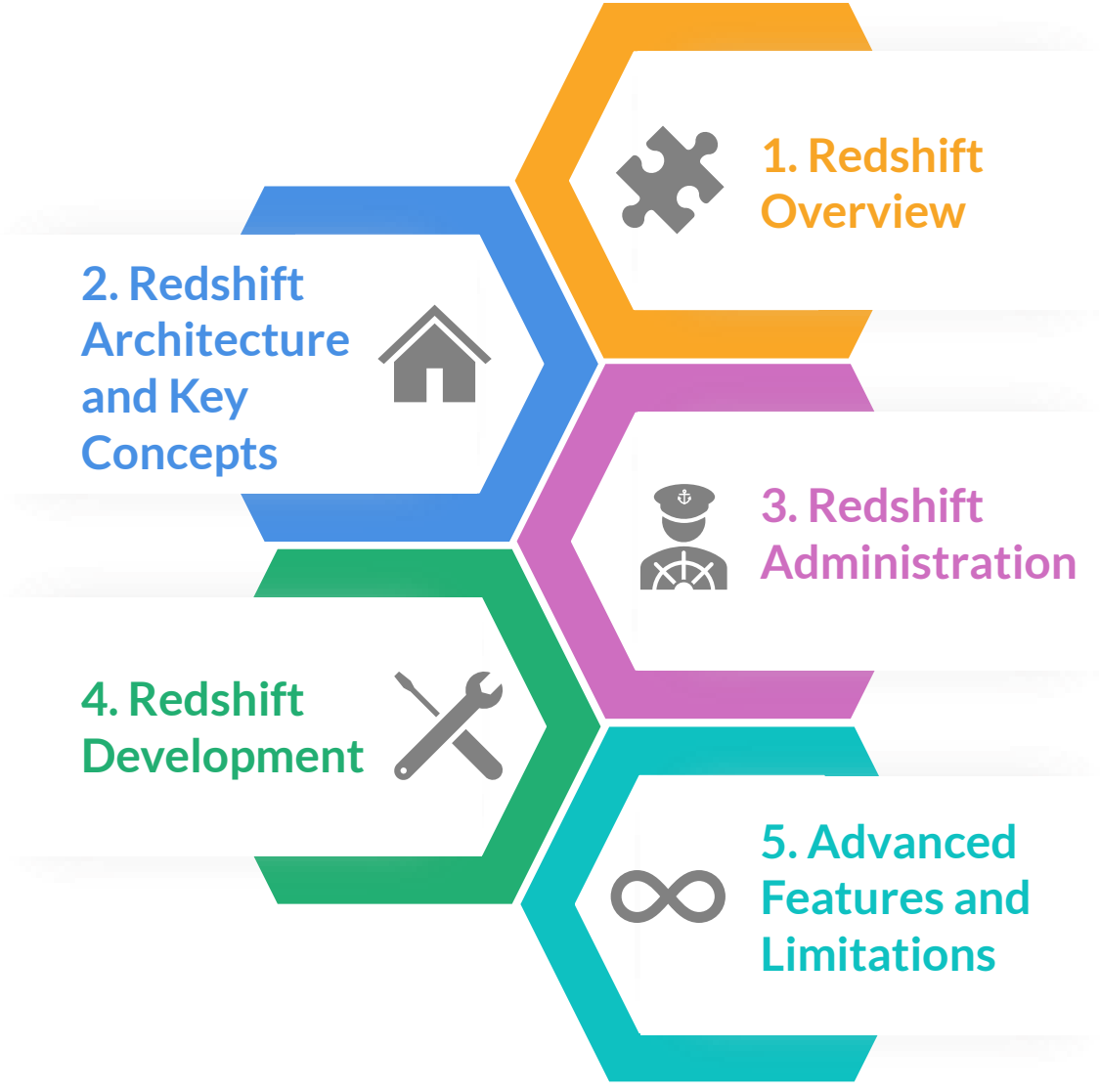
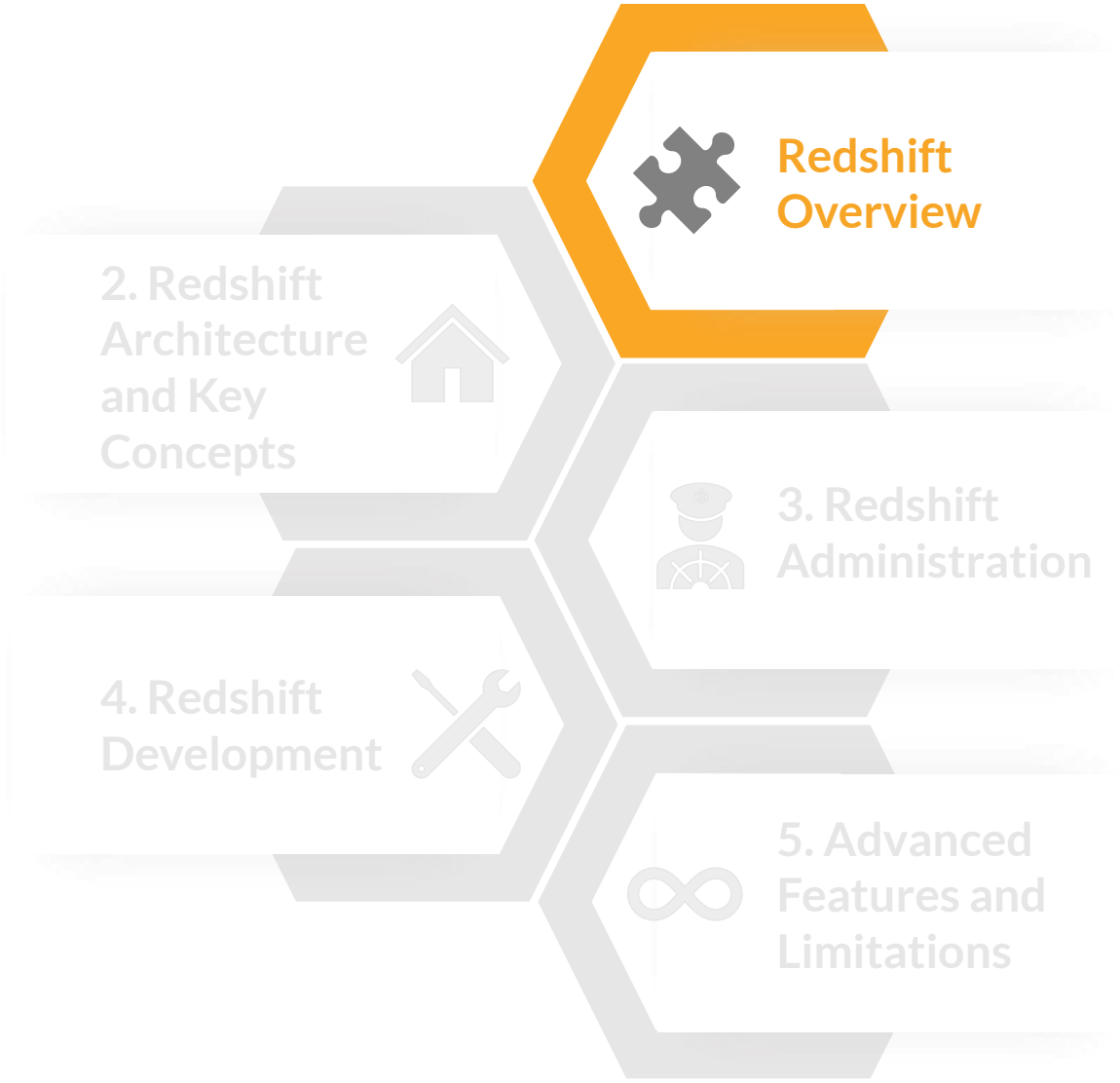

1. Redshift Overview
2. Redshift Architecture and Key Concepts
3. Redshift Administration
4. Redshift Development
5. Advanced Features and Limitations

Session 1: Redshift Overview

Upon completion of this session, you will be able to answer the following questions:

- What are the limitations of traditional data warehouses?
- Why industries choose Redshift as their data warehousing solution?
- What are some of the use-cases of Amazon Redshift?

We will also look at a case study of an industry using Amazon Redshift at the center of their data and analytics pipeline



ABOUT DATA – QUICK RECAP

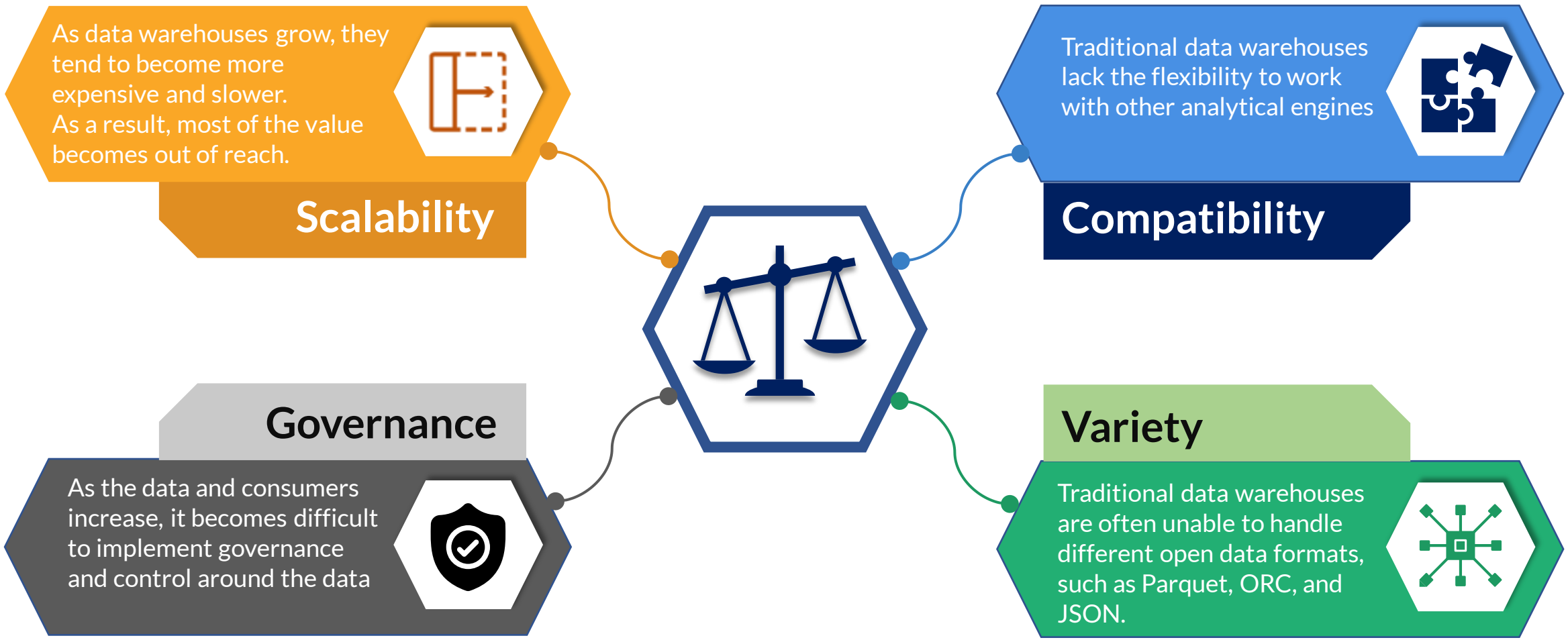
Data helps identify and drive insights, which helps in decision-making

01 What is data?

02 Why do we need to store data?

03 Where do we store data?

LIMITATIONS OF TRADITIONAL DATA WAREHOUSES



Why Redshift?

Performance

- Up to 3x faster than other cloud data warehouses
- Can handle virtually unlimited connections with Concurrency Scaling

Integration Capabilities

- Easily integrates with data on lake and other RDBMS (PostgreSQL)
- Native connectors available for most popular BI tools in the market

Scalability

- Virtually unlimited scaling capacity
- Scales up and down within minutes through elastic resize

Cost Effective

- Pay for what you use and not for what you provision
- Reservation of nodes helps save up to 60% cost over on-demand pricing

Manageability

- Fully managed, cloud Platform-as-a-Service offering
- Faulty nodes are replaced automatically within minutes

Rapid Innovation

- AWS listens to customer feedback and releases new features to Redshift at a very fast pace

Security

- AWS grade security; VPC, Enhanced Networking, At-rest, and in-transit encryption through KMS and SSL
- Integrates to Lake Formation for fine control on database access

SQL Compatibility

- Redshift uses industry standard ANSI standard SQL with added functionality for data analytics, compared to other DW such as Hive which uses HiveQL and converts it to MR jobs before execution

INDUSTRIAL USE-CASES FOR AMAZON REDSHIFT

Industry	Use Cases	Example
Healthcare	• Analyze clinical records to improve patient outcomes and predict diseases for preventive programs	Pfizer, AstraZeneca
Financial Services	• Analyse trading and market data, risk analyses, fraud detection	Nasdaq, London Stock Exchange
Food and Manufacturing	• Create personalised experiences and offers for customers	Hilton Hotels & Resorts, McDonald's
Gaming	• Aggregate data from games and players and analyse in-game behaviour	Warner Bros. Interactive Entertainment
Telecommunications & Media	• Store, process and analyse call data records for consumer billing • Analyse consumer behaviour for personalized recommendations	NTT docomo, 21st Century Fox
Advertising	• Analyse clickstream and ad impression logs to improve ad targeting	nielsen

CASE STUDY: Nasdaq

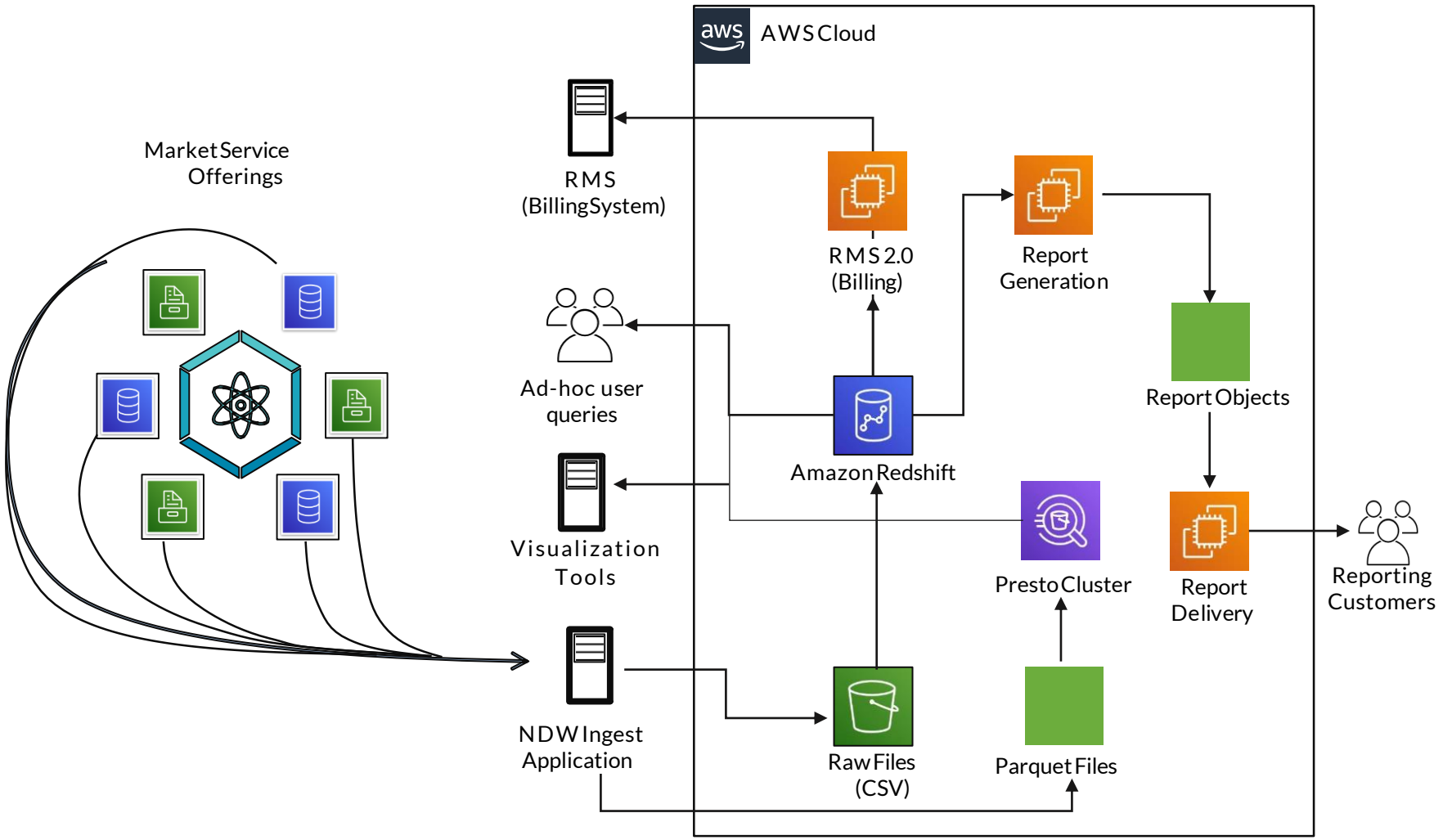
About Nasdaq

- Lists more than 3,500 companies in 35 countries, representing more than **$8.8tn** in total market value
- Leading index provider with **41,000+** indexes across asset classes and geographies
- Over **10,000 corporate clients in 60 countries**
- **100+ Data product offerings** supporting $2.5+mn investment professionals and users IN 98 countries
- Nasdaq technology powers over 70 MARKETPLACES, regulators, CSDs and clearing-houses in over 50 COUNTRIES

Technology Challenges and Key Drivers for Change (2013)

- Expensive on-premise data warehousing and billing systems (RMS), costing ~$1.16mn annually
- Limited storage and compute capacity
- Ever increasing data from orders, trades, and quotes (~4-6 bn rows inserted per trading day)
- Difficult to manage and maintain the hardware assets
- Service-level agreements required overnight for batch processes to complete prior to the next morning

CASE STUDY: Nasdaq



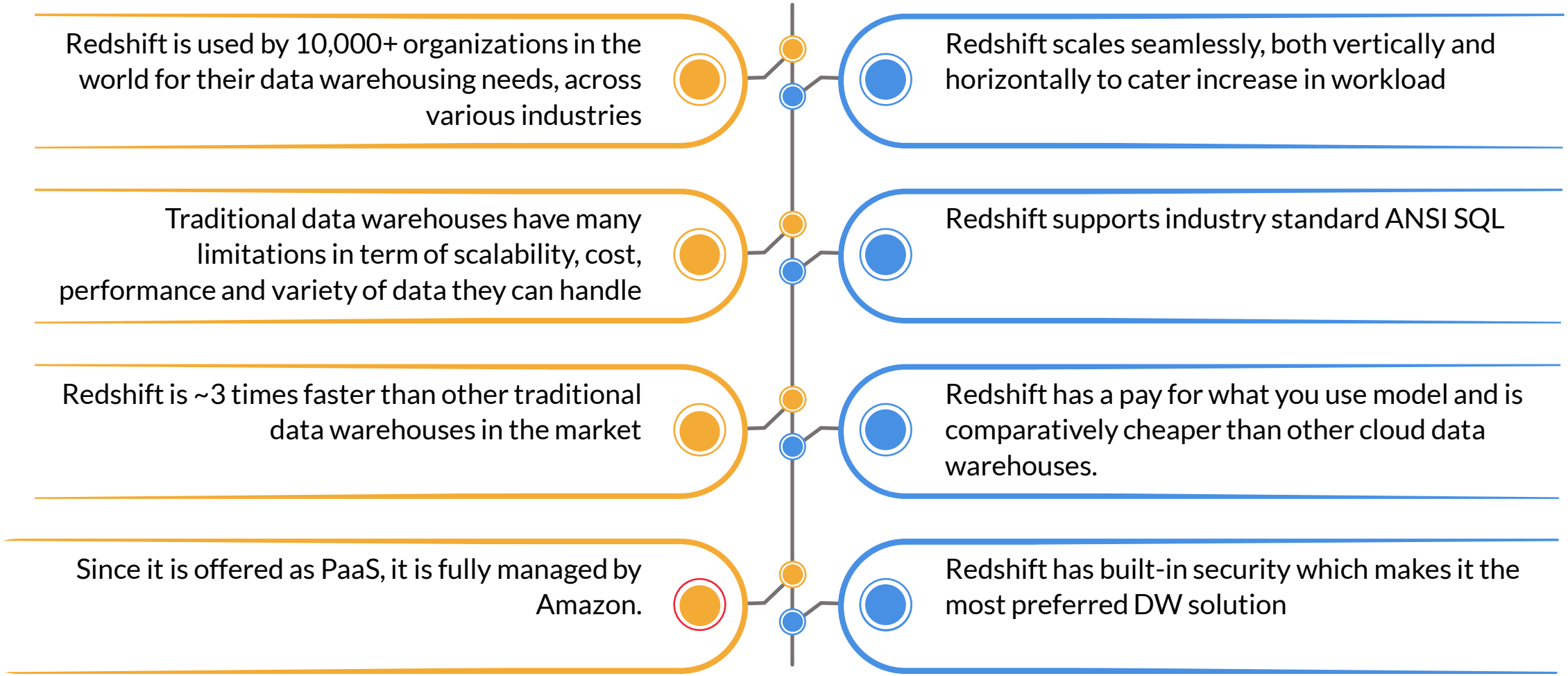
Nasdaq Data Warehouse 1.0 – 2013

Migrated on-premise DW to Redshift

Impact

- Costs reduced to 43% of the on-prem budget for same data set
- Handled increasing data (~14bn rows/day) at tremendous write speed (~2.76mn rows/second)
- Tuned queries running faster than the legacy on-premise data warehouse system
- Able to scale seamlessly to meet future needs

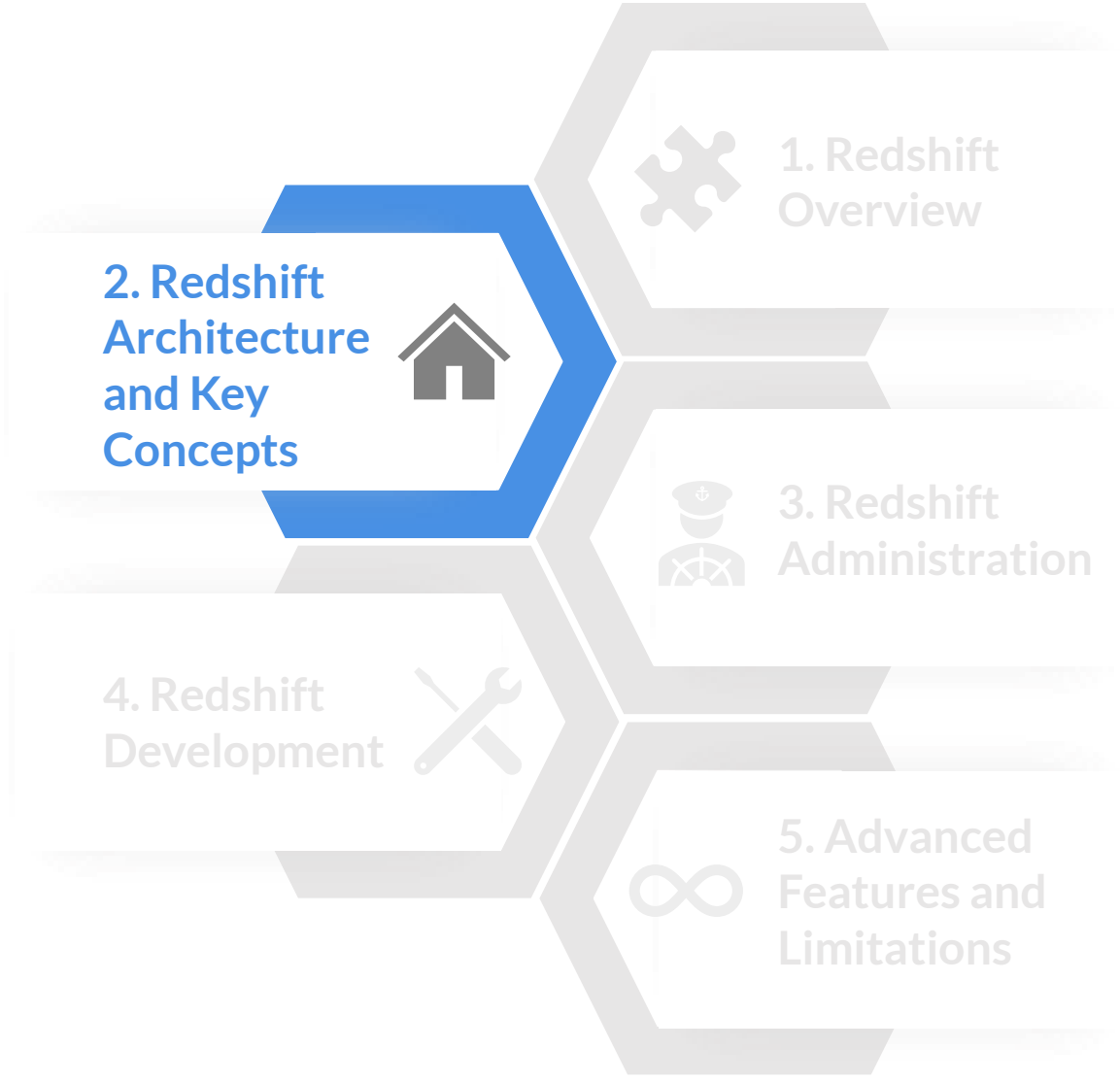
SESSION SUMMARY



Session 2: Redshift Architecture and Key Concepts

Upon completion of this session, you will be able to answer the following questions:

- What is Amazon Redshift?
- How does the architecture of Amazon Redshift look like?
- What is MPP?
- What are Sort Keys and Distribution Keys?
- What is Compression Encodings and why are these useful?
- What is Concurrency Scaling?

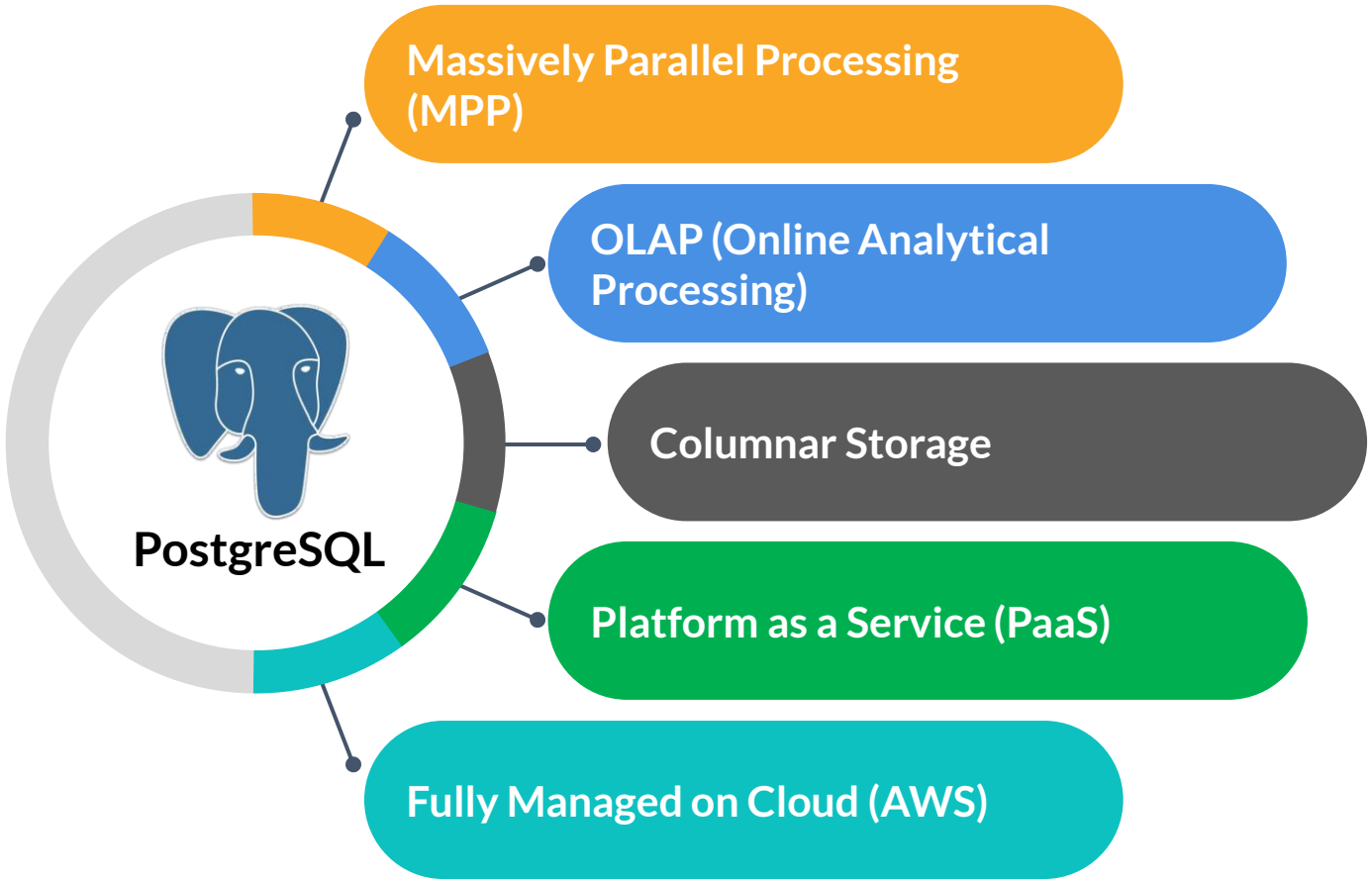


WHAT IS AMAZON REDSHIFT?

AMBASSADOR

AMBIEROD

WHAT IS AMAZON REDSHIFT?



AMAZON REDSHIFT

Amazon Redshift is a fast, scalable data warehouse that makes it simple and cost-effective to analyze all your data across your data warehouse and data lake. Amazon Redshift delivers 10 times faster performance than other data warehouses by using machine learning, massively parallel query execution, and columnar storage on a high-performance disk.

REDSHIFT ARCHITECTURE

Shared nothing, Massively parallel architecture

Leader node

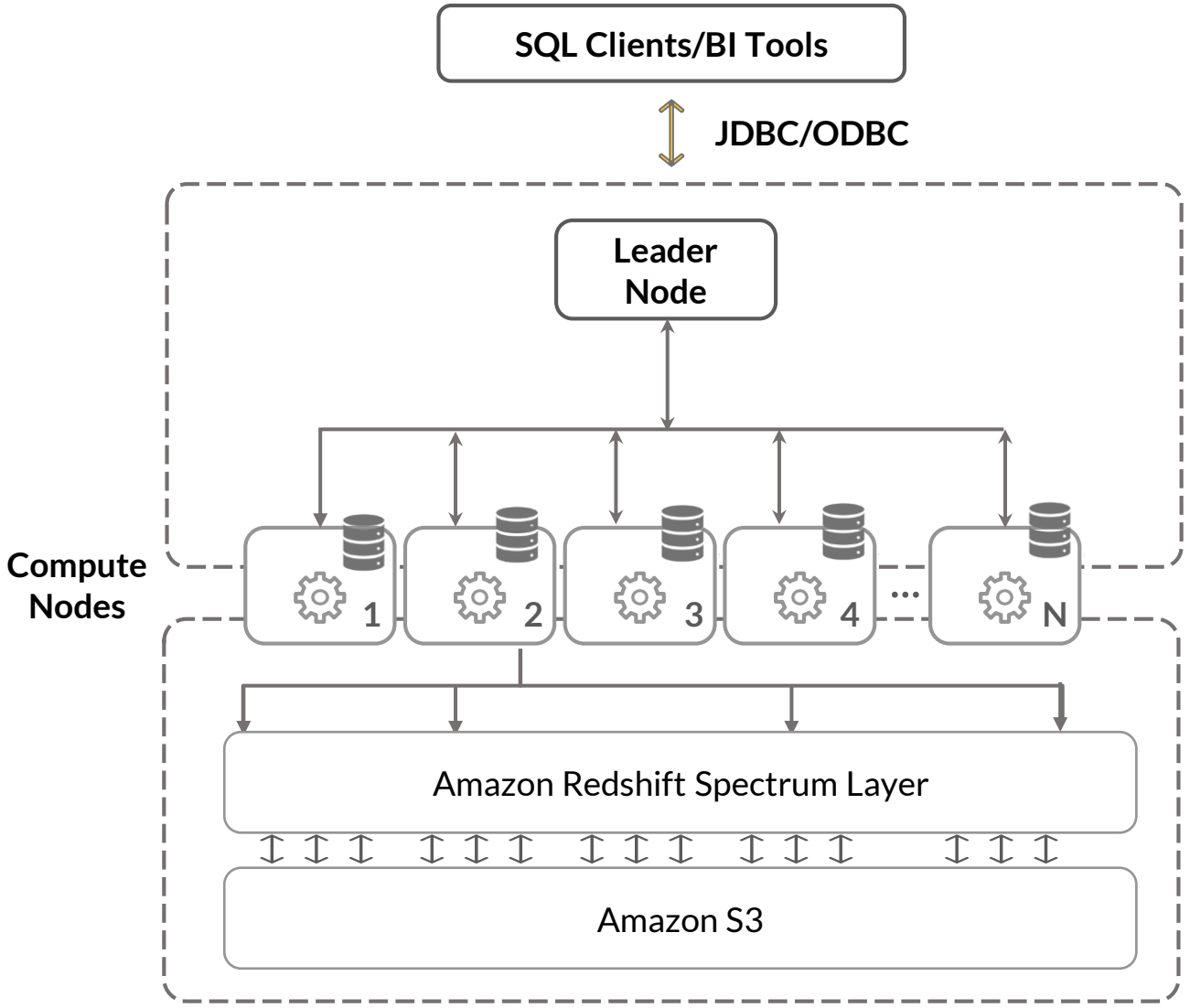
- SQL endpoint
- Stores metadata
- Coordinates parallel SQL processing

Compute nodes

- Local, columnar storage
- Executes queries in parallel
- Load, unload, backup and restore

Amazon Redshift Spectrum nodes

- Execute queries directly against
- Amazon Simple Storage Service (Amazon S3)



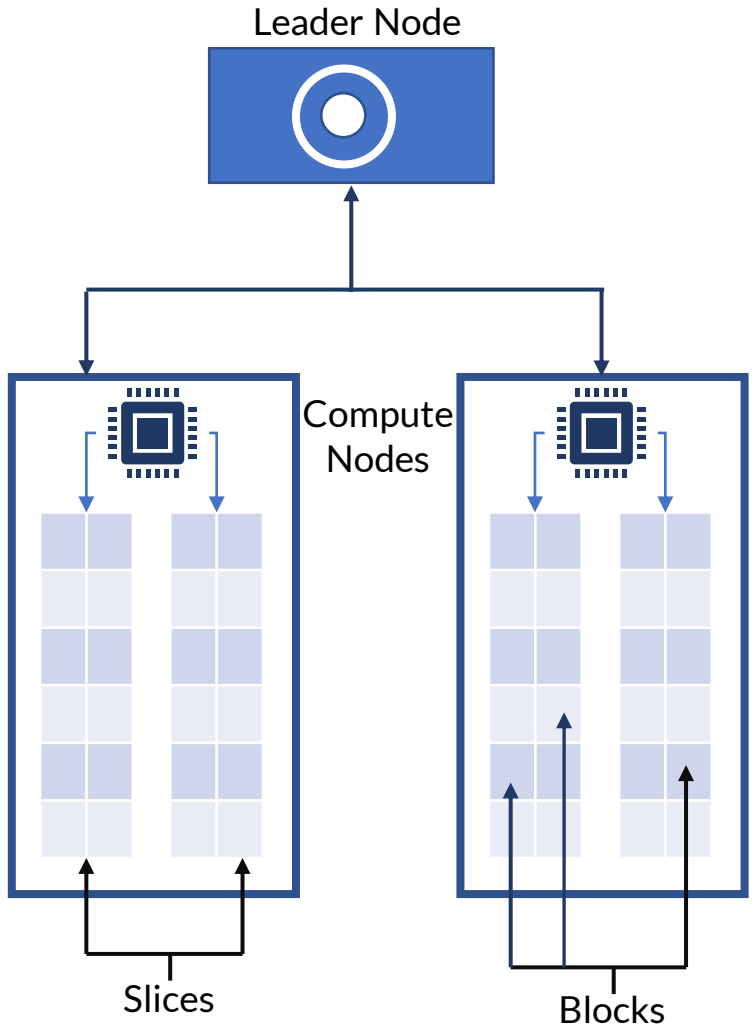
SLICES AND BLOCKS

Slices

- Each compute node is partitioned into 2 or 16 slices, which is determined by the node type
- A slice can be thought of like a virtual compute node (within a compute node); each slice is allocated a portion of the node's memory and disk space.
- Table rows (the actual data) are distributed to slices

Blocks

- Slices are further divided into blocks (1 MB each)
- A full block can contain millions of values
- Each block can be encoded/compressed with one of the 13 encodings available



COLUMNAR ARCHITECTURE

Amazon Redshift uses a columnar architecture for storing data on disk

Goals

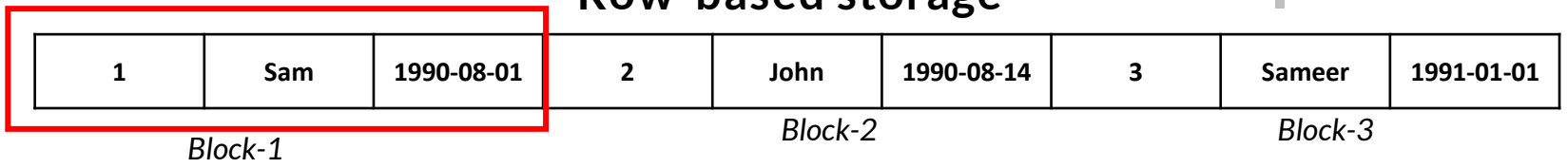
- Reduce I/O for analytics queries
- Only read the required column data

```
CREATE TABLE rs_tbl (
id          INT
,name       CHAR(30)
,dob        DATE
);
```

SELECT min(dob) FROM rs_tbl;

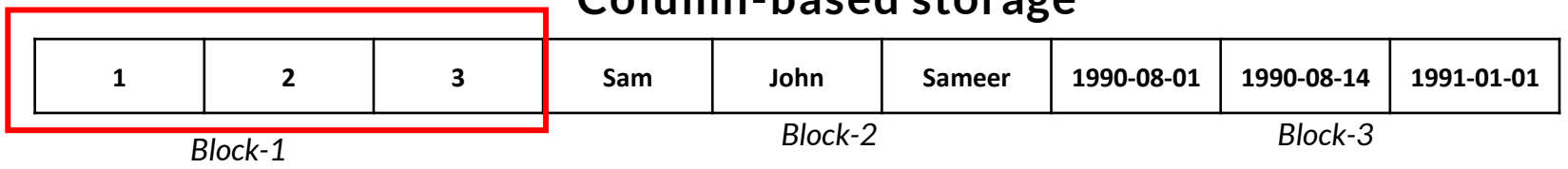
id	loc	dob
1	Sam	1990-08-01
2	John	1990-08-14
3	Sameer	1991-01-01

Row-based storage



.. Need to read everything unnecessary I/O

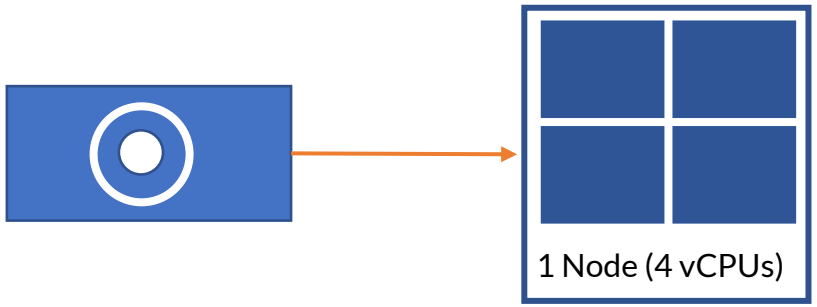
Column-based storage



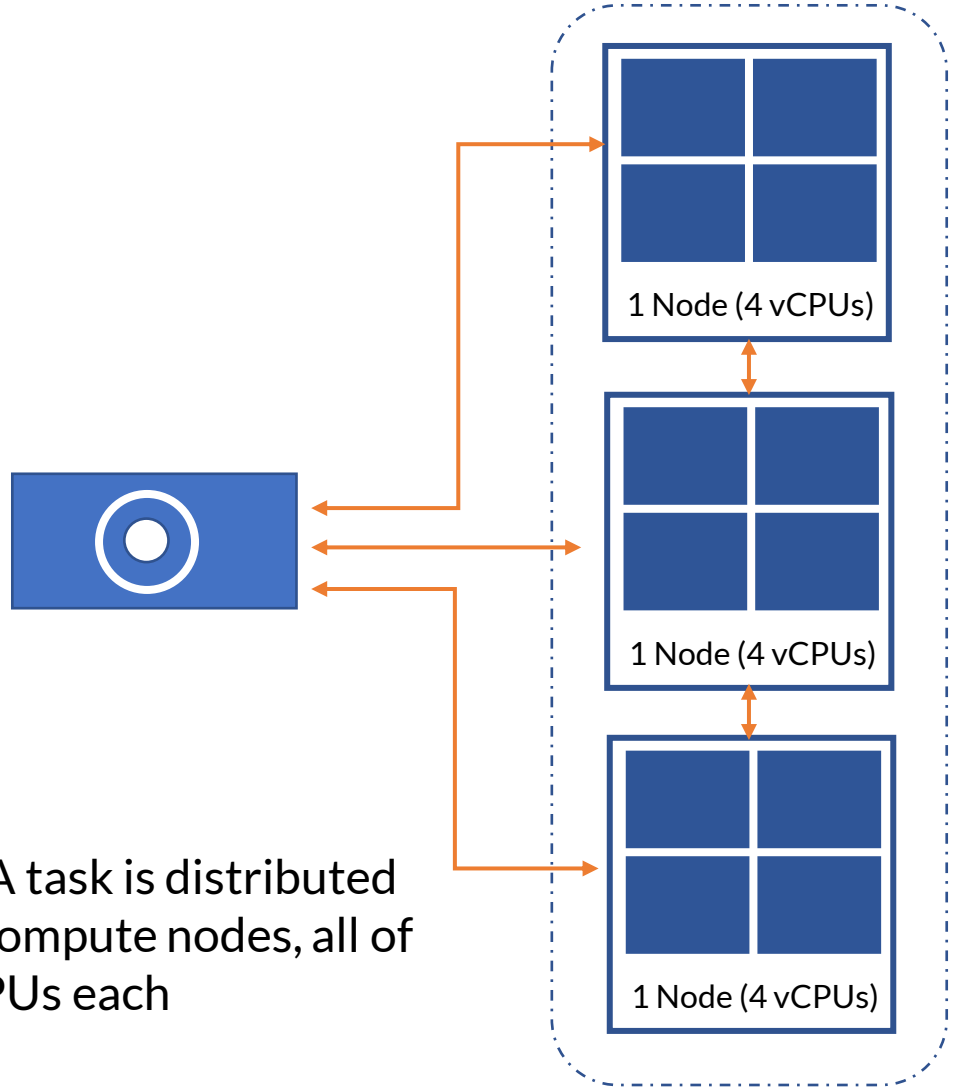
.. Scan only relevant blocks

MASSIVELY PARALLEL PROCESSING (MPP)

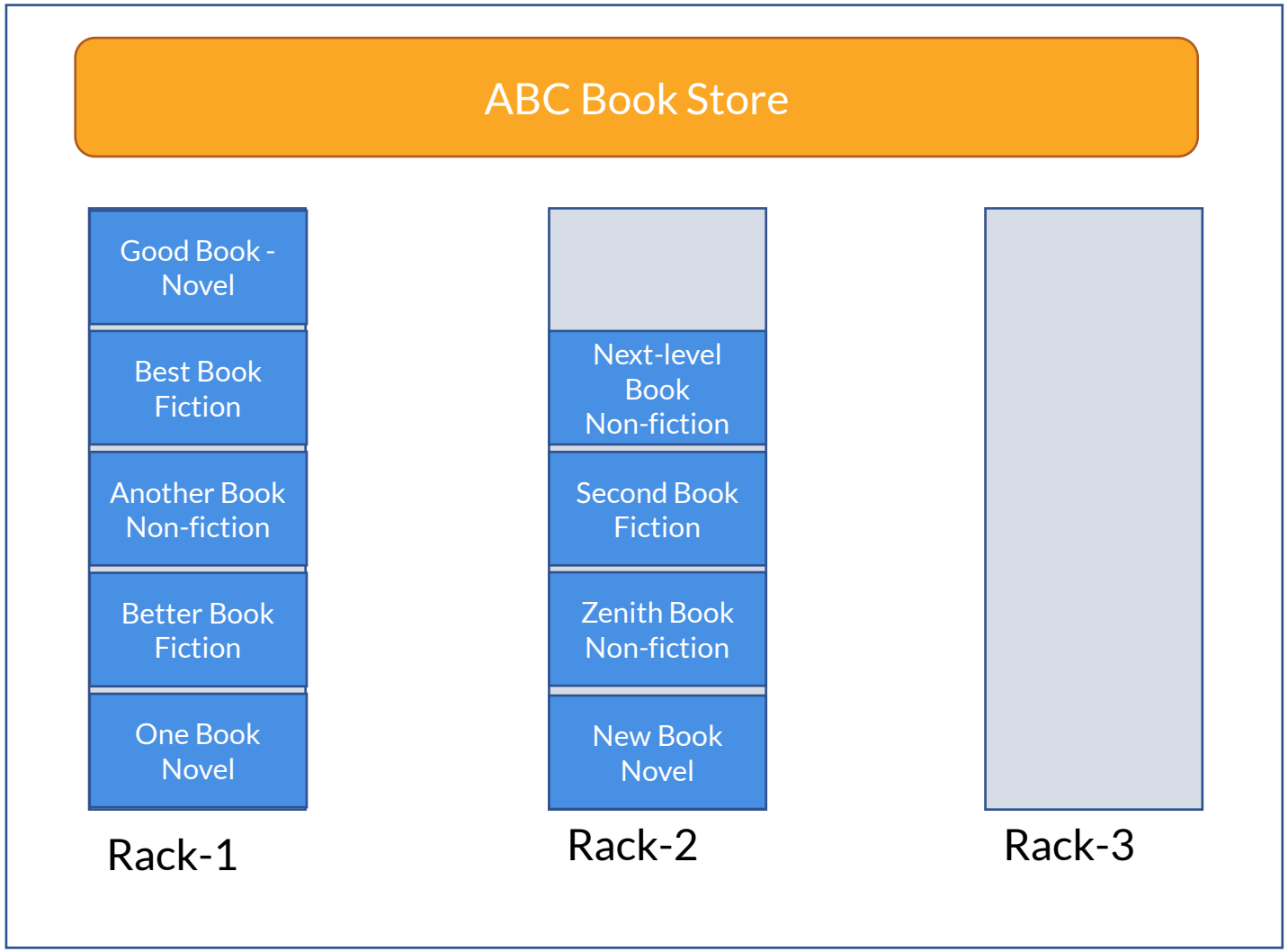
Single Node: A task can only be executed on one node, which has four CPUs.



Multiple Nodes: A task is distributed among multiple compute nodes, all of which have 4 vCPUs each



SORT KEY AND DISTRIBUTION KEYS - EXAMPLE



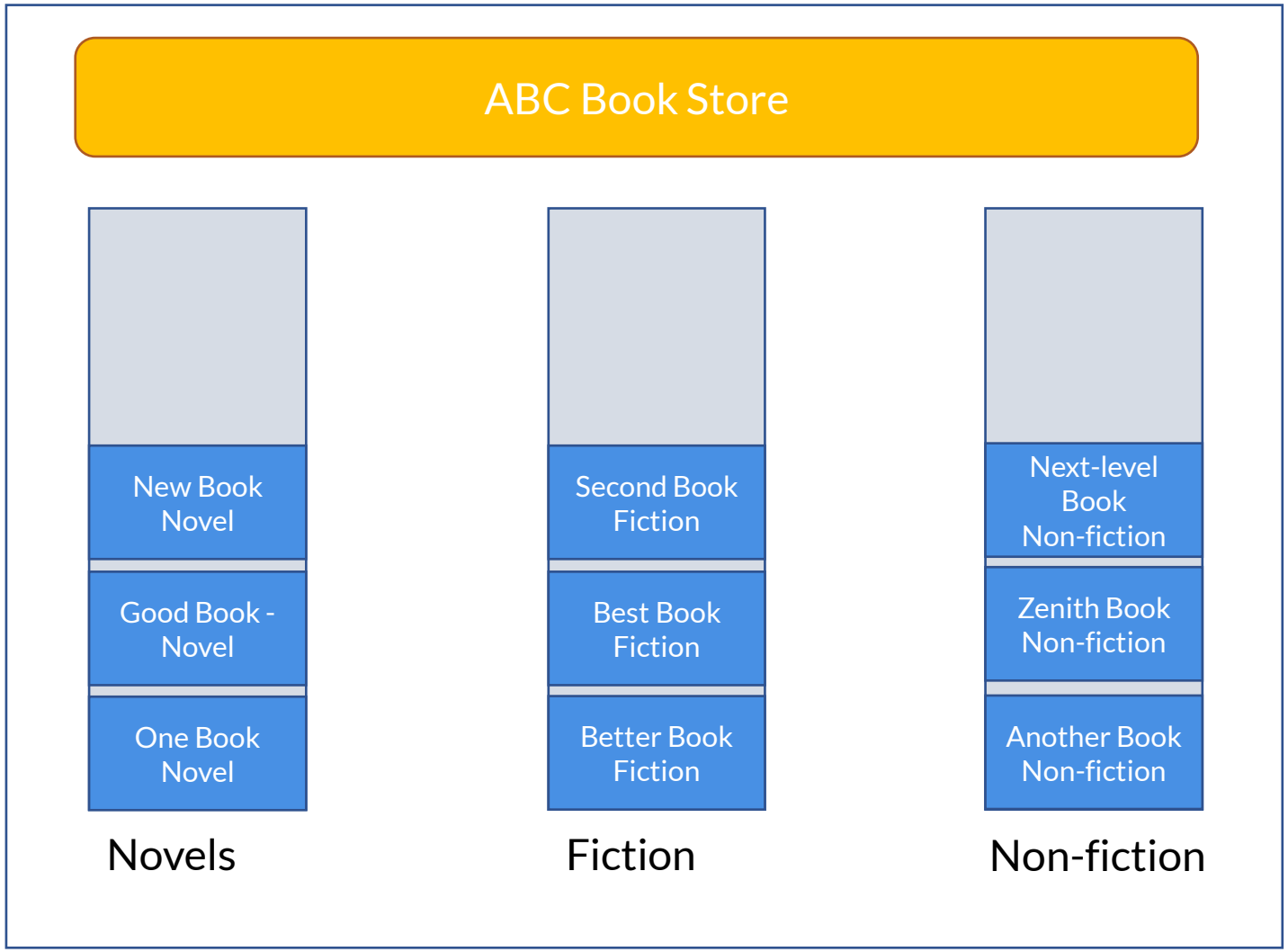
ABC Book Store sells a variety of Books such as novels, fiction and non-fiction. Today, they got the delivery of books they ordered from a wholesaler and now they need to arrange the books on the racks

Scenario 1: First Book goes on rack-1, followed by second Book, and so on…Once rack-1 is full, only then they move to rack-2

Problem Statement:
How difficult will it be to find a particular Book, say Zenith Book, even if you know the genre?

HARD

SORT KEY AND DISTRIBUTION KEYS - EXAMPLE



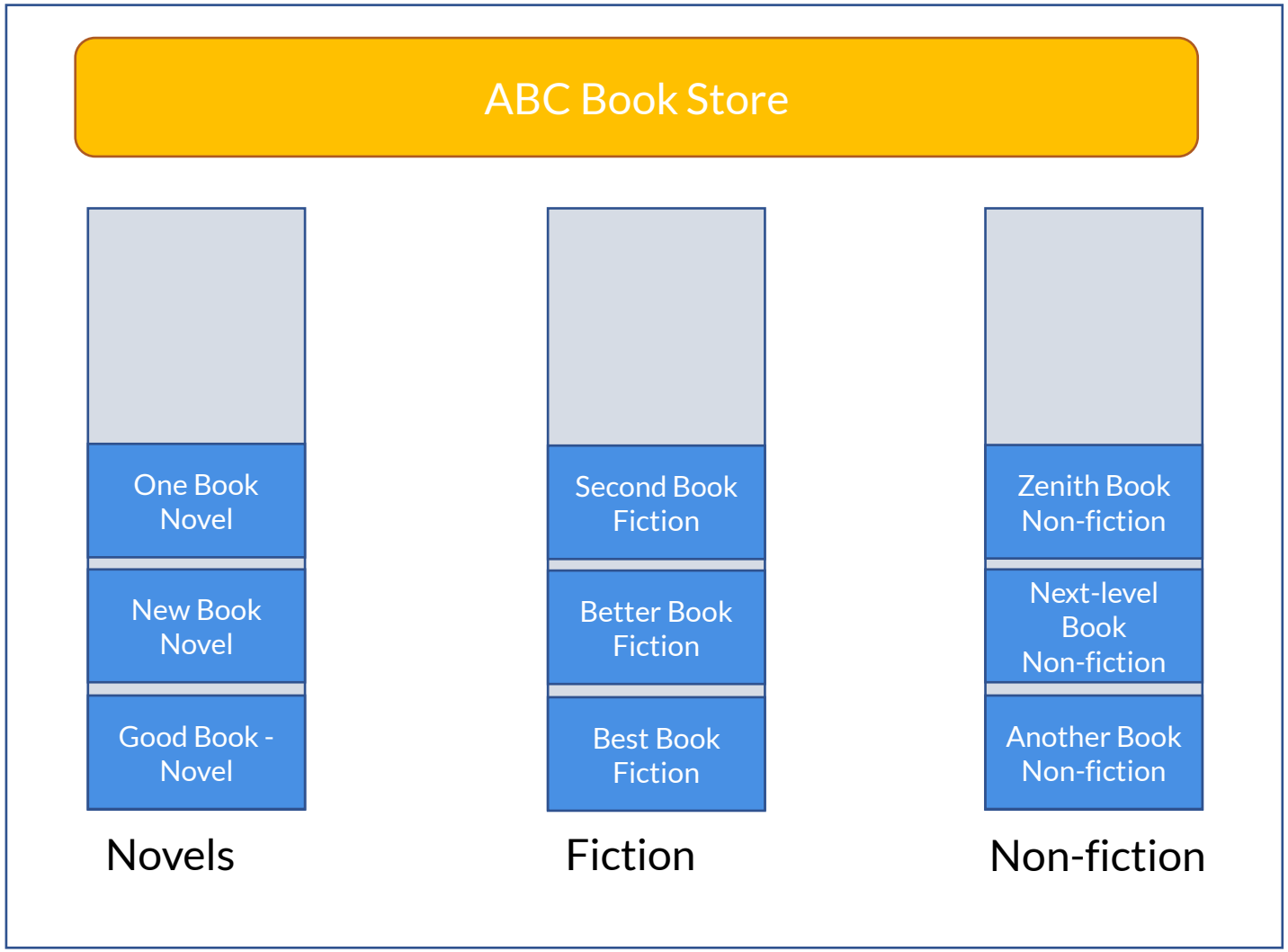
ABC Book Store sells a variety of Books such as novels, fiction and non-fiction. Today, they got the delivery of books they ordered from a wholesaler and now they need to arrange the books on the racks

Scenario 2: They name the shelves according to the Book 'genres' and arrange each Book in the respective genre rack

Problem Statement:
How difficult will it be to find a particular Book, say Zenith Book, even if you know the genre?

EASIER

SORT KEY AND DISTRIBUTION KEYS - EXAMPLE



ABC Book Store sells a variety of Books such as novels, fiction and non-fiction. Today, they got the delivery of books they ordered from a wholesaler and now they need to arrange the books on the racks

Scenario 3: Along with arranging the Book in different racks based on their 'genre', They arrange the Books in alphabetical order, according to their 'Title' within the racks.

Problem Statement:
How difficult will it be to find a particular Book, say Zenith Book, even if you know the genre?

EASIEST

SORT KEYS AND ZONE MAPS

Goal

- Make queries run faster by increasing the effectiveness of zone maps and reducing I/O

Impact

- Enables range-restrictedscans to prune blocks by leveraging zone maps

Zone Maps:

- In-memory blockmetadata
- Track the minimum and maximum value for each block.
- Effectively prunes blocks that do not contain data for a given query

SELECT * FROM RS_TABLE WHERE DATE = '09-JUNE-2020'

TYPES OF SORT KEYS

Single Column

SORTKEY (Date)

Date	Region	Country
2-JUNE-2020	OCEANIA	NEW ZEALAND
2-JUNE-2020	ASIA	SINGAPORE
2-JUNE-2020	AFRICA	ZAMBIA
3-JUNE-2020	ASIA	HONG KONG
3-JUNE-2020	EUROPE	GERMANY
3-JUNE-2020	ASIA	KOREA

- Best for queries that use the 1st column as primary filter
- Can speed up joins and group by statements

Compound

SORTKEY COMPOUND (Date, Region)

Date	Region	Country
2-JUNE-2020	AFRICA	ZAMBIA
2-JUNE-2020	ASIA	SINGAPORE
2-JUNE-2020	OCEANIA	NEW ZEALAND
3-JUNE-2020	ASIA	HONG KONG
3-JUNE-2020	ASIA	KOREA
3-JUNE-2020	EUROPE	GERMANY

- Table sorted by order as given in sort key
- Best for queries that use the 1st column as primary filter, then others

Interleaved

SORTKEY INTERLEAVED (Date, Region, Country)

Date	Region	Country
2-JUNE-2020	OCEANIA	NEW ZEALAND
2-JUNE-2020	ASIA	SINGAPORE
2-JUNE-2020	AFRICA	ZAMBIA
3-JUNE-2020	ASIA	HONG KONG
3-JUNE-2020	EUROPE	GERMANY
3-JUNE-2020	ASIA	KOREA

- Equal weight given to each column
- Best for queries that use different filter columns

DATA DISTRIBUTION

Distribution style defines how the data will be stored on the compute nodes.

KEY: Value is hashed, same value goes to same location (slice)

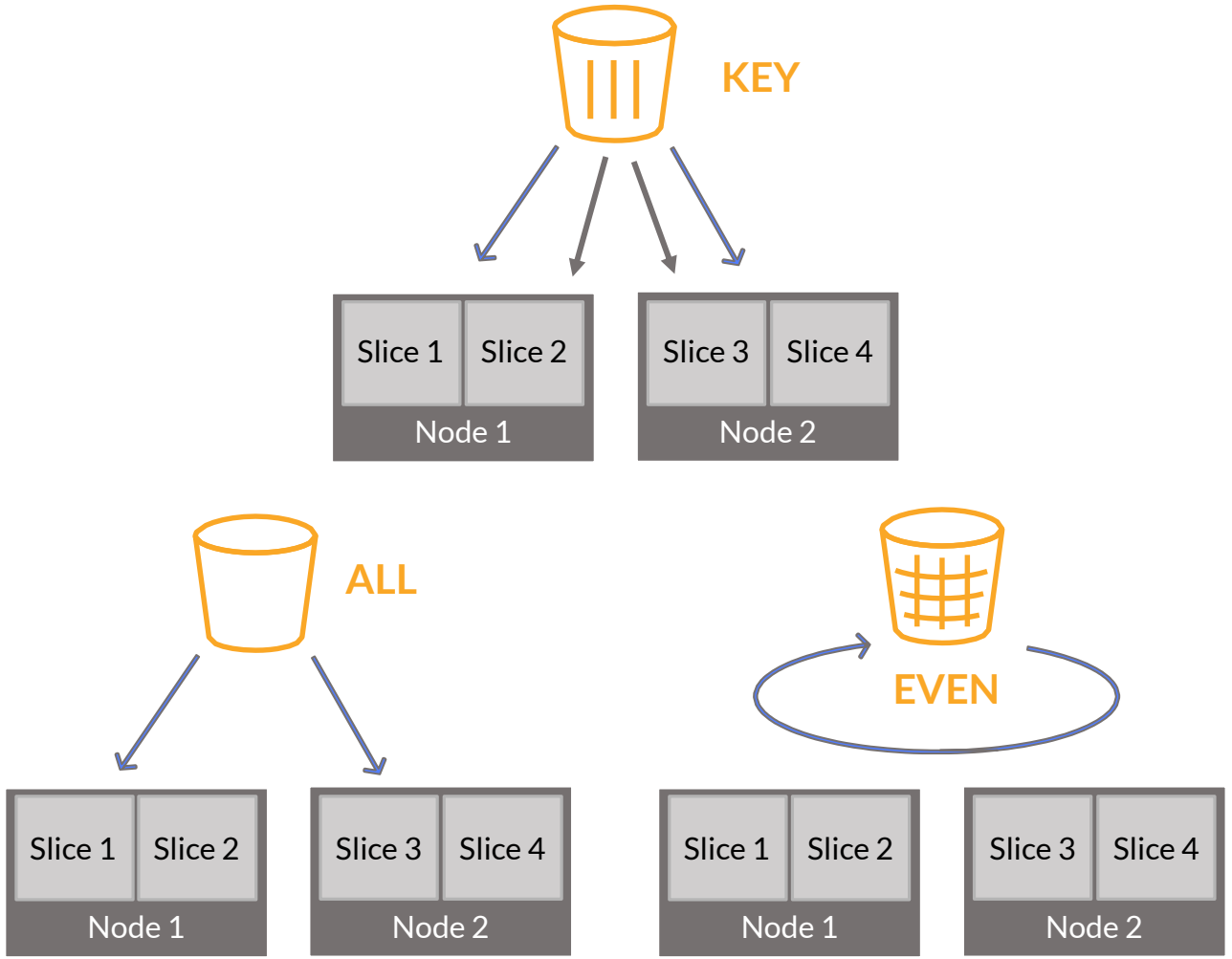
ALL: Full table data goes to the first slice of every node

EVEN: Round robin

AUTO: ALL, followed by EVEN

Goals:

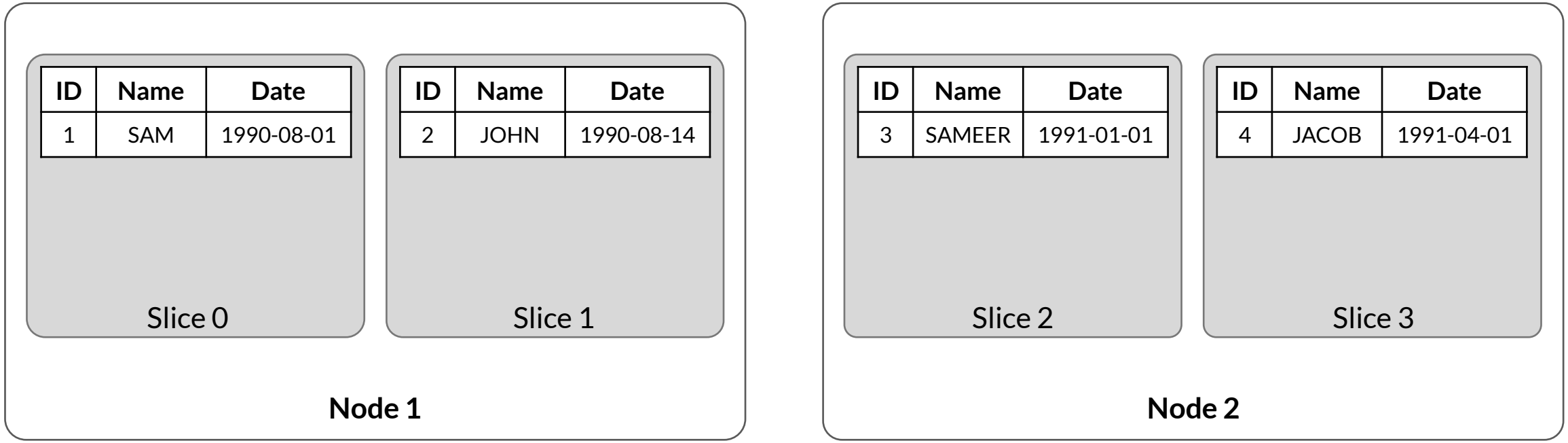
- Distribute data evenly for parallel processing
- Minimise data movement during query processing



DISTRIBUTION KEYS - EXAMPLE

```
CREATE TABLE RS_DEMO (
       id     INT,
       name   VARCHAR(20),
       dob    DATE
) DISTSTYLE EVEN;
```

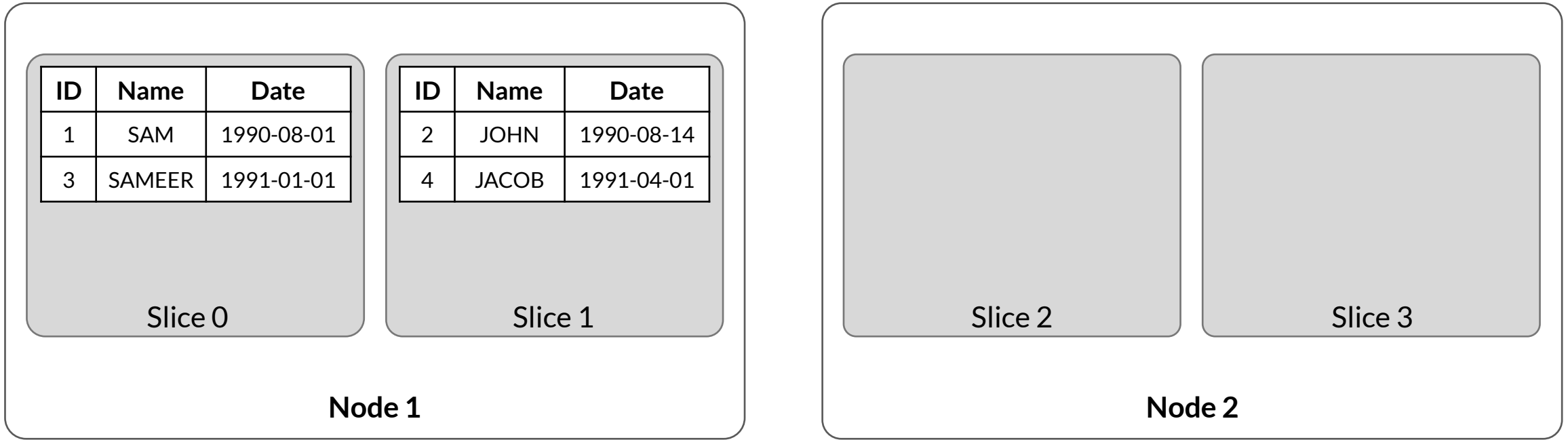
```
INSERT INTO RS_DEMO VALUES
(1, 'SAM', '1990-08-01'),
(2, 'JOHN', '1990-08-14'),
(3, 'SAMEER, '1991-01-01'),
(4, 'JACOB, '1991-04-01');
```



DISTRIBUTION KEYS - EXAMPLE

```
CREATE TABLE RS_DEMO (
       id     INT,
       name   VARCHAR(20),
       dob    DATE
) DISTSTYLE KEY DISTKEY (name);
```

```
INSERT INTO RS_DEMO VALUES
(1, 'SAM', '1990-08-01'),
(2, 'JOHN', '1990-08-14'),
(3, 'SAMEER, '1991-01-01'),
(4, 'JACOB, '1991-04-01');
```



DISTRIBUTION KEYS - EXAMPLE

```
CREATE TABLE RS_DEMO (
       id     INT,
       name   VARCHAR(20),
       dob    DATE
) DISTSTYLE ALL;
```

```
INSERT INTO RS_DEMO VALUES
(1, 'SAM', '1990-08-01'),
(2, 'JOHN', '1990-08-14'),
(3, 'SAMEER, '1991-01-01'),
(4, 'JACOB, '1991-04-01');
```

COMPRESSION ENCODINGS

Goals

- Allow more data to be stored within an Amazon Redshift cluster
- Improve query performance by decreasing I/O
- 13 different compression encodings available

Impact

- Allows storage of 2–4 times more data within the cluster

By default, COPY automatically analyses and compresses data on first load into an empty table

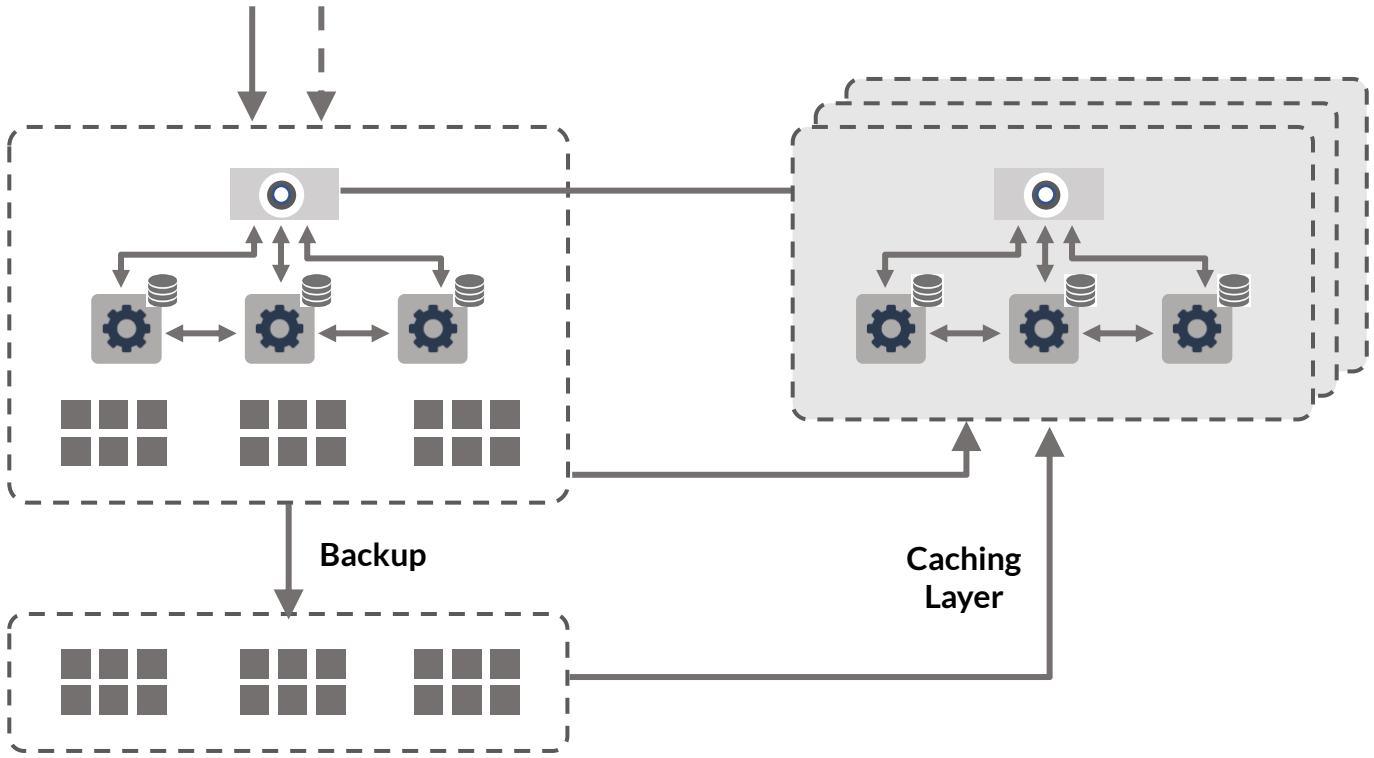
ANALYZE COMPRESSION is a built-in command that finds the optimal compression for each column in an existing table

AZ64 is a new, Amazon proprietary compression encoding algorithm, which offers better storage savings and high query performance compared to other Amazon Redshift encodings

	AZ64 storage savings	AZ64 performance speed ups
RAW	60–70% less storage	25–30% faster
LZO	35% less storage	40% faster
ZSTD	Comparable footprint	70% faster

CONCURRENCY SCALING

Amazon Redshift automatically adds compute capacity within seconds to serve unprecedented read queries. Thus supporting virtually unlimited number of connections to the cluster.

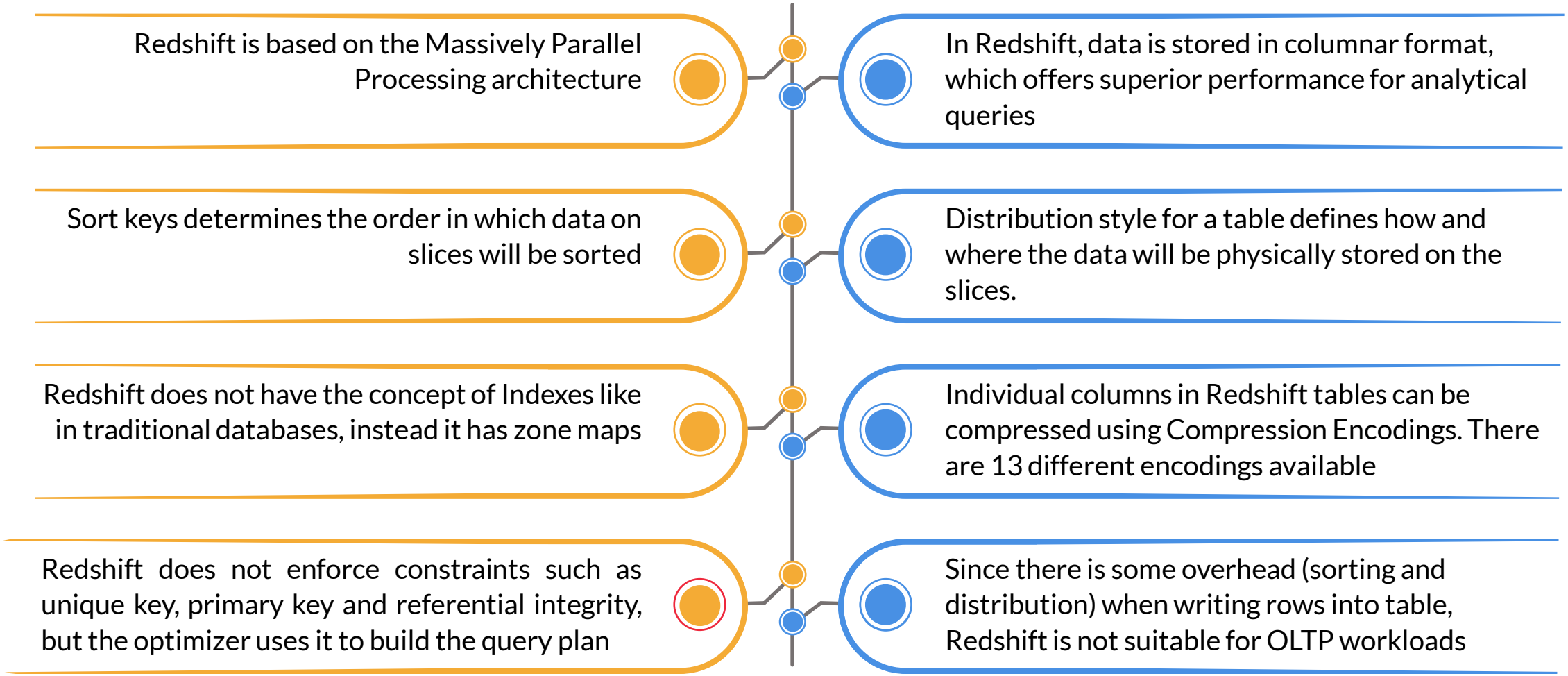


For every 24 hours that your main cluster is in use, you accrue a one-hour credit for Concurrency Scaling. This means that Concurrency Scaling is free for >97% of customers.

How it works:

1. All queries go to the leader node
2. When queries begin queuing, Amazon Redshift creates a snapshot of the cluster
3. Concurrency scaling cluster is provisioned with the help of snapshot
4. More capacity is added to the concurrency scaling cluster as needed

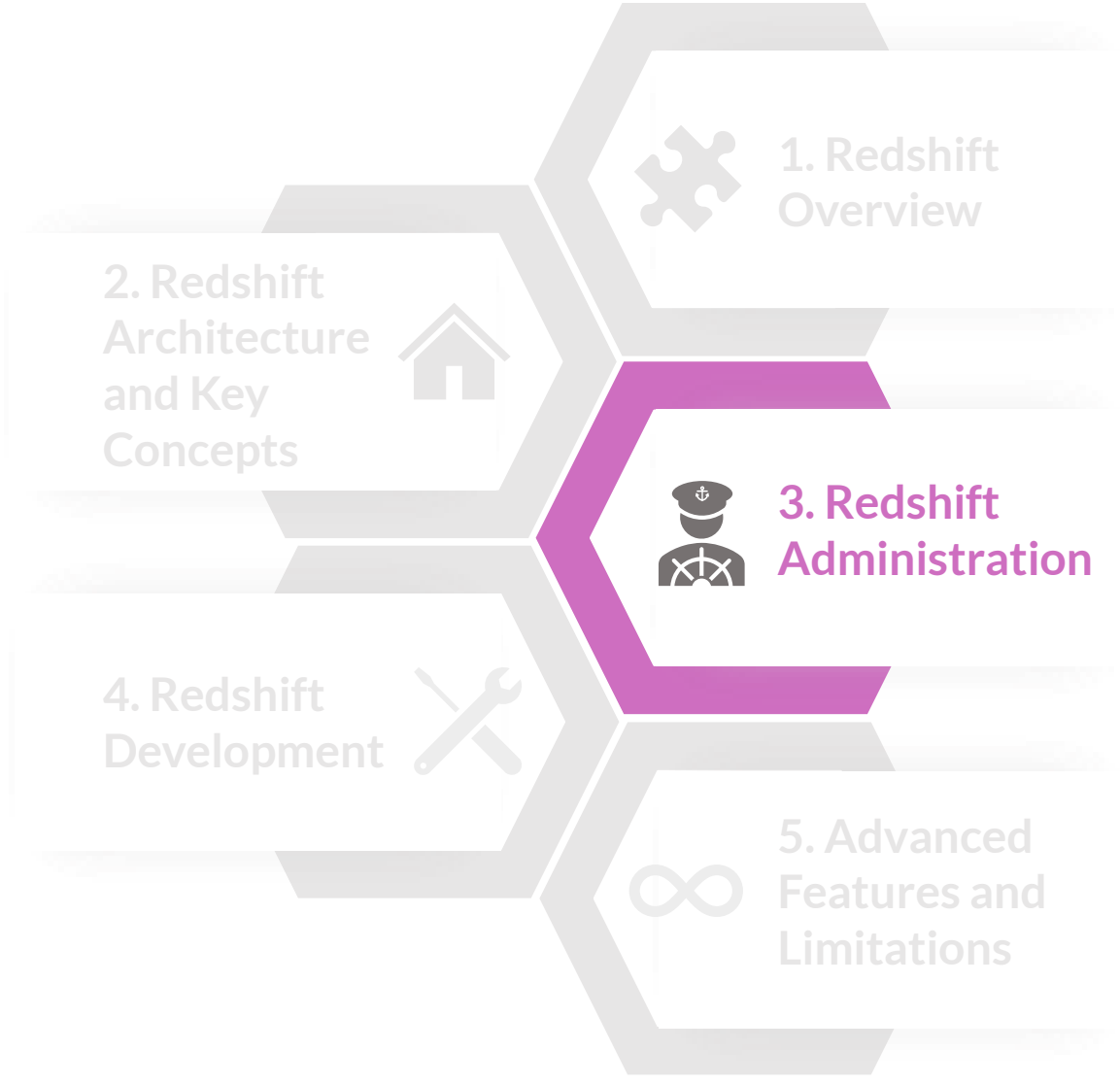
SESSION SUMMARY



Session 3: Redshift Administration

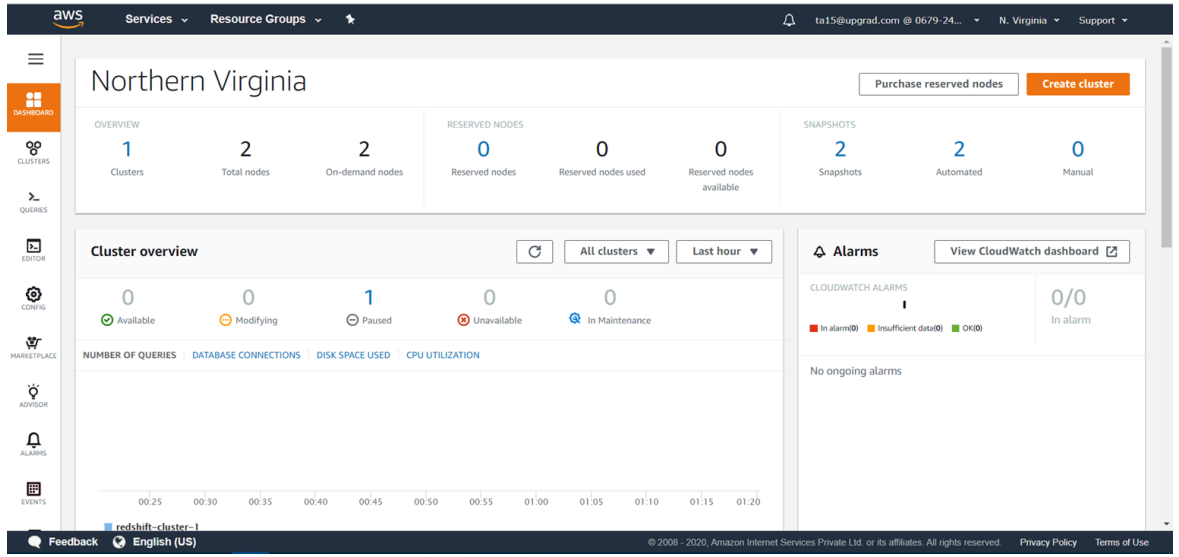
Upon completion of this session, you will learn:

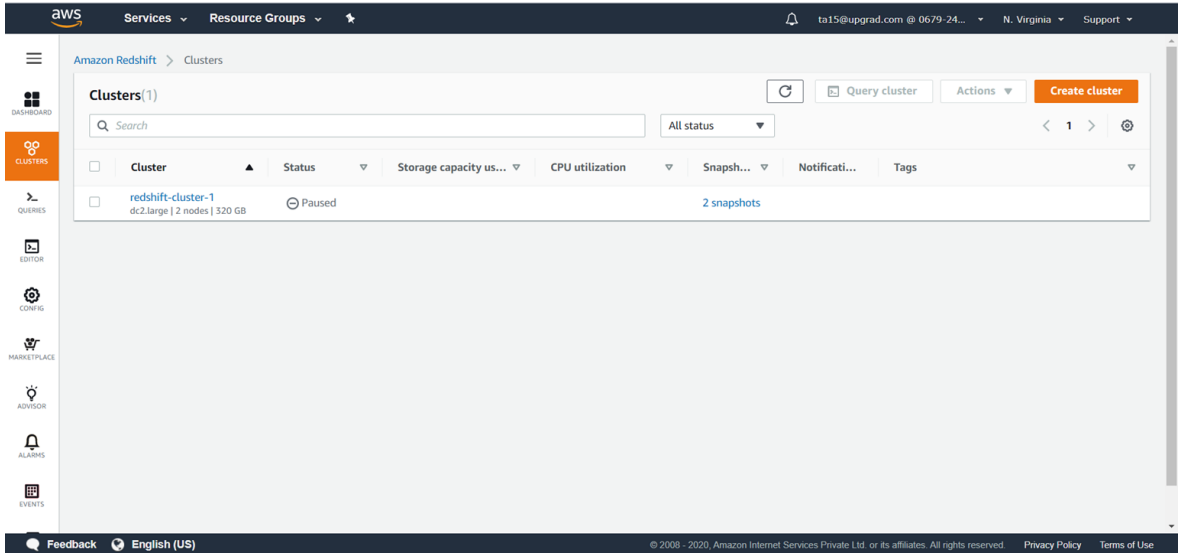
1. How to create a new Redshift cluster?
2. What are the different node types available for Redshift?
3. What is Workload Management (WLM)?
4. How to resize a Redshift cluster?
5. How to take backup of and restore a Redshift cluster?
6. What security measures are in place for Amazon Redshift?

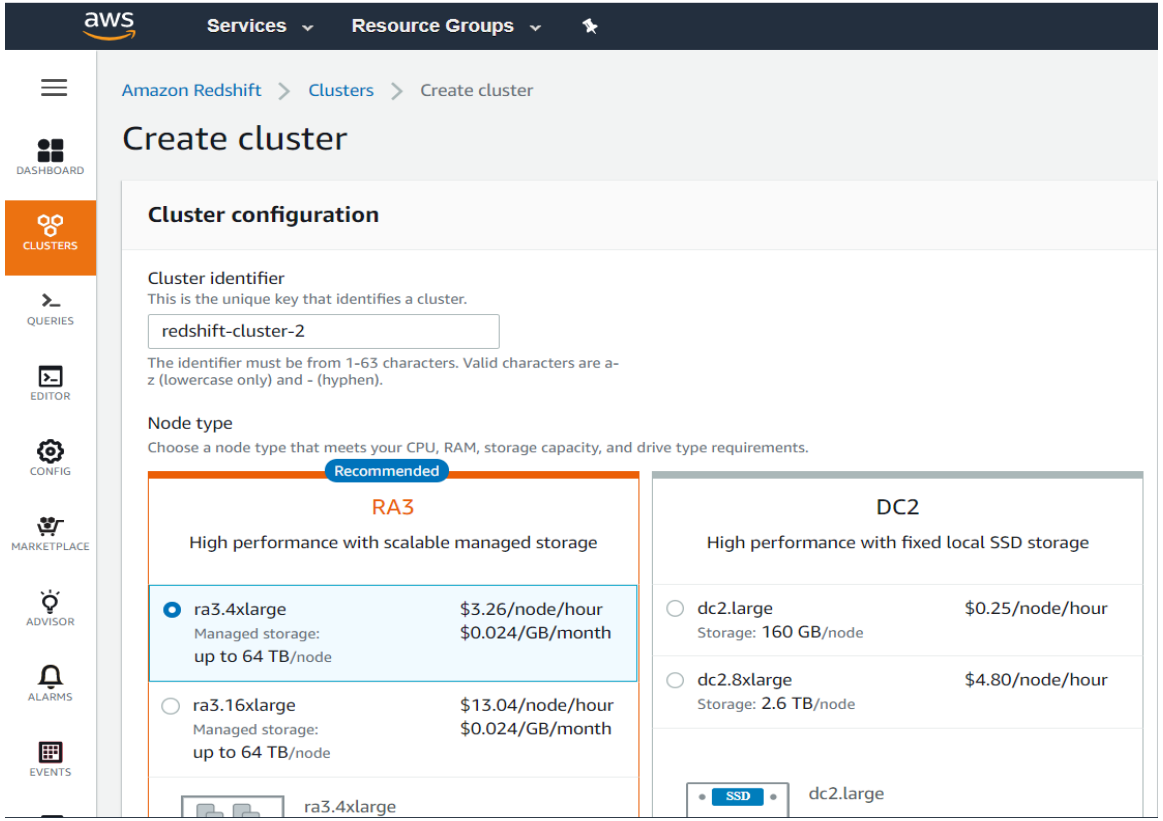


CREATING A REDSHIFT CLUSTER

Demo







REDSHIFT NODE TYPES

Dense storage—DS2

- For large data warehouses with HDD (Magnetic) disks
- For ds2.xlarge $0.85 per Hour
- For ds2.8xlarge $6.80 per Hour

Dense compute—DC2

- For compute-intensive data warehouses with Solid-state disks
- For dc2.large $0.25 per Hour
- For dc2.8xlarge $4.80 per Hour

Amazon Redshift analytics—RA3

- Amazon Redshift Managed Storage (RMS)—Solid-state disks + Amazon S3
- For ra3.4xlarge $3.26 per Hour
- For ra3.16xlarge $13.04 per Hour

Instance type	Disk type	Size	Memory	CPUs	Slices
RA3 4xlarge (new)	RMS	Scales to 16 TB	96 GB	12	4
RA3 16xlarge (new)	RMS	Scales to 64 TB	384 GB	48	16
DC2 large	SSD	160 GB	16 GB	2	2
DC2 8xlarge	SSD	2.56 TB	244 GB	32	16
DS2 xlarge	Magnetic	2 TB	32 GB	4	2
DS2 8xlarge	Magnetic	16 TB	244 GB	36	16

REDSHIFT MAINTENANCE

Amazon Redshift turbo charges query performance with machine learning-based automatic optimisations

VACUUM

- VACUUM removes rows that are marked as deleted and globally sort tables
- For the majority of workload, AUTO VACUUM DELETE will reclaim space and AUTO TABLE SORT will sort the needed portions of the table
- In cases where you know your workload, VACUUM can be run manually

ANALYZE

- The ANALYZE process collects table statistics for optimal query planning
- In the vast majority of cases, AUTO ANALYZE automatically handles statistics gathering

Automatic Analyze

Automatic Table Distribution Style

Automatic Vacuum

Automatic Table Sort

Sort Key Advisors

WORKLOAD MANAGEMENT

WLM allows for the separation of different query workloads

Goals

- Prioritize important queries
- Throttle/abort less important queries
- Control concurrent number of executing queries
- Divide cluster memory
- Set query timeouts to abort long-running queries

Queues:

- Assign a percentage of cluster memory
- SQL queries execute in queue based on
  - User group: which groups the user belongs to
  - Query group session-level variable

Query Slot:

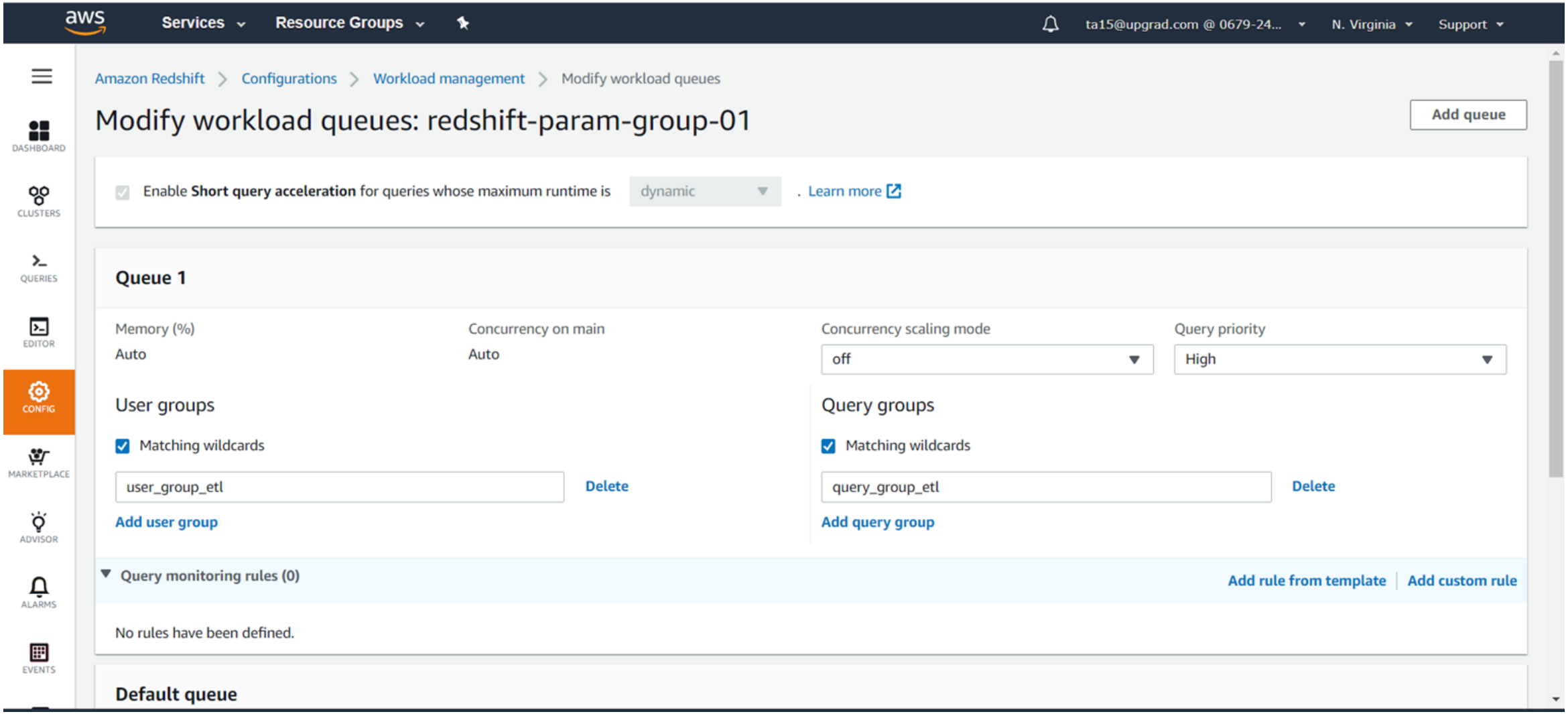
- Division of memory within a WLM queue, correlated with the number of simultaneous running queries
- WLM_QUERY_SLOT_COUNT is a session-level variable
- Useful to increase for memory-intensive operations (e.g. large COPY, VACUUM, large INSERT INTO SELECT)

Scenario 1: Single Queue for all workloads

Scenario 2: Separate queues for separate workloads

WORKLOAD MANAGEMENT

Demo



REDSHIFT SCALING – ELASTIC RESIZE

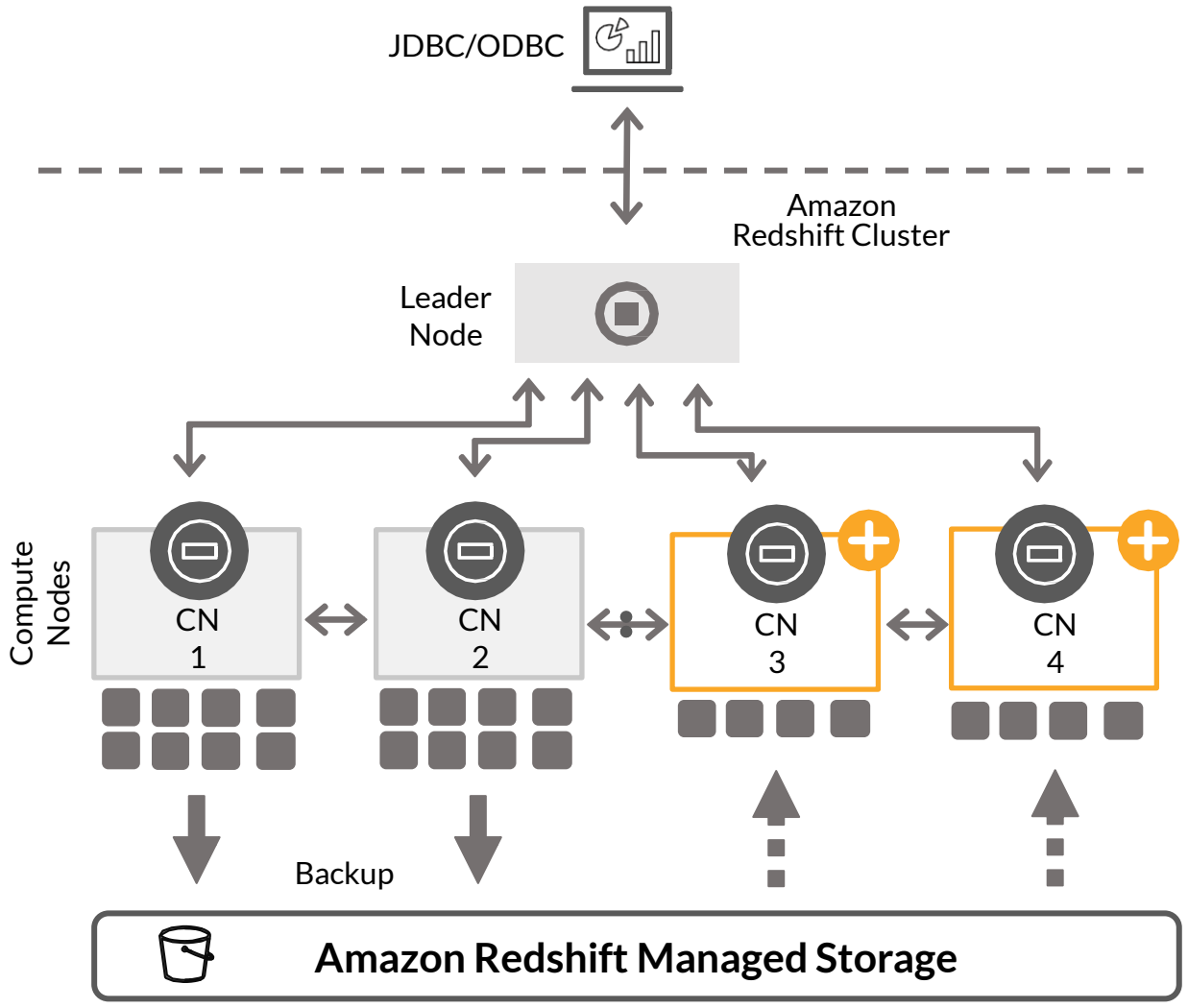
Elastic Resize

Nodes are added to/removed from the existing cluster (within minutes)

1. As soon as the resize is requested, a backup/snapshot is taken.
2. New nodes are added and made available to the cluster immediately.
3. Data is redistributed to the node slices in the background.
4. Queries are temporarily paused and connections held open, if possible.

Current Limitations

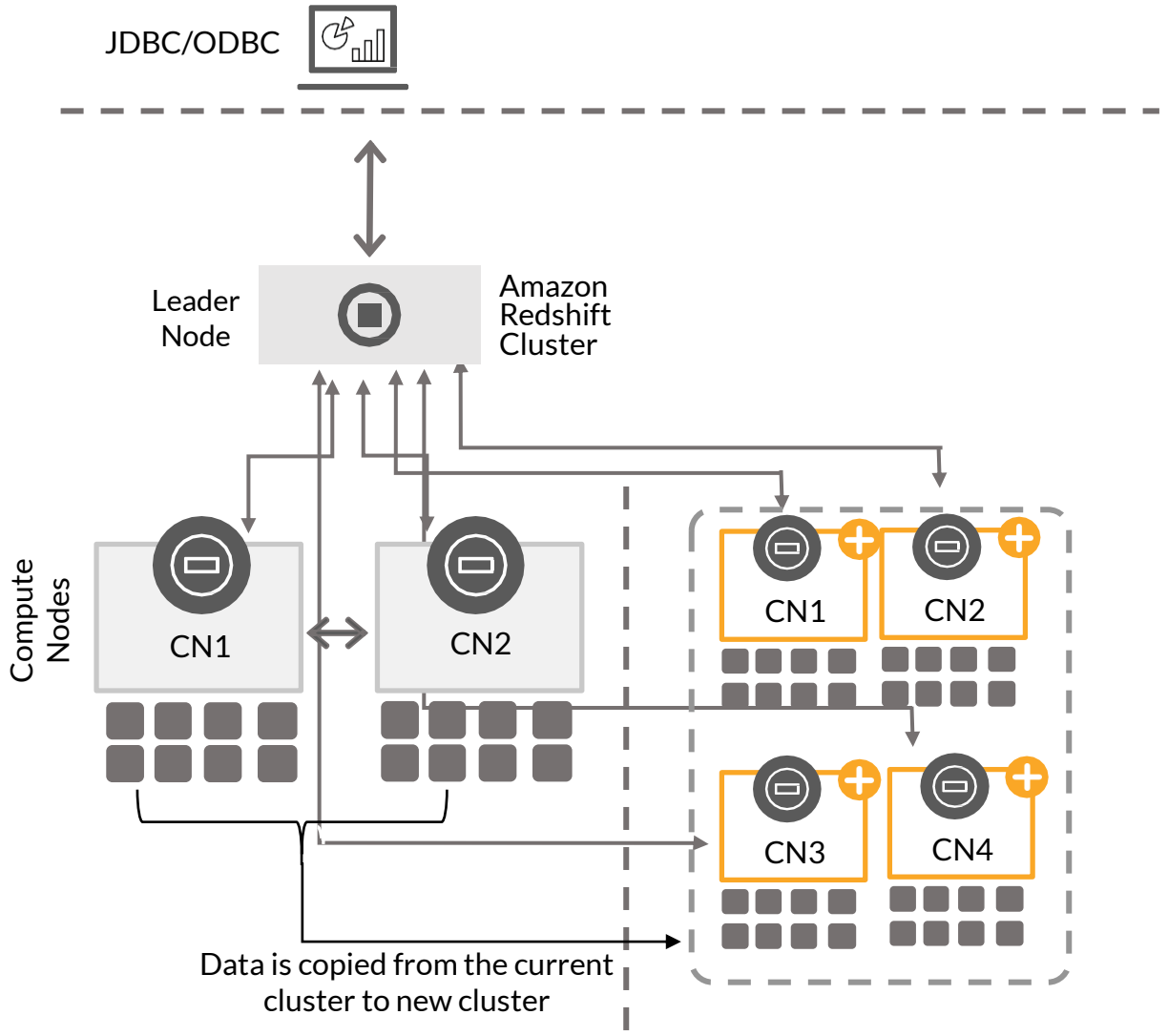
- For dc2.large or ds2.xlarge node types, you can either double or reduce **to** half the number of nodes of the original cluster, e.g., 4 to 2 or 8
- For dc2.8xlarge, ds2.8xlarge, ra3.4xlarge, or ra3.16xlarge node types, you can either reduce **till** half or double the current number of nodes, e.g., 4 to 2,3,5,6,7,8.



REDSHIFT SCALING – CLASSIC RESIZE

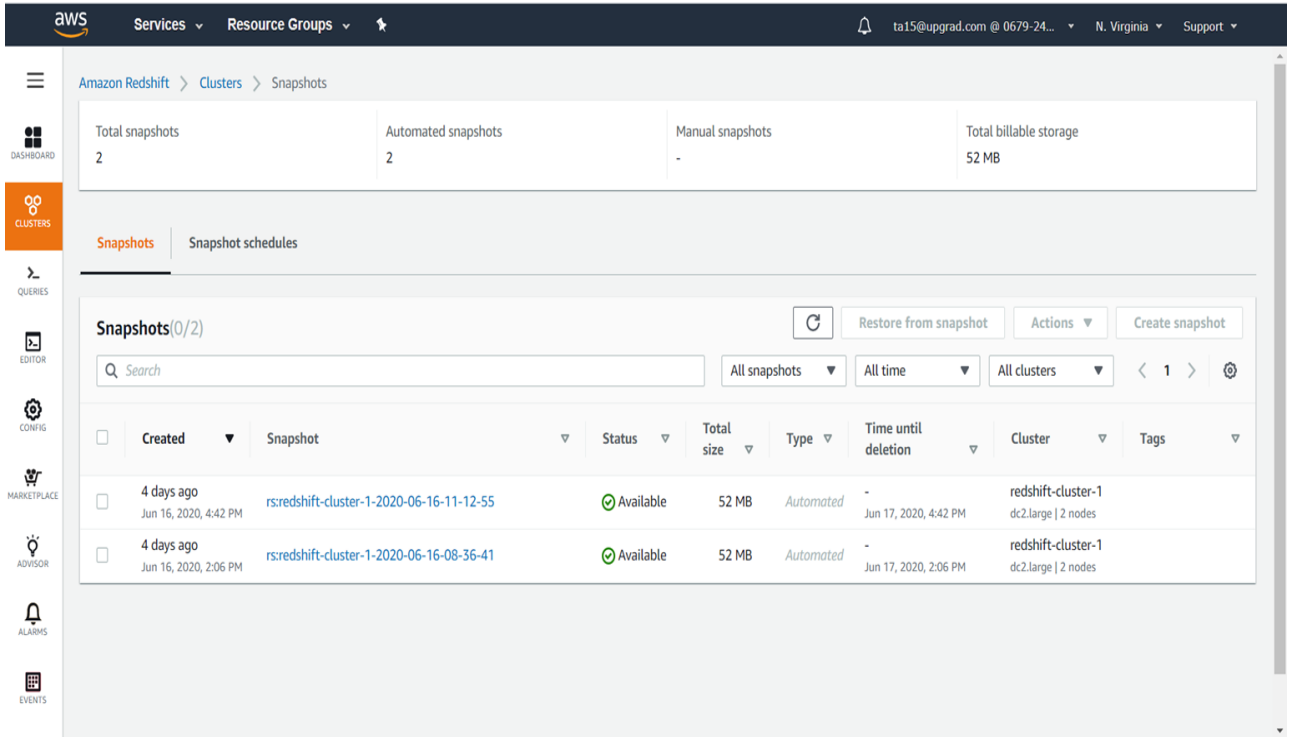
Classic Resize

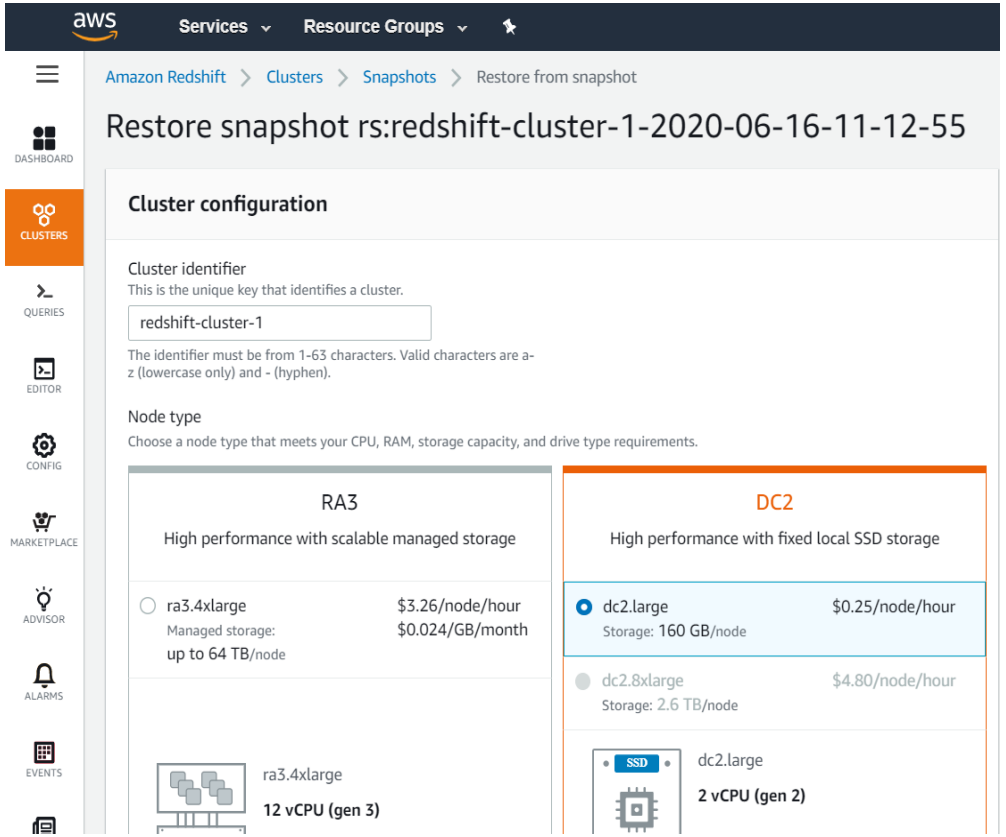
1. As soon as the resize is requested, a new cluster is created in the backend
2. The original cluster goes into read-only mode, and data is copied from the original cluster to the new cluster
3. As soon as the data copy is completed, the leader node points to the new compute nodes
4. This may take a very long time, depending upon the size of data and number of nodes in the cluster



REDSHIFT BACKUP AND RESTORE

Demo

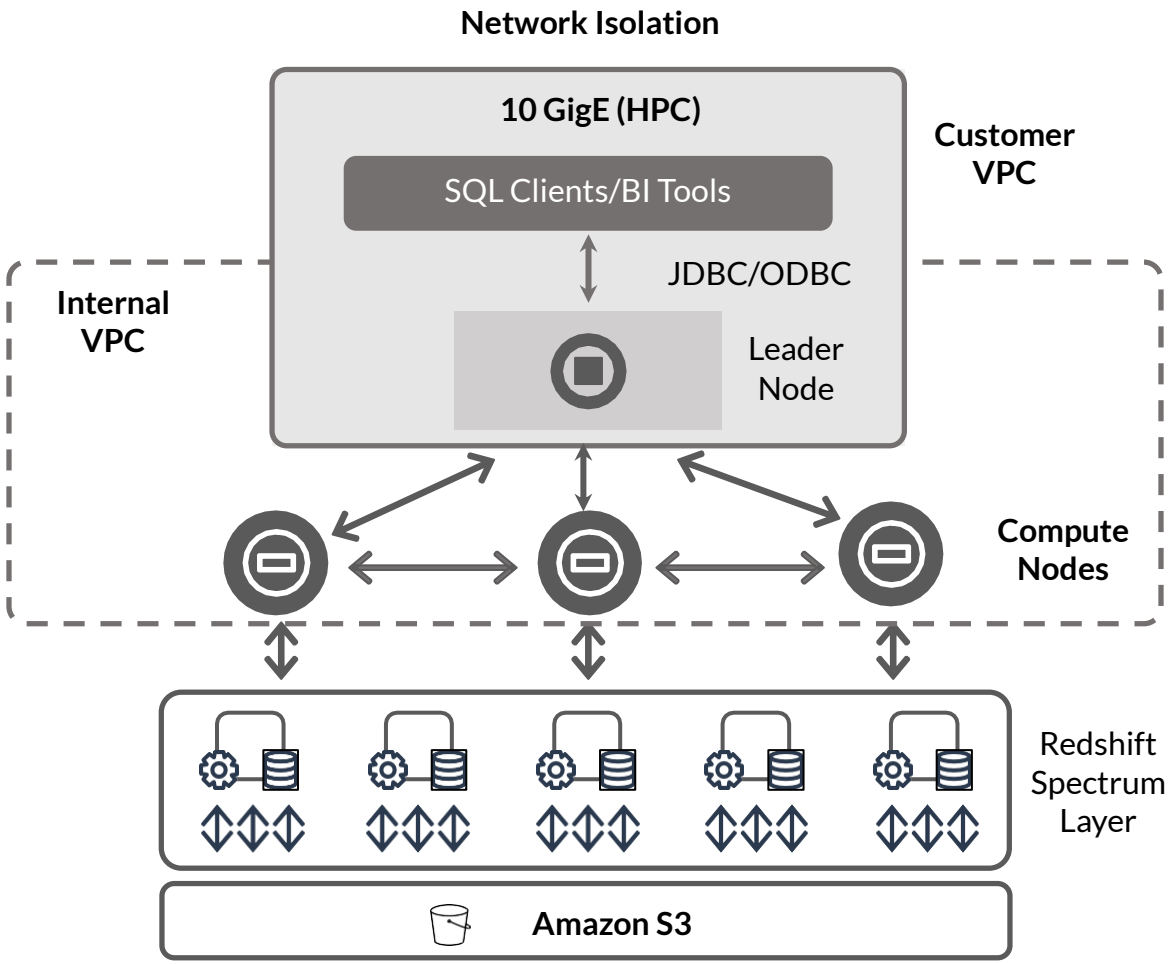




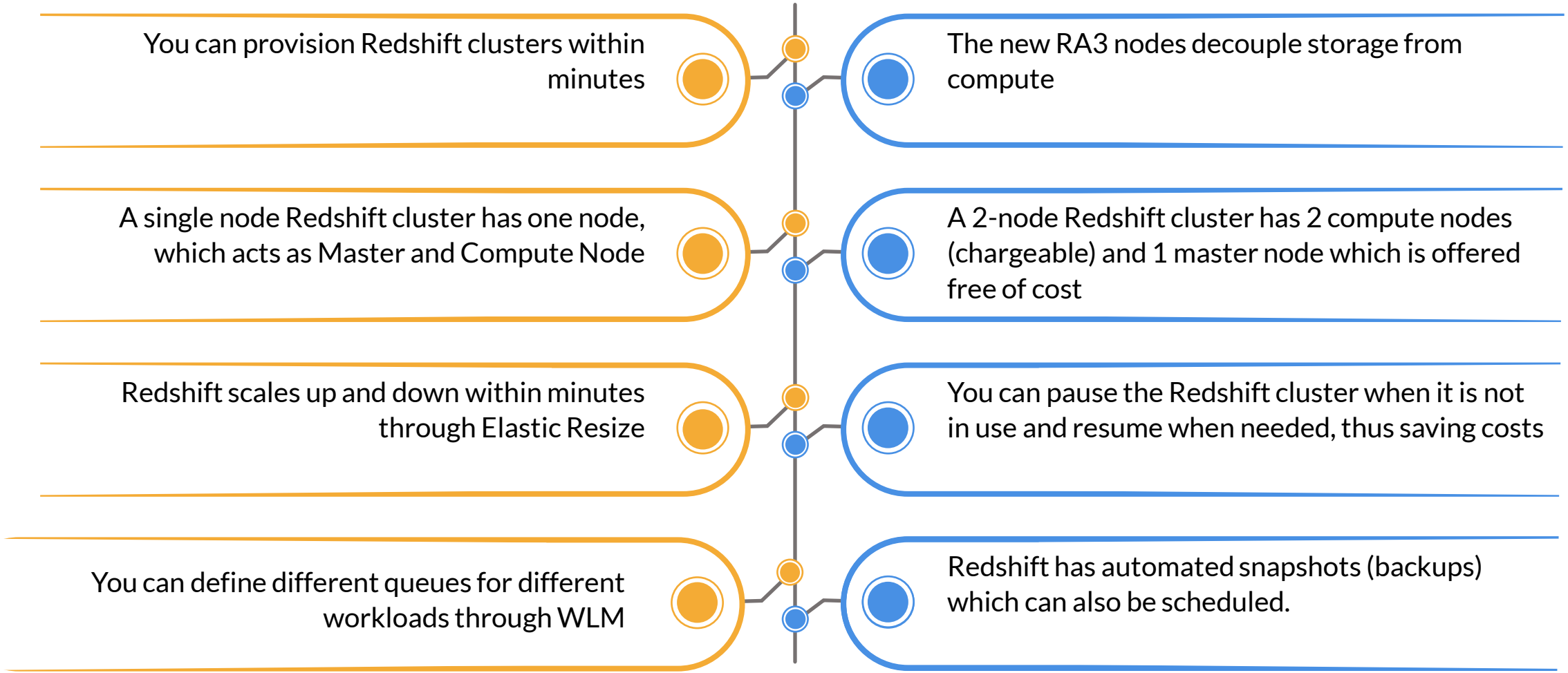
FAULT TOLERANCE AND SECURITY

- SSL to secure data in transit
- Encryption to secure data at rest
  - AES-256; hardware-accelerated
  - All data blocks encrypted
- Audit logging at AWS CloudTrail integration
- Amazon VPC support
- SOC 1/2/3, PCI-DSS level 1, FedRAMP, etc.

Select Compliance Certifications*



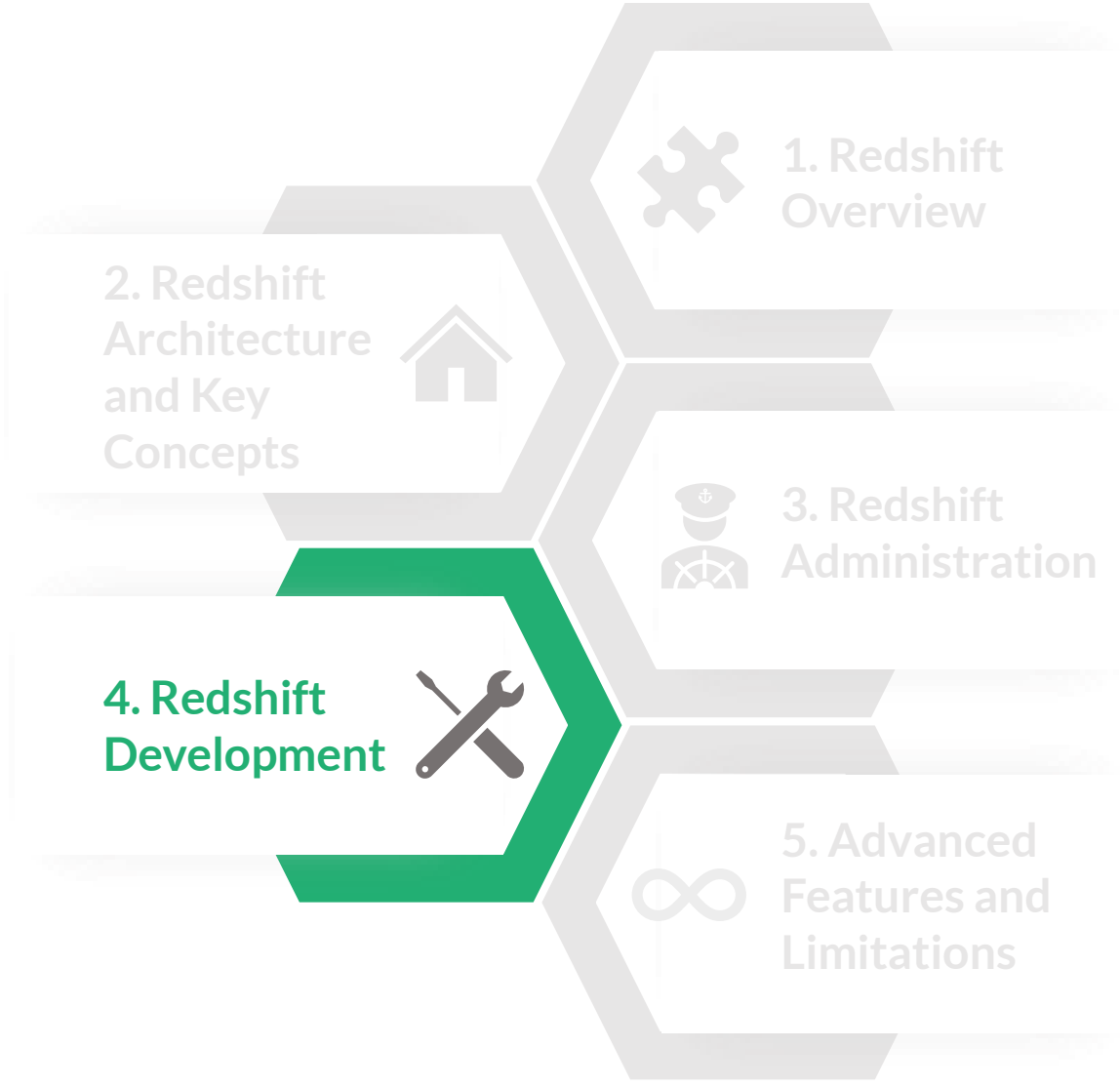
SESSION SUMMARY



Session 4: Redshift Development

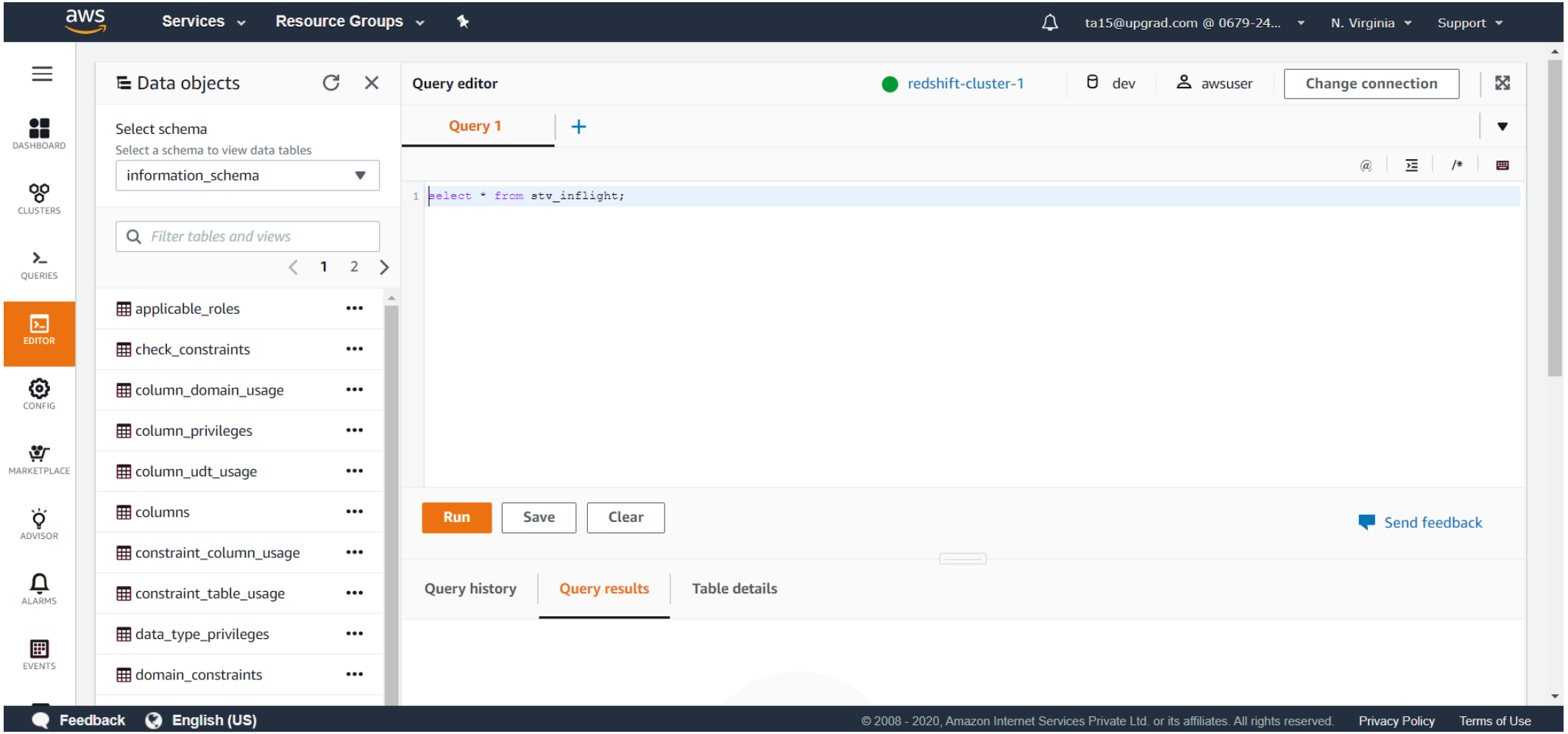
Upon completion of this session, you will learn:

1. How to connect and query a Redshift cluster?
2. How to create objects (database, schema, table) in Redshift?
3. What best practices we need to follow while designing a table?
4. How to load data into Redshift?
5. How to analyze data within Redshift?
6. What are stored procedures and UDFs?
7. Basics of how to tune Redshift queries?



CONNECT TO REDSHIFT CLUSTER

Demo



BASIC SQL OPERATIONS IN REDSHIFT

Demo

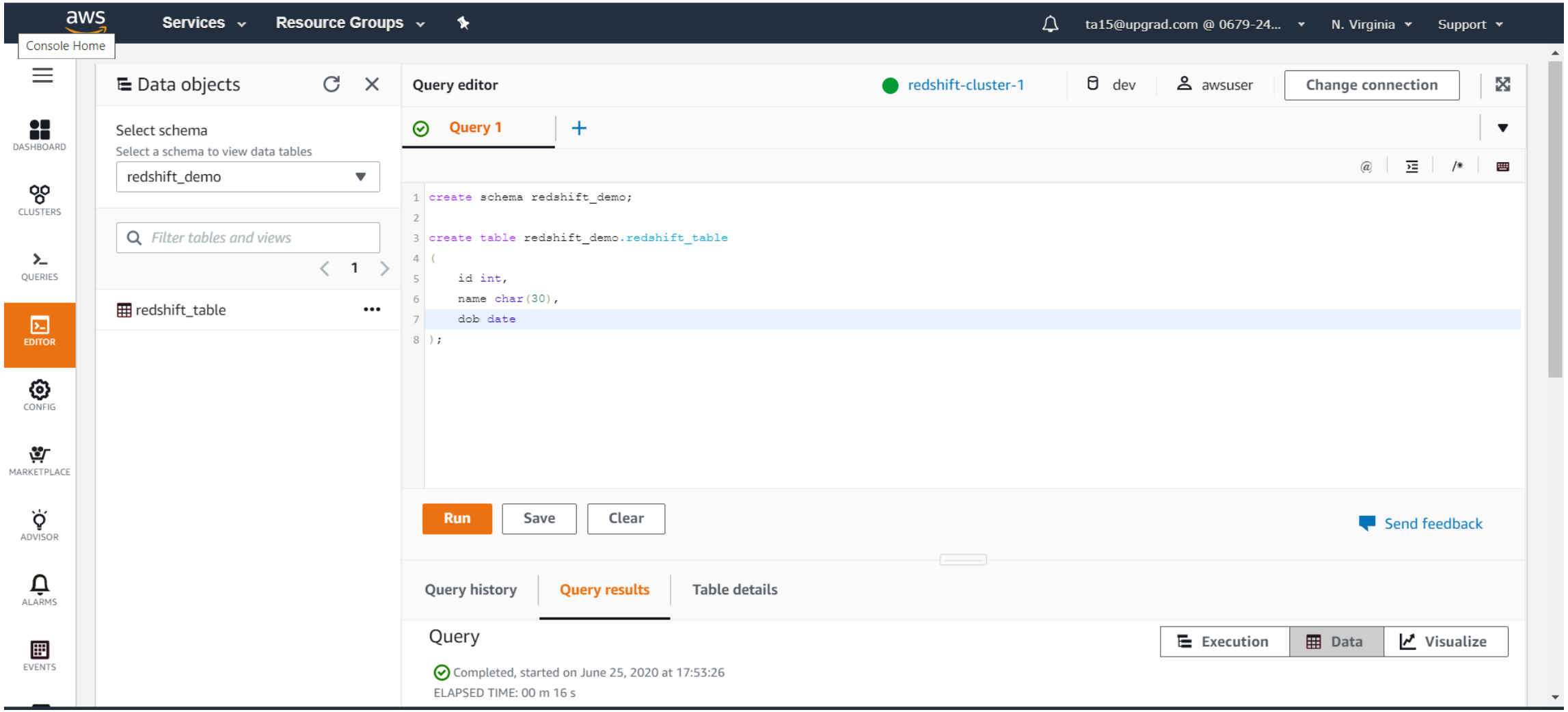


TABLE DESIGN BEST PRACTICES

Table Compression

Use AZ64 where possible, ZSTD/LZO for most (VAR)CHAR columns

Sort Keys

Add sort keys to the columns that are frequently filtered on

Distribution Keys

A good distribution key should have commonly joined columns, with no/less skew

Define Constraints

Though Redshift does not enforce foreign/primary keys, optimizer uses those constraints to generate more efficient query plans

Use Spectrum

Create external tables for infrequently used data and query using Spectrum

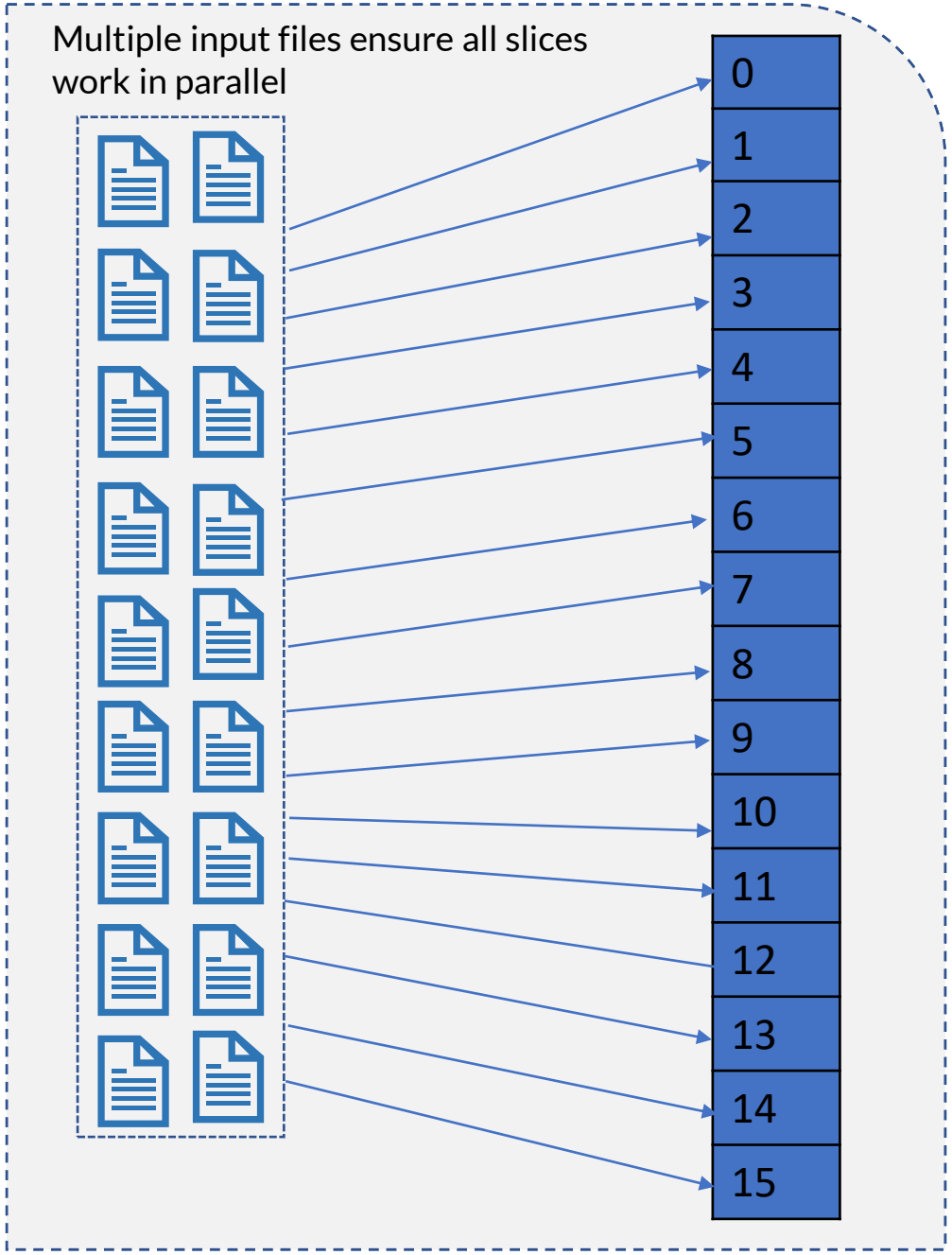
Data Types

Use appropriate data types, such as timestamp for date columns instead of char/varchar

LOADING DATA IN REDSHIFT

COPY command facilitates loading the data in Redshift from S3

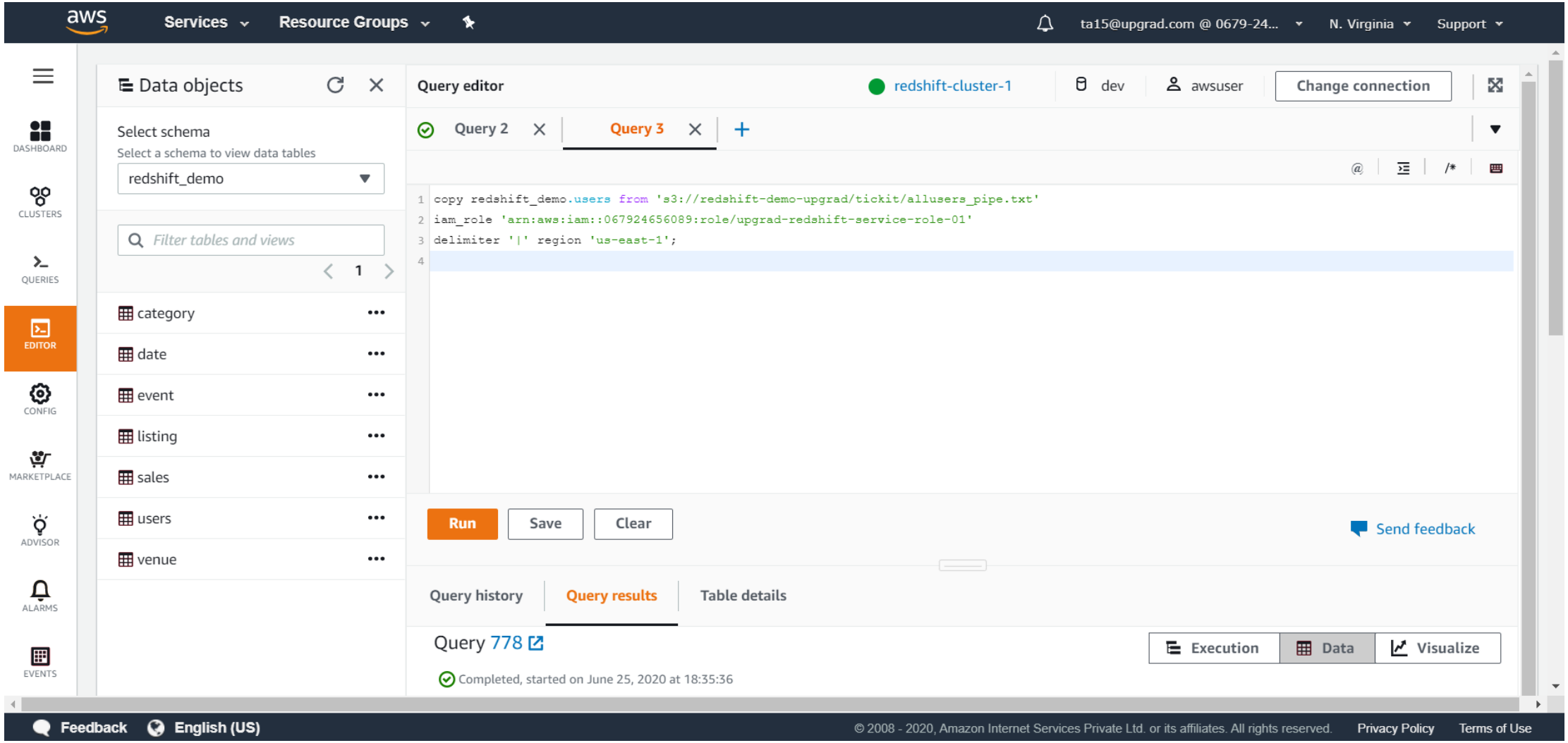
- Redshift is designed to load large amounts of data in parallel
- Write throughput is almost directly proportional to the number of nodes; in other words, the more the number of files, better is the COPY performance
- Recommendation is to use delimited files—1 MB to 1 GB after compression (gzip)
- Number of input files should be a multiple of the number of slices in the Redshift cluster
- Loading data in alternative file formats like Parquet enables significant performance improvement



dc2.8xlarge
16 slices

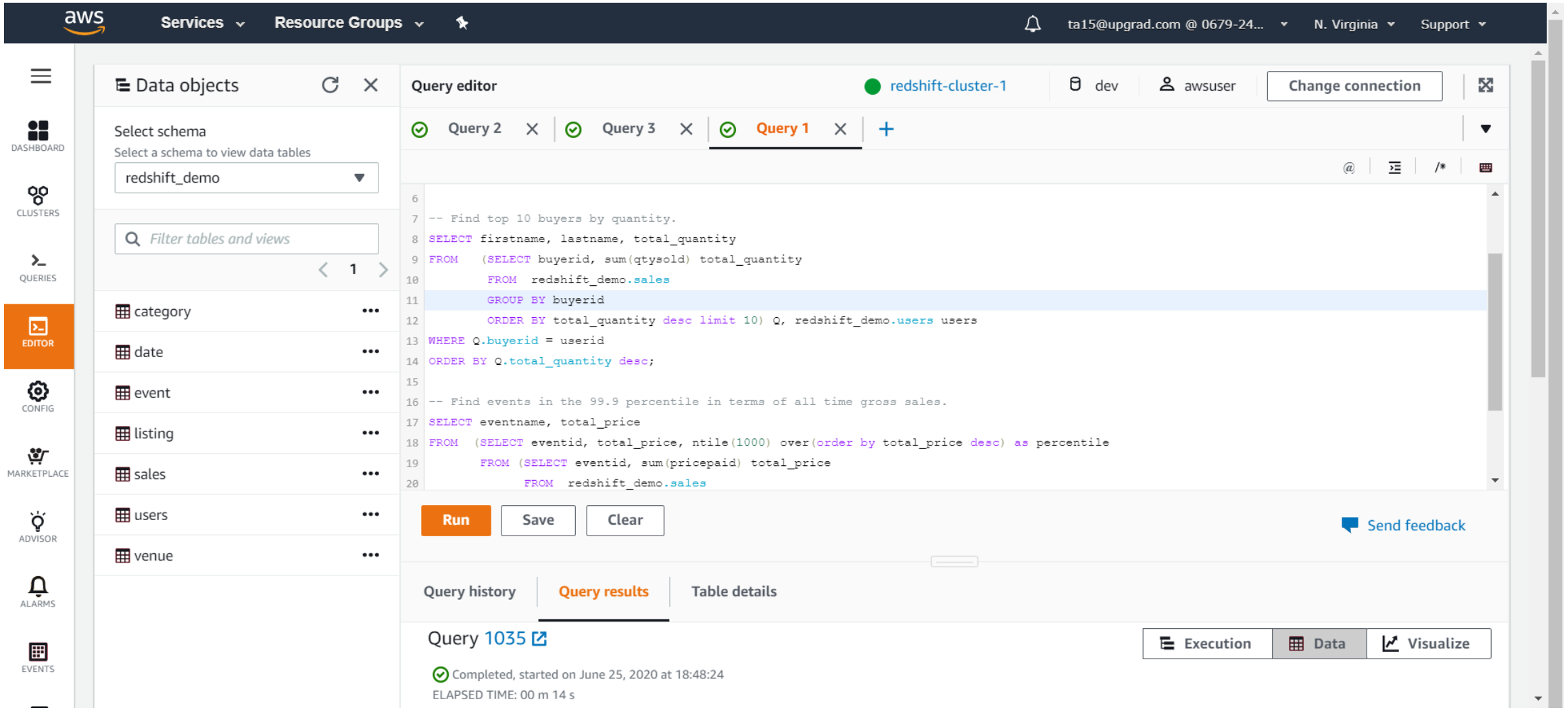
LOADING AND UNLOADING DATA IN REDSHIFT

Demo



ANALYSING DATA WITH REDSHIFT

Demo



UDFs AND STORED PROCEDURES

Demo

User defined Functions (UDFs)

- Redshift supports custom user-defined functions (scalar) using Python language
- Ability to import custom Python modules makes it a useful feature
- Allows you to create reusable components for tasks such as complex arithmetic calculations
- Cannot use SQL queries within a UDF

Example: Returning the next business day with respect to US Federal Holidays and a M-F work week.

Stored Procedures

- Redshift supports stored procedures in the PL/pgSQL format, including loops, conditionals, case statements, and IN/OUT/INOUT argument passing
- Allows you better 'lift-and-shift' from traditional data warehouses
- Loops and conditional logic run on the leader node of Amazon Redshift cluster, and any SQL within it is distributed to compute nodes

Example: Running batch SQLs on your redshift cluster

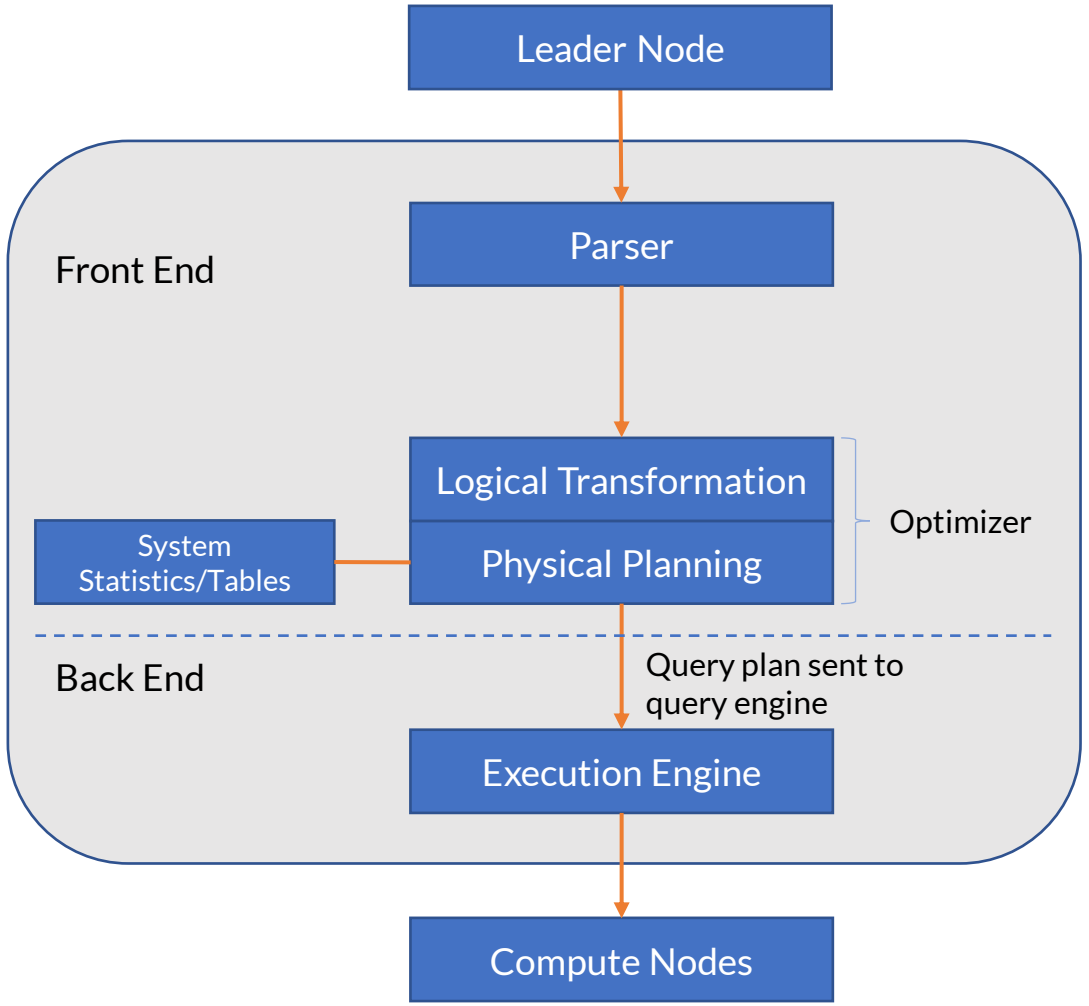
PERFORMANCE TUNING BASICS

Redshift Query Lifecycle

1. Planning
SQL query is parsed and optimized in order to create a query plan

2. Compilation
Compile cache is checked for a query plan match; otherwise the query plan is converted to subtasks and subtasks are compiled into C++

3. Execution
Compiled code is executed by compute nodes slices in parallel, and results are aggregated by the leader node



PERFORMANCE TUNING BASICS

Demo

EXPLAIN select * from redshift_demo.category, redshift_demo.event where category.catid=event.catid;

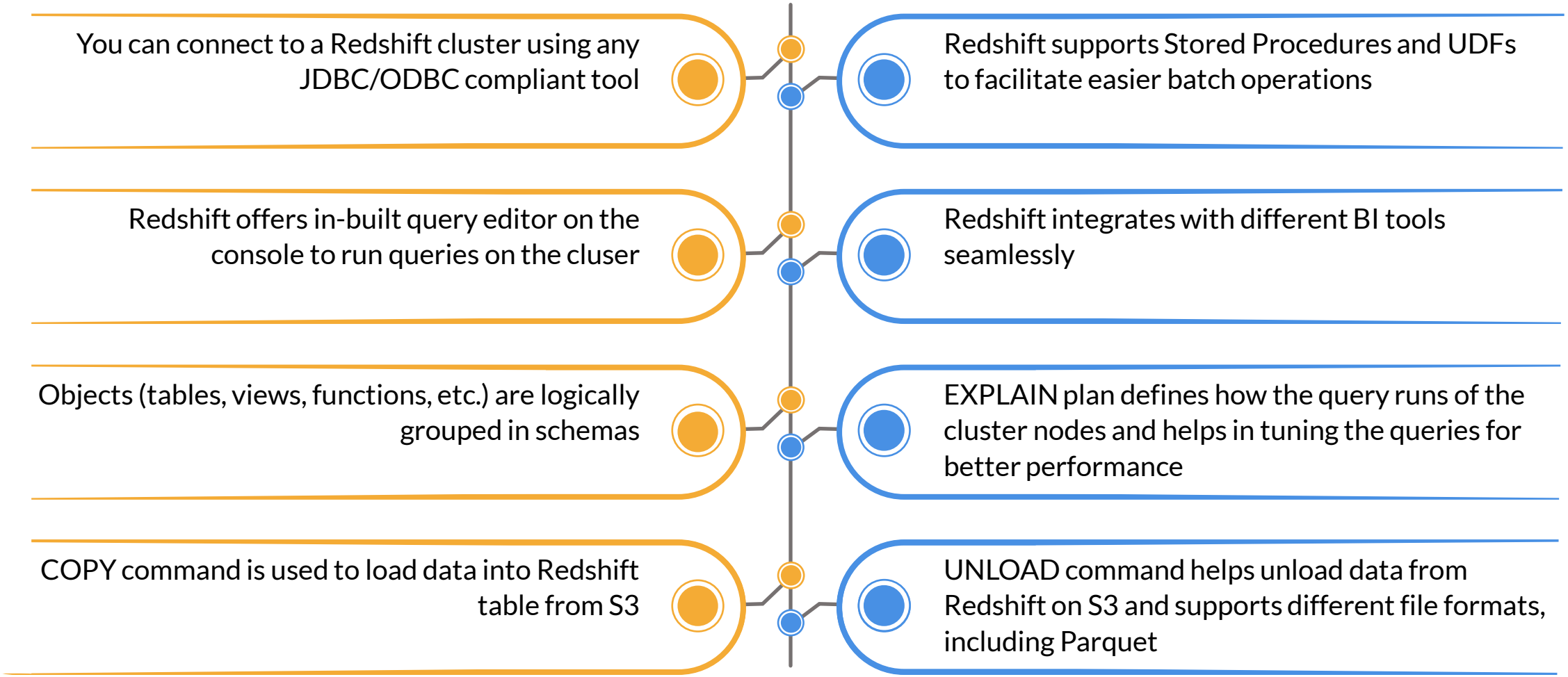
Bad Query Plan

```
XN Hash Join DS_BCAST_INNER
(cost=0.14..6600286.07 rows=8798
width=84)
Hash Cond: ("outer".catid =
"inner".catid)
-> XN Seq Scan on event
(cost=0.00..87.98 rows=8798
width=35)
-> XN Hash (cost=0.11..0.11
rows=11 width=49)
-> XN Seq Scan on category
(cost=0.00..0.11 rows=11
width=49)
```

Good Query Plan

```
XN Hash Join DS_DIST_ALL_NONE
(cost=109.98..747.87 rows=8798
width=84)
Hash Cond: ("outer".catid =
"inner".catid)
-> XN Seq Scan on category
(cost=0.00..0.11 rows=11 width=49)
-> XN Hash (cost=87.98..87.98
rows=8798 width=35)
-> XN Seq Scan on event
(cost=0.00..87.98 rows=8798
width=35)
```

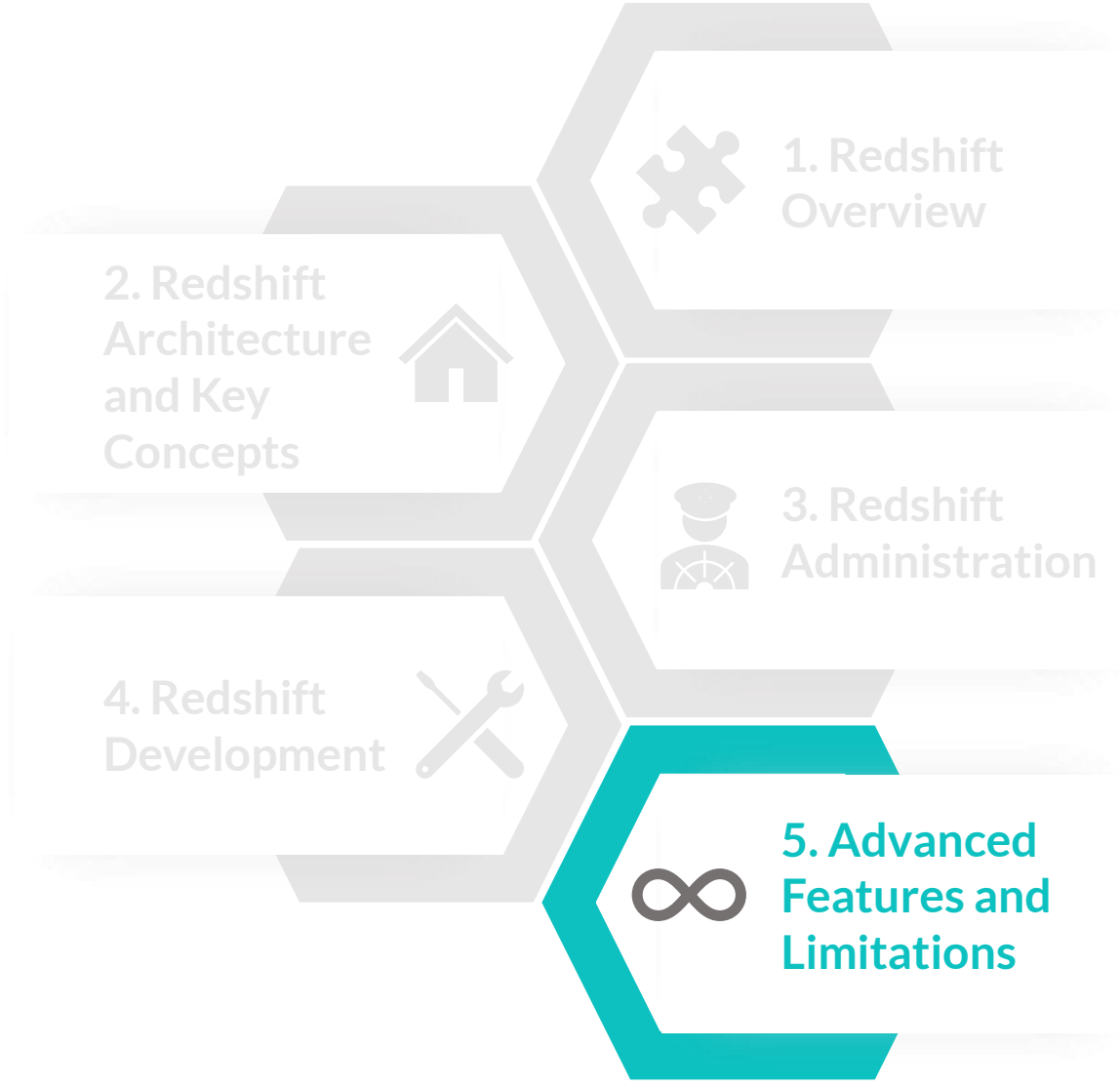
SESSION SUMMARY



Session 5: Redshift Advanced Features and Limitations

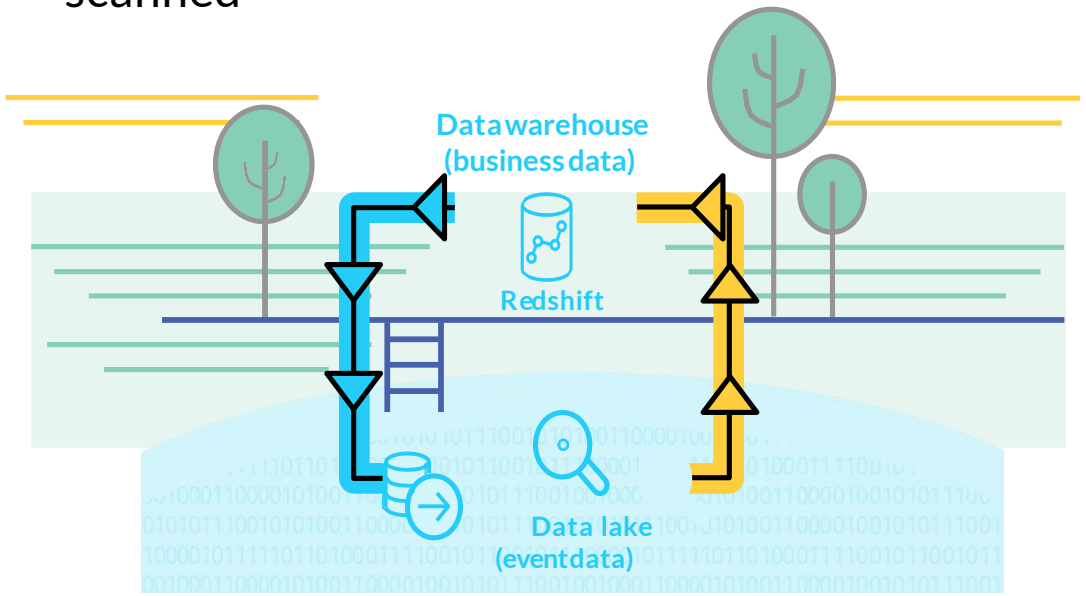
Upon completion of this session, you will learn:

1. About Amazon Redshift Spectrum
2. About Advanced Query Accelerator (AQUA)
3. What are some limitations of Redshift?
4. How do we optimize Redshift costs?

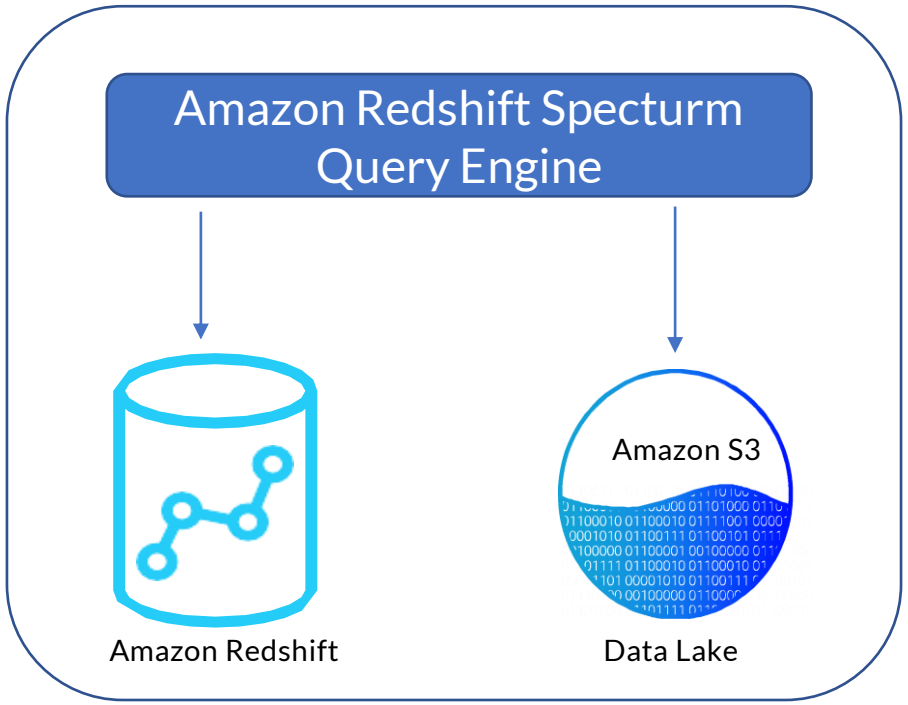


AMAZON REDSHIFT SPECTRUM

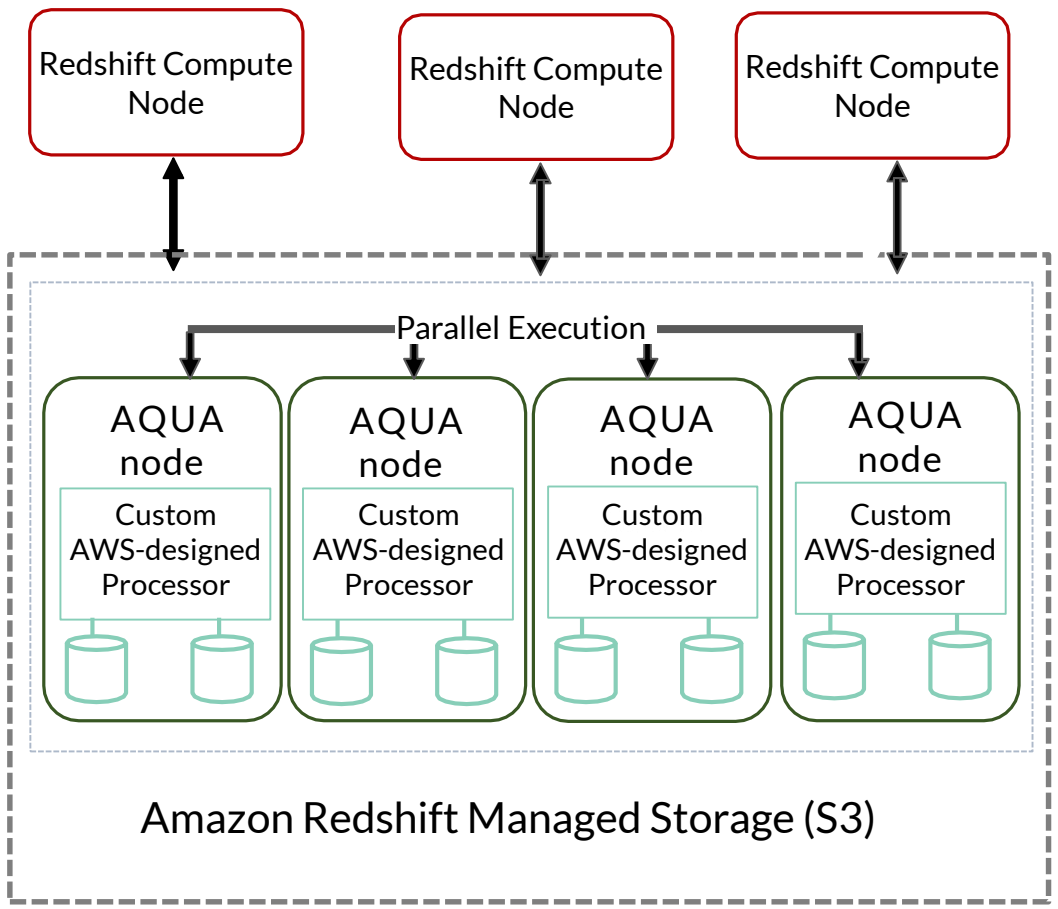
- Redshift Spectrum enables customers to have a lake house approach
- Run SQL queries against S3 without loading data into Redshift
- Supports multiple open-source data formats such as CSV, Avro and Parquet
- On-demand pricing; pay per query and data scanned



- Spectrum utilizes a fleet of Amazon Redshift clusters, which are independent of your cluster
- Filtering and aggregation are performed at Spectrum, reducing load on the Redshift cluster



ADVANCED QUERY ACCELERATOR (AQUA)



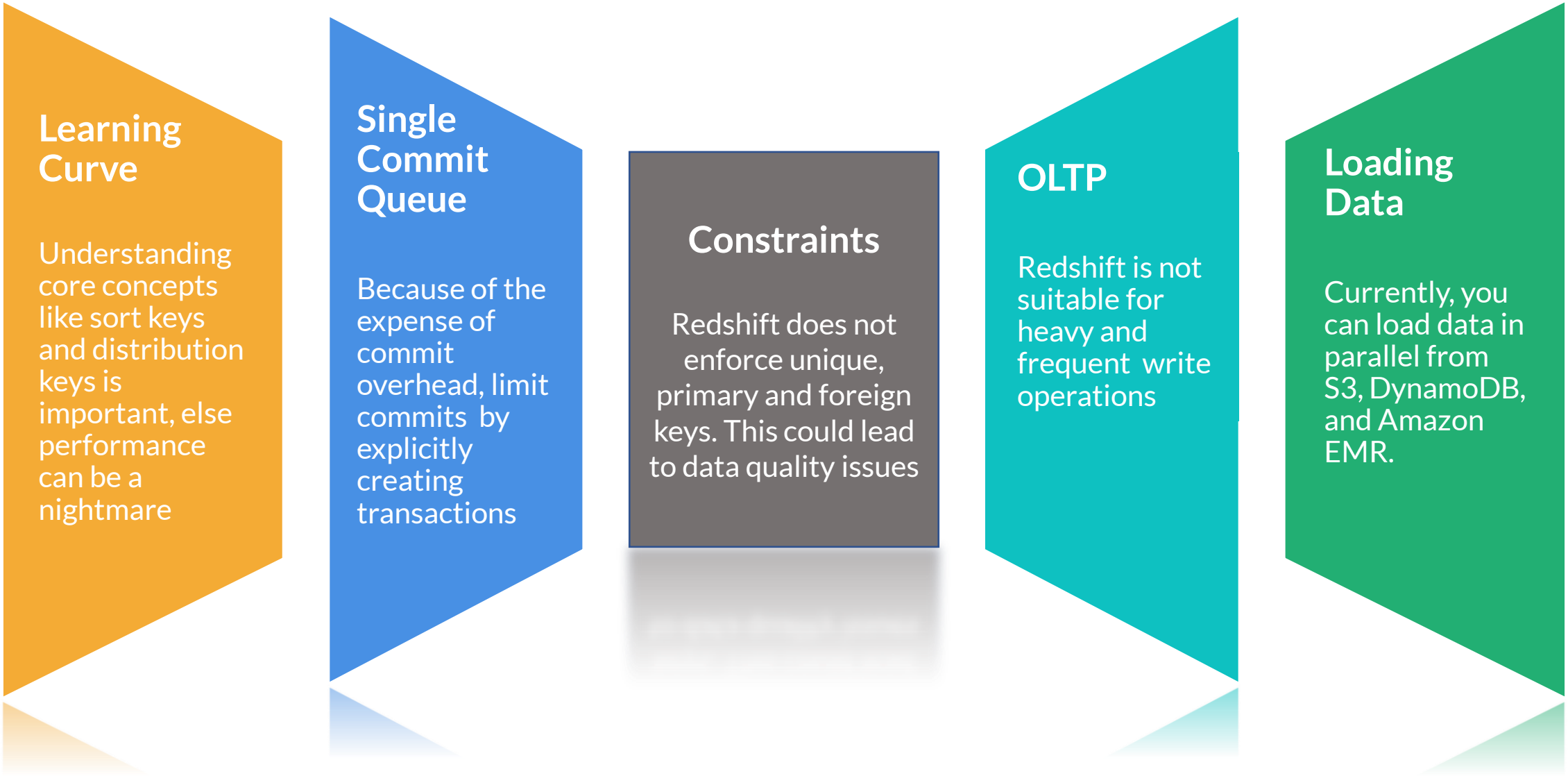
New distributed and hardware-accelerated processing layer

With AQUA, Amazon Redshift is up to **10x faster** than any other cloud data warehouse at no extra cost

AQUA Nodes with custom AWS-designed analytics processors to make operations (compression, encryption, filtering, and aggregations) faster than traditional CPUs

Available in Preview with RA3 with no code changes required

REDSHIFT LIMITATIONS



OPTMIZING REDSHIFT COSTS

Region Selection

Selecting the right AWS region for your cluster is very important as the cost varies per region.

For example, a 1-node ds2.xlarge cluster in US-EAST-1 would cost $0.85 per hour, but it would cost $1.36 per hour in SA-EAST-1 (South America)

Reservations

If you expect the cluster to run for at least a year, it is better to reserve the nodes for a 1-year or 3-year term, which offers significant (20% or 75%) cost savings over the on-demand pricing

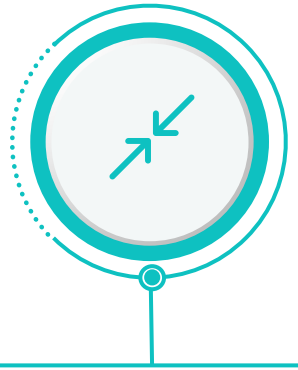
Redshift Spectrum

Offloading infrequently accessed data to S3 and querying it via Redshift Spectrum enables you to use less number of nodes on Redshift cluster, thus saving costs

Pause & Resume

In order to save costs, you can pause your Redshift clusters, especially in lower environments (development & test stages) when they are not in use.

OPTMIZING REDSHIFT COSTS



Compression

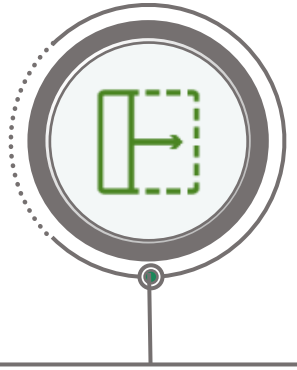
Proper column encoding ensures that data is compressed according to data type. This helps save disk space and costs, besides ensuring performance improvement

Vacuum

Ensuring that Vacuum is running (either Auto or Manual) helps free up disk space, resulting in cost savings, as less number of nodes will be required

Snapshots

AWS charges separately for the manual snapshots. This can incur significant costs. An optimal snapshot strategy may help achieve better cost savings



Rightsizing

Selection of the right size and type of Redshift cluster (memory, CPU, or storage) is important as it can easily lead to over-provisioned resources if not selected properly.

HELPFUL LINKS AND UTILITIES

https://github.com/awslabs/amazon-redshift-utils

https://github.com/awslabs/amazon-redshift-monitoring

https://github.com/awslabs/amazon-redshift-udfs

Admin scripts

Collection of utilities for running diagnostics on your cluster

Admin views

Collection of utilities for managing your cluster, generating schema DDL, and so on

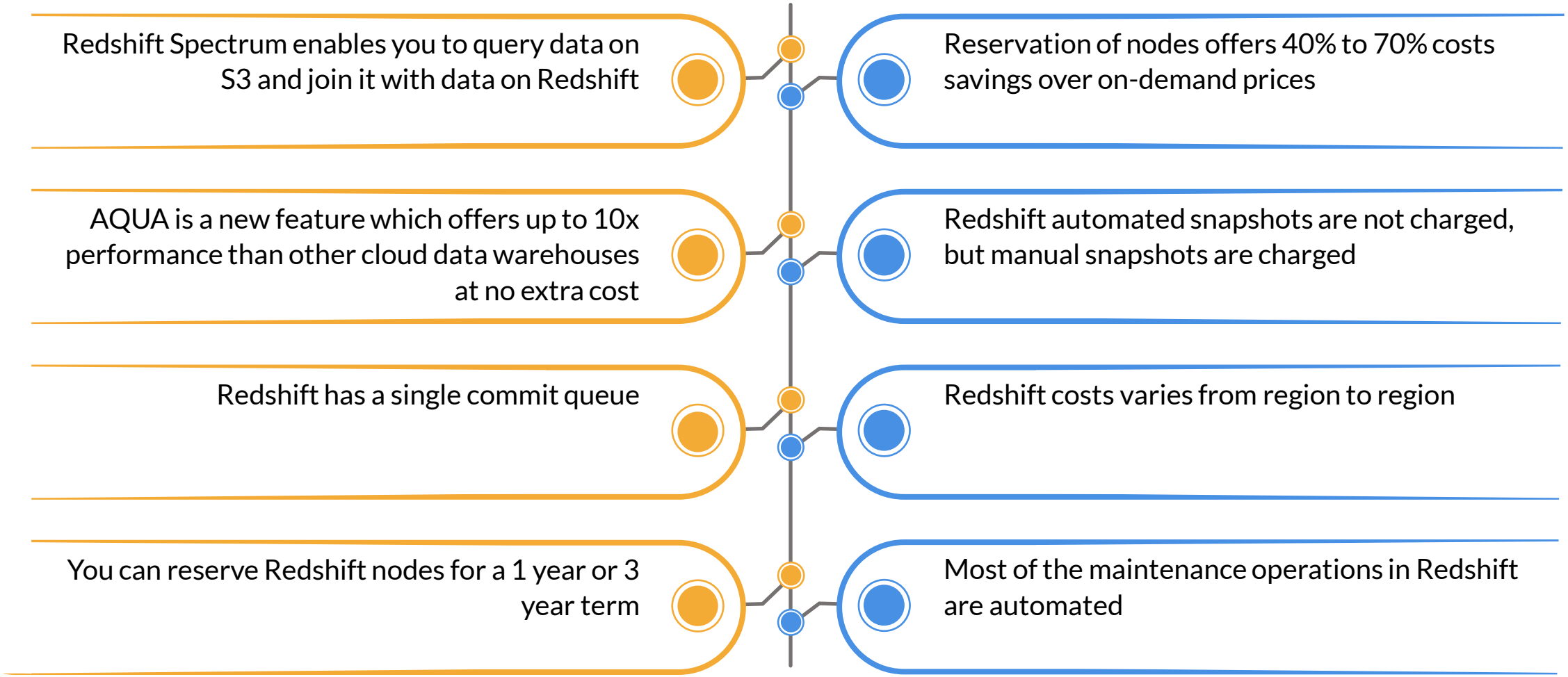
Analyze Vacuum utility

Utility that can be scheduled to vacuum and analyze the tables within your Amazon Redshift cluster

Column Encoding utility

Utility that will apply optimal column encoding to an established schema with data already

SESSION SUMMARY



MODULE SUMMARY

01 Amazon Redshift is based on PostgreSQL with additional capabilities to support OLAP operations

02 Redshift offers up too 3x performance than other DW and you can store PBs of data without impacting the performance

03 Redshift does not enforce constraints and does not support traditional indexes; instead it has sort key and distribution keys

04 Using WLM, you can define separate queues for separate workloads and assign portion of cluster memory to each queue

05 Most maintenance activities, like VACUUM and ANALYZE in Redshift are automated

06 Using Redshift Specturm, you can query the data on S3 and join it with data on Redshift

07 The new RA3 nodes decouples storage from compute and comes with AQUA, which provides up to 10x performance

08 Using COPY command, you can load in parallel into Redshift

09 Redshift offers automated snapshots which can also be scheduled

10 With node reservations, you can save up to 60% costs over on-demand pricing