

Data Warehousing

Amazon Redshift

References/Acknowledgements

<http://docs.aws.amazon.com/redshift>

Agenda

- **Use Case**
- **Data Warehouse (Intro)**
 - OLAP vs OLTP
- **Amazon Redshift**
 - AWS Service for Data Warehousing
 - Redshift Cluster Architecture
 - Key Characteristics
- **Design Considerations**
- **Launching and Accessing Redshift Cluster**

Use Case/Scenario

- **Assume Dulles Airport has become overly-crowded lately**
 - Runways are running at their full capacities
- **Govt/FAA seeks GMU's help to alleviate overcrowding of Dulles airport**
- **Option includes**
 - Expand existing airport(s)
 - Identify a new location to build a new airport
 - in Washington DC Metro area
- **What will be our approach to help guide the government to make the right decision?**
 - Which option should we recommend and how do we decide that option?
 - What is needed to answer this question?

Use Case/Scenario

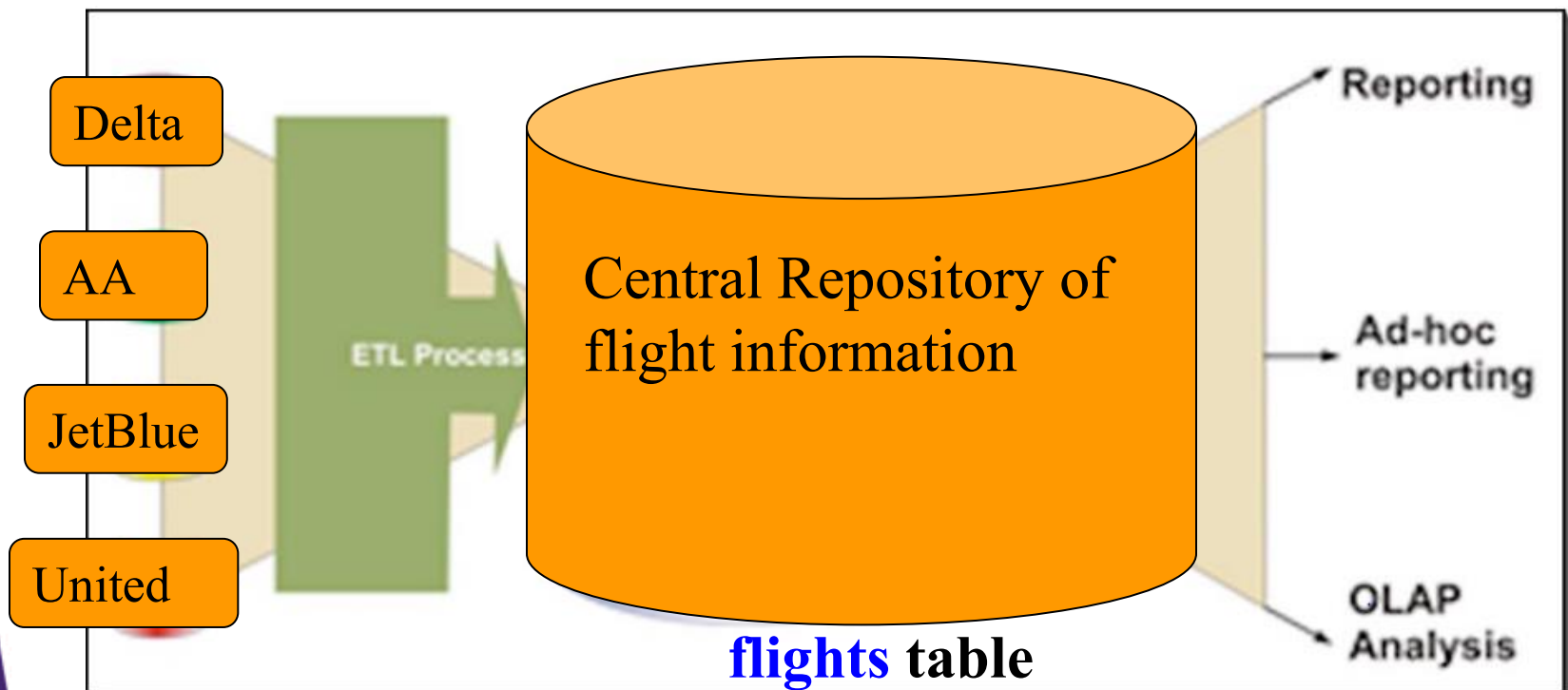
- **A crucial information:** What is the geographical area where majority of travelers live?

```
select zipcode, count(*) from flights group by zipcode;
```

Use Case/Scenario

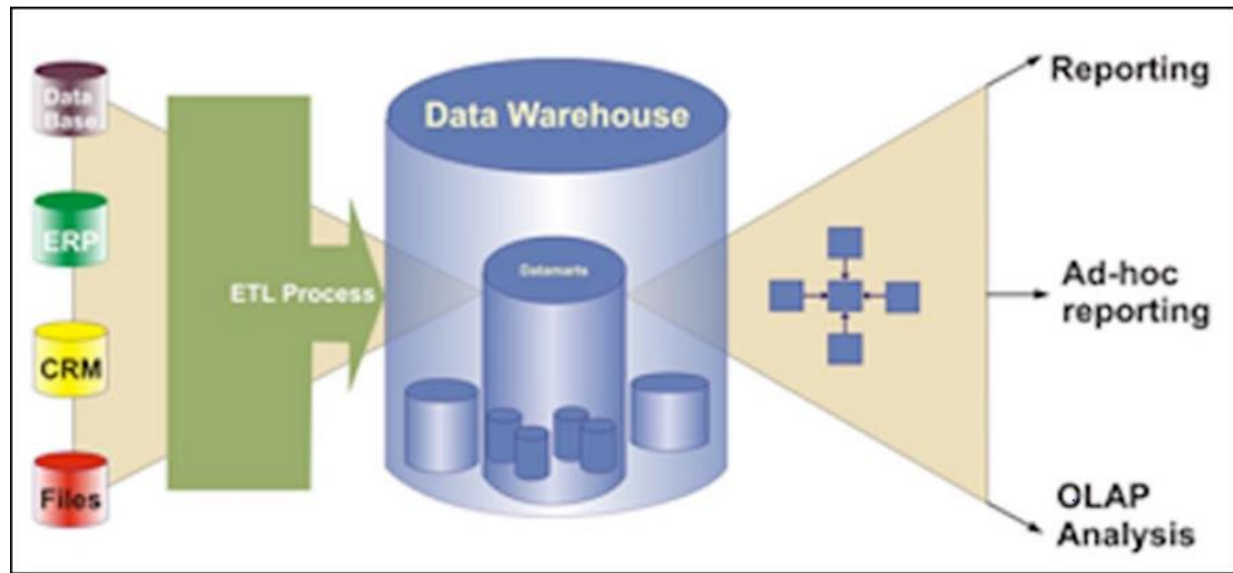
- **A crucial information:** What is the geographical area where majority of travelers live?

select zipcode, count(*) from flights group by zipcode;



What is a Data Warehouse?

- **A large store of data accumulated from a wide range of sources within an enterprise**
 - Used for reporting and data analysis to help **guide management make right decisions**
 - Central repositories of integrated data from one or more disparate sources



What is a Data Warehouse?

- A data warehouse, in general, is based on a **relational database**
 - designed for **query** and **analysis** rather than for transaction processing.
- It **usually contains historical data**
 - derived from transaction data
- It **separates analysis workload**
 - from transaction workload
- **Typically includes**
 - an ETL solution,
 - an OLAP engine, and
 - client analysis tools



What is Amazon Redshift

- Amazon's **relational data warehouse service** in AWS
- **Fully Managed**
 - Most **administrative tasks** are **fully automated**
 - Provisioning, configuration, backups, patching, replication
 - Continuous monitoring and recovery from failures
- **Petabyte Scale**
 - Optimized for datasets ranging from a few hundred gigabytes to a petabyte or more
 - 1PB = 1000 terabytes = 1000000 gigabytes
 - Think the size of **10 billion photos on FACEBOOK**



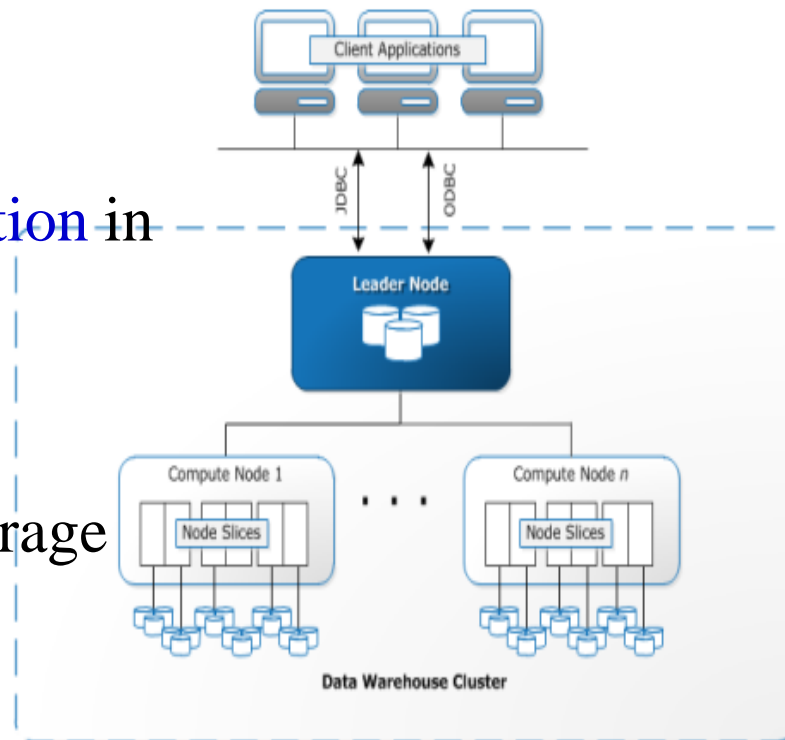
What is Amazon Redshift

- **Delivers fast query and I/O performance for virtually any size dataset using**
 - Columnar storage technology
 - Massively parallel processing (MPP) architecture
 - Parallelizes and distributes queries across multiple nodes
- **Designed for OLAP and BI applications**
 - ANSI SQL compliant
- **A powerful and cost-effective data warehouse solution**
 - “Pay as you go” pricing model
 - \$1000 per TB per year



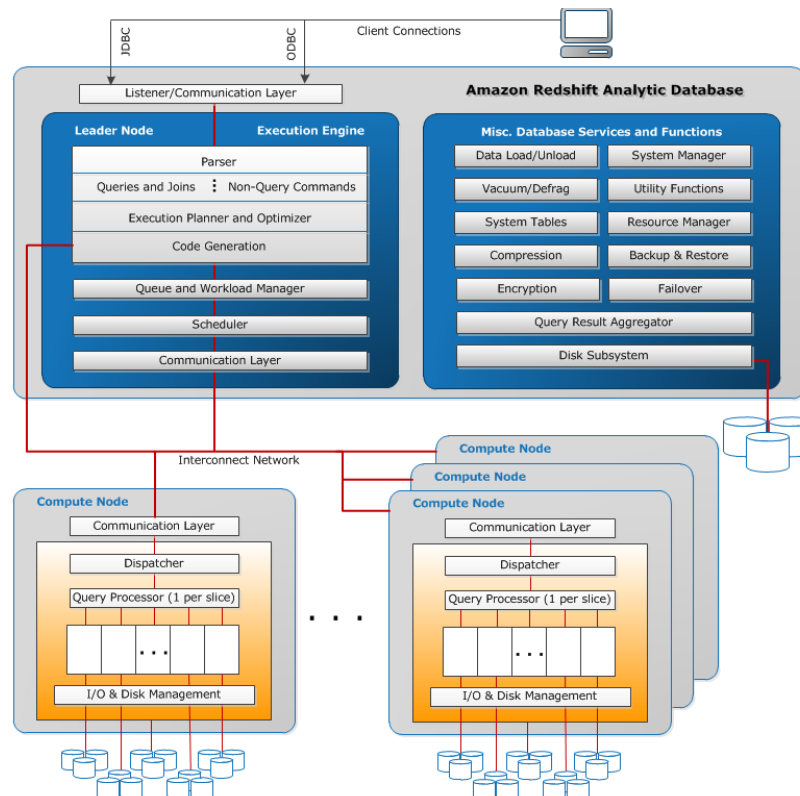
Redshift Architecture

- Redshift data warehouse is a **cluster of nodes** (computing resources)
- **Leader Node (One)**
 - Has SQL endpoint
 - For client connectivity
 - Stores metadata about the cluster
 - Coordinates parallel query execution in compute nodes
 - The **brain** behind Redshift cluster
- **Compute Nodes**
 - Stores data in local, **columnar** storage
 - Execute queries in parallel
 - Load, backup, restore



Redshift Architecture

- **Massively parallel processing architecture**
 - Redshift distributes workload to each node in the cluster and processes work in parallel

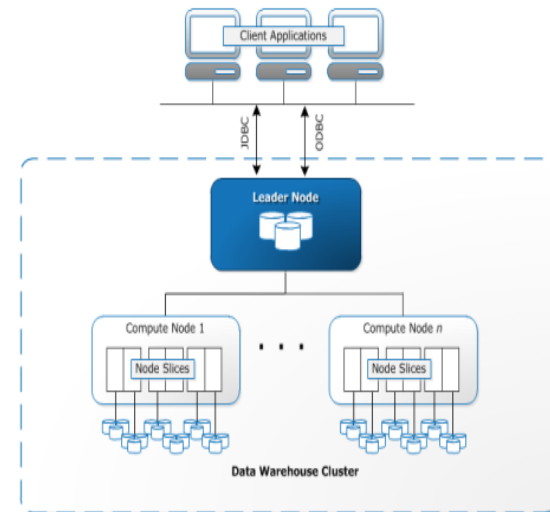


High throughput b/w components

- Direct Attached Storage
- 10 GigE mesh network

Hardware Platforms used by Redshift

- **Dense Storage (DS2) Nodes**
 - Based on **Hard Disk Drives (HDD)**
 - Can scale from 2TB to 2PB
 - Ideal for large data warehouses
- **Dense Compute (DC1) Nodes**
 - Use **Solid State Drive (SSD)**
 - Can scale from 160GB to 326TB
 - Ideal for high performance databases



Family	Node Type	Storage type	Storage per node	Maximum Storage
Dense Compute	dc1.large	SSD	0.16TB	32 nodes = 5.12TB
Dense Compute	dc1.8xlarge	SSD	2.56TB	128 nodes = 326TB
Dense Storage	ds2.xlarge	Magnetic	2TB	32 nodes = 64TB
Dense Storage	ds2.8xlarge	Magnetic	16TB	128 nodes = 2PB

- **Optimized for data processing**

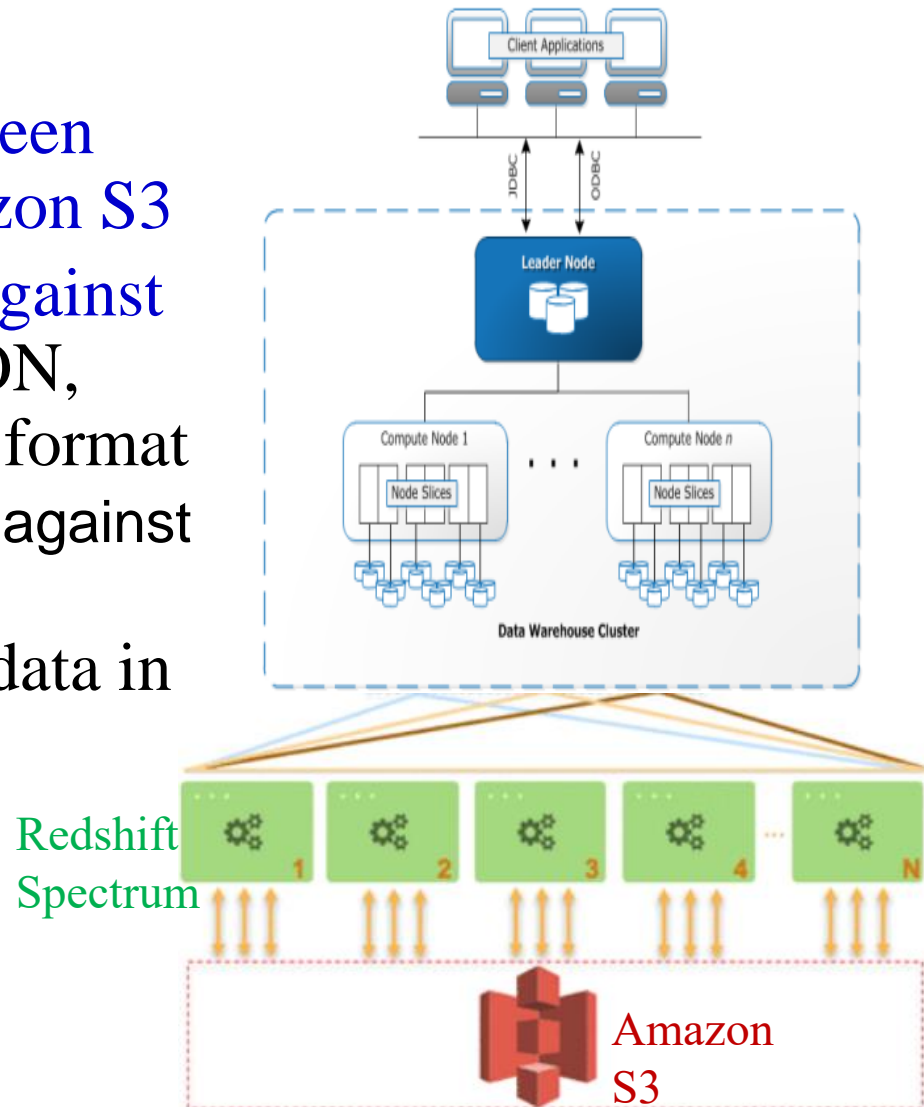
Redshift Spectrum

- An extension to Amazon Redshift
- Allows you to **query** and **retrieve** structured and semi-structured **data** from **files** in **Amazon S3**
 - without having to load the data into Amazon Redshift tables.
 - Much of the processing occurs in the Redshift Spectrum layer, and most of the data remains in Amazon S3.
 - Multiple clusters can concurrently query the same dataset in Amazon S3 without the need to make copies of the data for each cluster.
- Redshift Spectrum queries employ **massive parallelism** to execute very fast against large datasets.

Redshift Spectrum

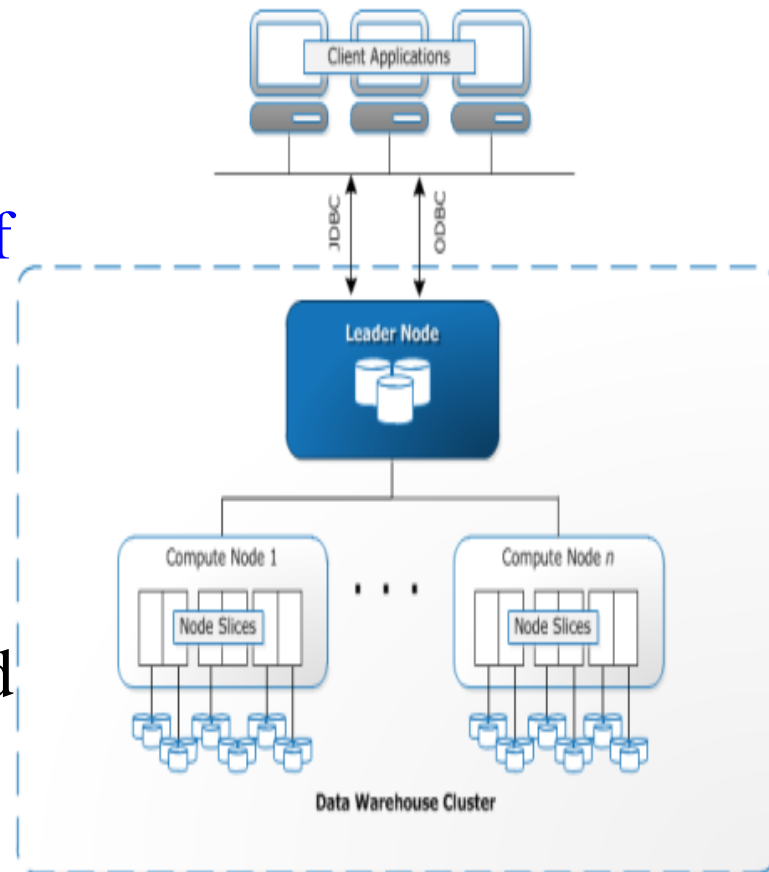
- **Redshift Spectrum**

- Spectrum layer lays **between** **compute nodes** and **Amazon S3**
- Executes **query directly against Amazon S3** stored in JSON, ORC, CSV, Parquet, etc. format
 - **Executes Redshift SQL** against those files **in S3**
- Can **join data in S3** with data in **Redshift** cluster



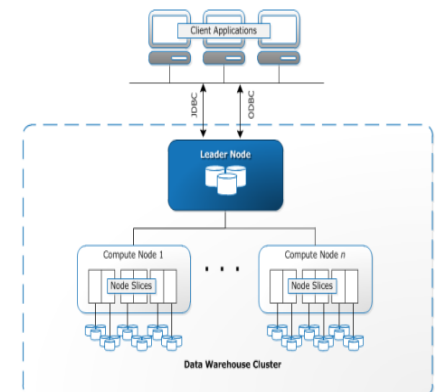
Provisioning Redshift Cluster

- **To provision a cluster:**
 - Log into the Redshift console
 - Specify the **type** and **number of nodes** that will make up the cluster
 - **Node type** determines the **storage size, memory, CPU**, and price of each node
 - Select a few configurations and launch the cluster
 - See an example later



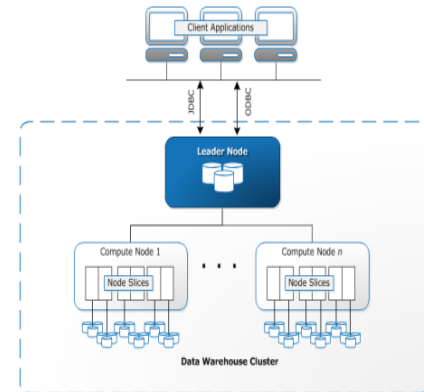
Scalability in Redshift cluster

- **Achieved by resizing the cluster**
 - Scale out or in by adding or removing nodes
 - Scale up or down by specifying a different node type
 - Or, you can do both
- **Resizing replaces the old cluster at the end of the resize operation**
 - The source cluster remains in read-only mode until the resize operation is complete
 - Resizing involves minimal downtime



Characteristics that makes Redshift unique

- **Variety of innovations to reduce disk I/O**
 - Columnar Storage
 - Data Compression
 - Zone Maps
 - Data Sorting
 - Sort Keys
 - Node Slices
 - Support for massive parallelism
 - Data Distribution
 - Dist Key
- **Data protection and Redshift security**



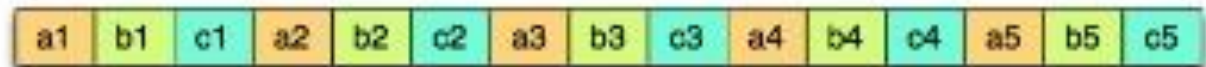
Columnar Storage

- **Redshift stores data on disk by columns**
 - rather than storing data values together for a whole row
- **For analytics queries, which typically reads only subset of columns, only scans blocks relevant to columns**
 - Results in massive reduction of disk I/O
 - Less data to load from disk

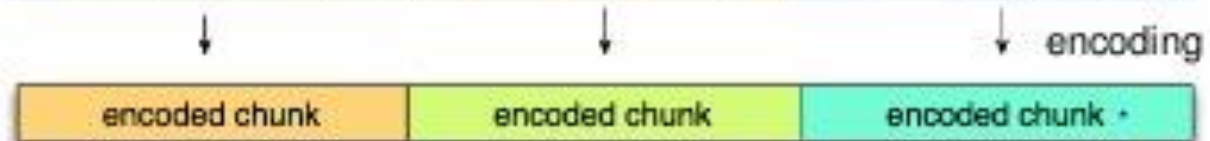
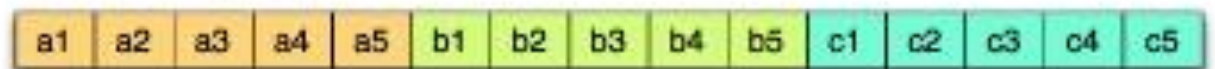
Logical table representation

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5

Row layout

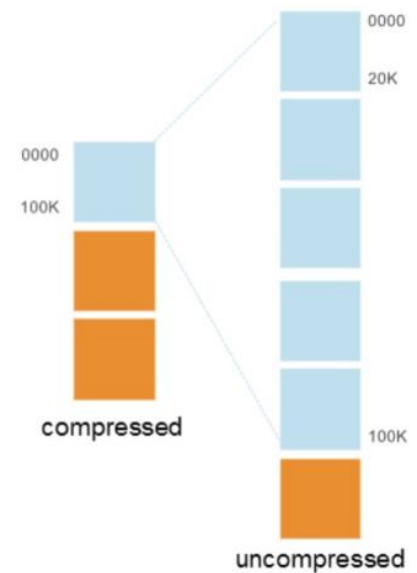


Column layout



Columnar Compression

- **Compression is a column-level operation** that reduces the size of the data when stored
 - Reduces storage requirements by 50 to 75%
 - Storing less data lowers the cost of ownership
 - Reduces disk I/O during query execution
 - Improves query performance
 - In Redshift, **compression** and **encoding** are used interchangeably



Columnar Compression

- Enables **effective compression** for each **data type per column** due to like data
 - Compression **encoding** is **specified** in **DDL** statement
 - As are sort key(s) and distribution style

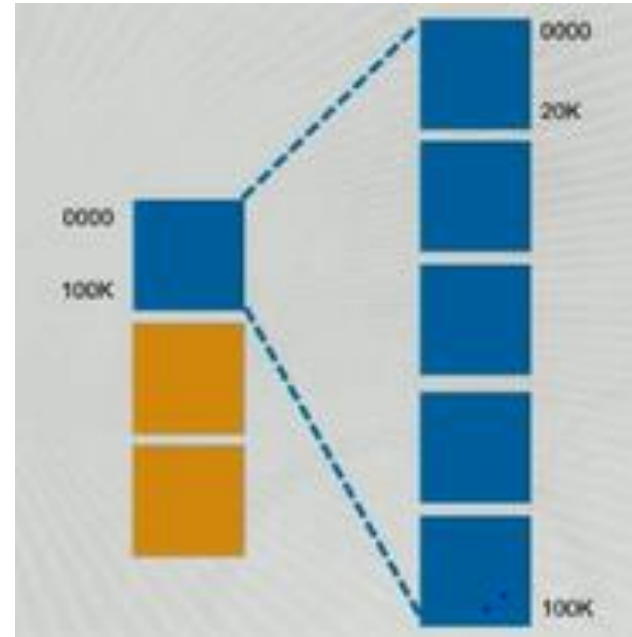
Compression encodings

- 📦 Raw (RAW)
- 📦 Byte-dictionary (BYTEDICT)
- 📦 Delta (DELTA / DELTA32K)
- 📦 Mostly (MOSTLY8 / MOSTLY16)
- 📦 Runlength (RUNLENGTH)
- 📦 Text (TEXT255 / TEXT32K)
- 📦 LZO

```
CREATE TABLE public.orders
(
    o_orderkey BIGINT NOT NULL ENCODE mostly32 DISTKEY,
    o_custkey BIGINT NOT NULL ENCODE mostly32,
    o_orderstatus CHAR(1) NOT NULL ENCODE lzo,
    o_totalprice NUMERIC (12, 2) NOT NULL ENCODE mostly32,
    o_orderdate DATE NOT NULL ENCODE delta,
    o_orderpriority CHAR(15) NOT NULL ENCODE bytedict,
    o_clerk CHAR(15) NOT NULL ENCODE lzo,
    o_shippriority INTEGER NOT NULL ENCODE runlength,
    o_comment VARCHAR(79) NOT NULL ENCODE text255
)
SORTKEY
(
    o_orderdate
);
```

Blocks

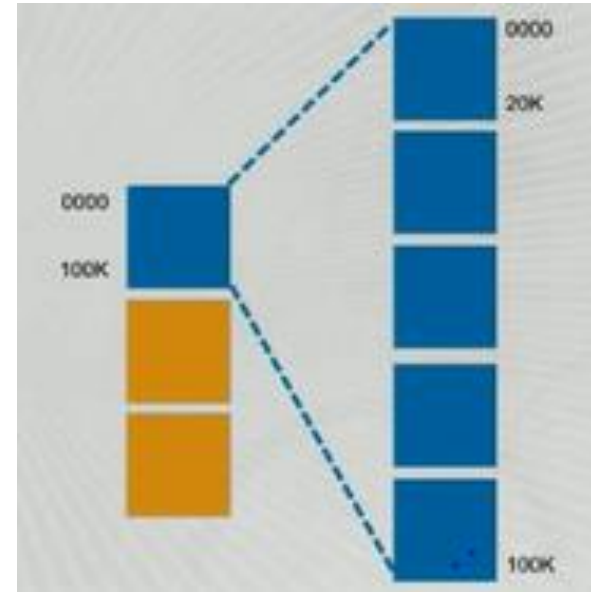
- **Blocks** are where the column data is stored in
- Typically, in **1 MB immutable** blocks
- Blocks are **individually encoded** with 1 of 11 encodings
- A full block can contain millions of values



Data
Blocks
(disk)

Zone Maps

- **In-memory block metadata**
- **Store min and max values for each block**
 - All blocks have zone maps
- **Allows to prune data (blocks) out at query time**
 - For queries with predicates on “where” clause, checks min-max values in memory before reading the disk
 - Exclude the non-relevant blocks
 - Minimize unnecessary I/O
- **An Indexing strategy**



Zone Map
(memory)

Data
Block
(disk)

Data Sorting

- Redshift **stores data in tables in sorted order based on sort keys**
 - Sorting makes **zone maps** more effective to skip over the data blocks to minimize I/Os
 - Sorting makes **pruning** much more effective

```
SELECT COUNT(*)  
FROM LOGS  
WHERE DATE =  
      '09-JUNE-2013'
```

Unsorted Table

READ	MIN: 01-JUNE-2013 MAX: 20-JUNE-2013
READ	MIN: 08-JUNE-2013 MAX: 30-JUNE-2013
	MIN: 12-JUNE-2013 MAX: 20-JUNE-2013
READ	MIN: 02-JUNE-2013 MAX: 25-JUNE-2013
READ	MIN: 06-JUNE-2013 MAX: 12-JUNE-2013

Sorted By Date

	MIN: 01-JUNE-2013 MAX: 06-JUNE-2013
READ	MIN: 07-JUNE-2013 MAX: 12-JUNE-2013
	MIN: 13-JUNE-2013 MAX: 18-JUNE-2013
	MIN: 19-JUNE-2013 MAX: 24-JUNE-2013
	MIN: 25-JUNE-2013 MAX: 30-JUNE-2013

Sort Keys

- **Picking a sort key is typically based on query patterns**
 - Also, on business requirements, and/or cardinality of data
 - Sort key can span more than one column
- **Single Column Sort Key**
 - Consists of a **single column** in the table
- **Compound Sort Keys**
 - Made up of **nested collection of columns**
 - Columns **specified** in their **order of priority**
 - Primary sort column is given more weight
 - Useful with JOINS, ORDER BY, GROUP BY operations
- **Interleaved Sort Keys**
 - Made up of **two or more columns**
 - Gives **equal weight to each column** in the sort key
 - **Order** in which columns are listed **does not matter**
 - Useful when don't know which column will be used in the query

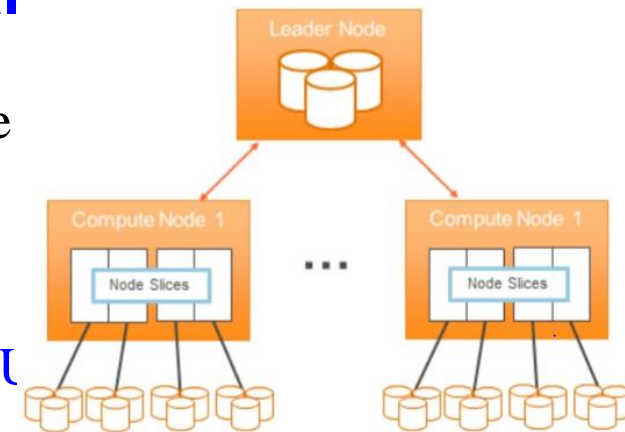
```
CREATE TABLE  
CUST_SALES_DT_SINGLE  
SORTKEY (C_CUSTKEY)  
AS SELECT * FROM  
CUST_SALES_DATE;
```

```
CREATE TABLE  
CUST_SALES_DT_COMPOUND  
COMPOUND SORTKEY  
(C_CUSTKEY, C_REGION,  
C_MKTSEGMENT, D_DATE)  
AS SELECT * FROM  
CUST_SALES_DATE;
```

```
CREATE TABLE  
CUST_SALES_DT_INTERLEAVED  
INTERLEAVED SORTKEY  
(C_CUSTKEY, C_REGION,  
C_MKTSEGMENT, D_DATE)  
AS SELECT * FROM  
CUST_SALES_DATE;
```

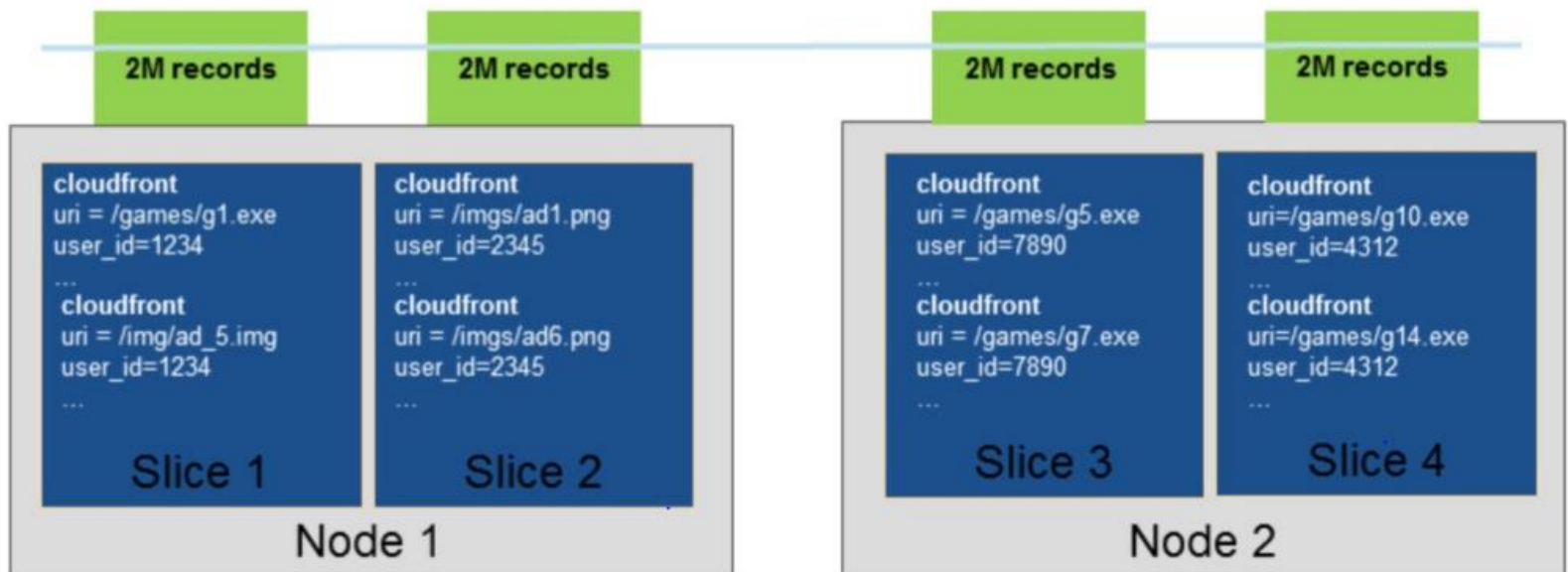
Node Slices

- A **slice** can be thought of as a “**virtual compute node**”
 - This is how parallelism is achieved in each of the Redshift cluster nodes
- **Compute Node is partitioned into Slices**
 - Each **slice** is allocated a portion of the node's **CPU memory**, and **disk space**
 - When data is written to Redshift, **the data is actually stored in slices**
 - A slice processes its own data
- **Support parallel query processing in compute nodes**
 - Each slice processes a piece of workload in parallel
 - Number of slices per node depends on the node size
 - A compute node has either 2, 16, 32 slices



Distribution Styles

- Used to **distribute data** (table rows) across **compute nodes and slices**
 - The goal is to distribute data evenly across the nodes
 - Minimize data movement: co-located joins and aggregates



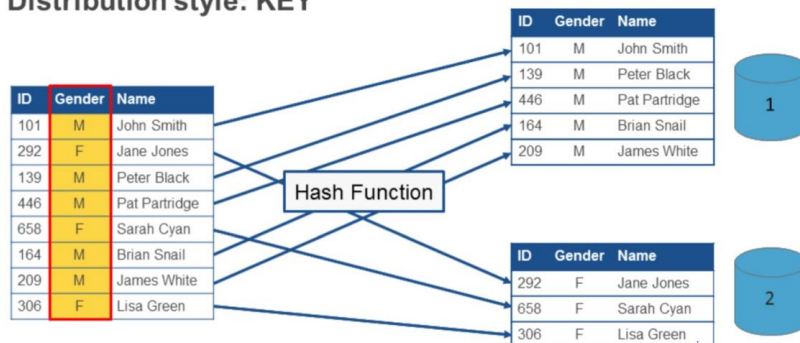
Evenly distributed data improves query performance.

Three Distribution Styles: Key, Even, and All

- **KEY** (hash function)

- Here a **column** is selected to be the **distribution key**
- Value of the **column** in the row is **hashed**, and the hash **corresponds** to one of the **slices** in the cluster
 - The same value goes to the same location (slice)
- Use this when the Key/column has high cardinality values and creates an even distribution of the data across all nodes
 - Useful in **query performance** in **table joins** and **group by**
 - Place **matching rows on the same node slice**

Distribution style: KEY



```
CREATE TABLE loft_deep_dive (  
  aid  INT      --audience_id  
  ,loc  CHAR(3)  --location  
  ,dt   DATE     --date  
) DISTSTYLE KEY DISTKEY (aid);
```

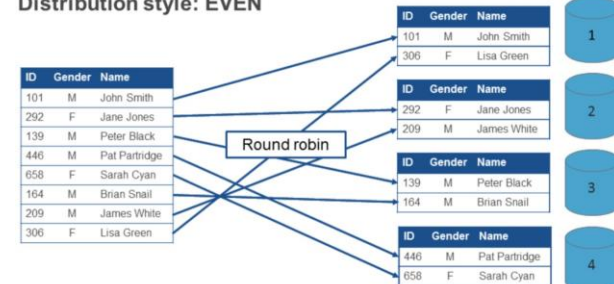
Three Distribution Styles

- **EVEN** (round robin – default choice)

- Rows distributed across the slices on compute nodes in a round-robin fashion

```
CREATE TABLE loft_deep_dive (  
  aid INT      --audience_id  
  ,loc CHAR(3) --location  
  ,dt DATE     --date  
) DISTSTYLE EVEN;
```

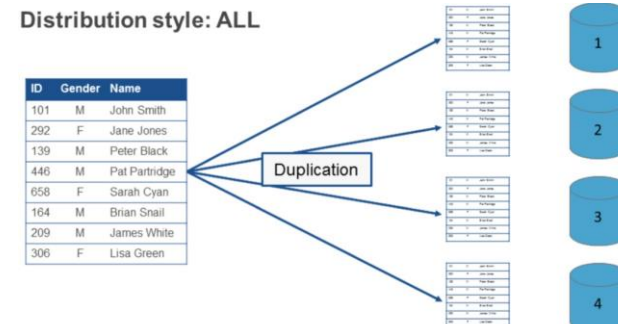
Distribution style: EVEN



- **ALL** (duplication)

- A copy of the entire table is distributed to every compute node
- Useful for reasonably sized data (2-3 millions rows)

Distribution style: ALL



Design Consideration: Column Properties

- **DISTKEY, SORTKEY, Compression Style (in that order) can influence performance significantly**
 - Can have an order of magnitude difference in performance
- **Distribution Keys**
 - A poor DISTKEY can introduce data skew and an unbalanced workload
 - A query completes only as fast as the slowest slice completes
- **Sort Keys**
 - Picking SORT Key(s) depends on business requirements and query patterns
 - Based on range or equality predicate (WHERE clause)
 - If you access recent data frequently, sort based on timestamp
- **Compression**
 - COPY automatically analyzes and compresses data when loading into empty table
 - ANALYZE COMPRESSION checks existing table and proposes optimal compression algorithm for each column
 - Changing column encoding requires a table rebuild

Analyzing Table Design

- Properly specifying **dist keys, sort keys, and compression encodings** can significantly improve storage, I/O, and **query performance**
- **AWS provides a SQL script** to identify tables where sort keys, dist keys, and column encodings are missing or performing poorly
- **table_inspector.sql**

Analyzing Table Design

- **table_inspector.sql**

```
SELECT SCHEMA schemaname,  
       "table" tablename,  
       table_id tableid,  
       size size in mb,  
       CASE  
         WHEN diststyle NOT IN ('EVEN','ALL') THEN 1  
         ELSE 0  
       END has_dist_key,  
       CASE  
         WHEN sortkey1 IS NOT NULL THEN 1  
         ELSE 0  
       END has_sort_key,  
       CASE  
         WHEN encoded = 'Y' THEN 1  
         ELSE 0  
       END has_col_encoding,  
       CAST(max_blocks_per_slice - min_blocks_per_slice AS FLOAT) /  
       GREATEST(NVL (min_blocks_per_slice,0)::int,1) ratio_skew_across_slices,  
       CAST(100*dist_slice AS FLOAT) / (SELECT COUNT(DISTINCT slice) FROM  
       stv_slices) pct_slices_populated  
FROM svv_table_info ti  
   JOIN (SELECT tbl,  
                MIN(c) min_blocks_per_slice,  
                MAX(c) max_blocks_per_slice,  
                COUNT(DISTINCT slice) dist_slice  
        FROM (SELECT b.tbl,  
                     b.slice,  
                     COUNT(*) AS c  
                FROM STV_BLOCKLIST b  
                GROUP BY b.tbl,  
                         b.slice)  
        WHERE tbl IN (SELECT table_id FROM svv_table_info)  
        GROUP BY tbl) ig ON ig.tbl = ti.table_id;
```


Analyzing Table Design

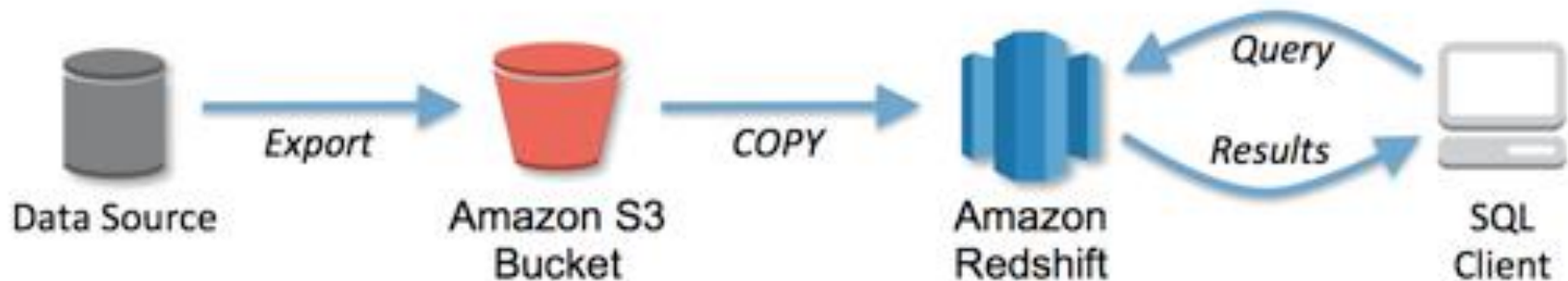
- table_inspector.sql

```
SELECT SCHEMA schemaname,
       "table" tablename,
       table_id tableid,
       size_in_mb,
       CASE
         WHEN diststyle NOT IN ('EVEN','ALL') THEN 1
         ELSE 0
       END has_dist_key,
       CASE
         WHEN sortkey1 IS NOT NULL THEN 1
         ELSE 0
       END has_sort_key,
       CASE
         WHEN encoded = 'Y' THEN 1
         ELSE 0
       END has_col_encoding,
       CAST(max_blocks_per_slice - min_blocks_per_slice AS FLOAT) /
       GREATEST((min_blocks_per_slice - 0)::float, 1) ratio_skew_across_slices,
       CAST(100 * dist_slice AS FLOAT) / (SELECT COUNT(DISTINCT slice) FROM
       stv_slices) pct_slices_populated
FROM stv_table_info tbl
JOIN (SELECT tbl,
            MIN(c) min_blocks_per_slice,
            MAX(c) max_blocks_per_slice,
            COUNT(DISTINCT slice) dist_slice
       FROM (SELECT b.tbl,
                    b.slice,
                    COUNT(*) AS c
              FROM stv_blocklist b
              GROUP BY b.tbl,
                     b.slice)
       WHERE tbl IN (SELECT table_id FROM stv_table_info)
       GROUP BY tbl) id ON id.tbl = tbl.table_id;
```

<u>schemaname</u>	<u>tablename</u>	<u>tableid</u>	<u>size_in_mb</u>	<u>has_dist_key</u>	<u>has_sort_key</u>	<u>has_col_encoding</u>	<u>ratio_skew_across_slices</u>	<u>pct_slices_populated</u>
public	category	100553	28	1	1	0	0	100
public	date	100555	44	1	1	0	0	100
public	event	100558	36	1	1	1	0	100
public	listing	100560	44	1	1	1	0	100
public	nation	100563	175	0	0	0	0	39.06
public	region	100566	30	0	0	0	0	7.81
public	sales	100562	52	1	1	0	0	100
public	skew1	100547	18978	0	0	0	.15	50
public	skew2	100548	353	1	0	0	0	1.56
public	venue	100551	32	1	1	0	0	100
public	users	100549	82	1	1	1	0	100
public	venue	100551	32	1	1	0	0	100

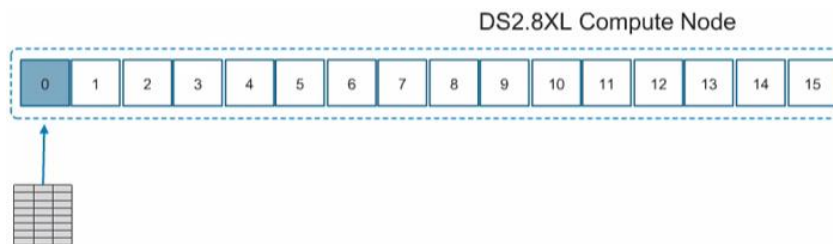
Loading data into Redshift

- COPY command** can **load multiple input files simultaneously** from **S3, DynamoDB, or EMR**

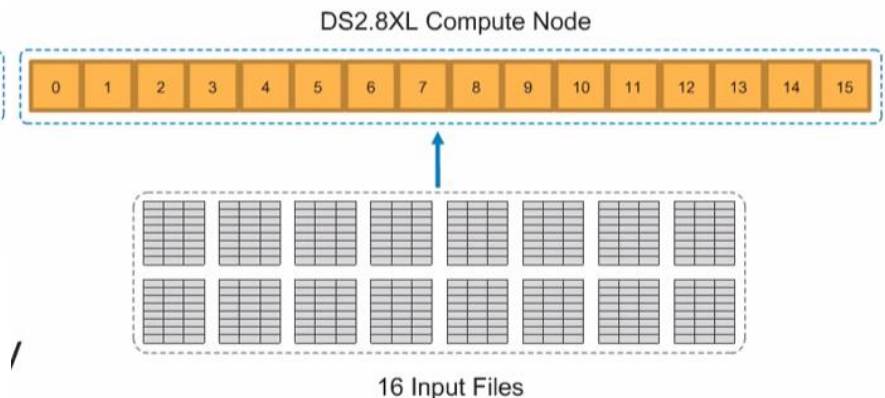


Loading data into Redshift

- **Node Slices can be loaded simultaneously**
 - To maximize throughput, have at least as many input files as number of slices
 - Single input file means only one slice is ingesting the data
 - Avoid single insert, update, delete operations (or use for small inserts/updates only)



Single **Large** Input File:
Effective throughput=6.25 MB/s



16 small input files (splitting the **Large** Input File): Effective throughput=100 MB/s

Loading Data into Redshift

- An example of COPY command:

```
copy customer
from 's3://mybucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

- Can use a **Manifest file** to control exactly what is loaded
 - An example of Manifest file called [cust.manifest](#)

```
{
  "entries": [
    {"url": "s3://mybucket-alpha/2013-10-04-custdata", "mandatory": true},
    {"url": "s3://mybucket-alpha/2013-10-05-custdata", "mandatory": true},
    {"url": "s3://mybucket-beta/2013-10-04-custdata", "mandatory": true},
    {"url": "s3://mybucket-beta/2013-10-05-custdata", "mandatory": true}
  ]
}
```

- An example of COPY command using the manifest file

```
copy customer
from 's3://mybucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

Data Protection in Redshift

- **Replicates all the data** in data warehouse when data is loaded
 - Replicates in compute nodes
- **Continuously backs up** all the data in Amazon S3
 - Backups are incremental and restores are quick
 - with frequently accessed data streamed first
- **Can asynchronously replicate your snapshots to Amazon S3 in another region**
 - Useful for disaster recovery

Redshift Security

- **Encryption at rest and in transit are built-in**
 - Includes data and backups
 - By default, takes care of key management
- **Option to manage your keys using your own hardware security modules**
 - AWS CloudHSM or
 - AWS KMS
- **SSL and Amazon VPC usage support built in**

Launching Redshift Cluster

Launching Redshift Cluster



Services ▾

Resource Groups ▾



Redshift dashboard

Clusters
Snapshots
Security
Parameter groups
Reserved nodes
Events
Connect client

Resources



You are using the following Amazon Redshift resources in the US West (Oregon) region (used):

Clusters (0)

[Increase cluster limit](#)

Snapshots (0)

[Manual \(0\)](#)

[Automated \(0\)](#)

Security

[Subnet groups \(0\)](#)

[HSM connections \(0\)](#)

[HSM certificates \(0\)](#)

Parameter groups (1)

[Reserved nodes \(0\)](#)

[Events \(0\)](#)

[Event subscriptions \(0\)](#)


Launch cluster

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse solution that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools.



[Launch cluster](#)

Note: Your cluster will launch in the US West (Oregon) region

Service health

Current Status		Details
	Amazon Redshift (Oregon)	Service is operating normally

Launching Redshift Cluster

 Services ▾ Resource Groups ▾ 

Redshift dashboard

Clusters

Snapshots

Security

Parameter groups

Reserved nodes

Events

Connect client

CLUSTER DETAILS

NODE CONFIGURATION

ADDITIONAL CONFIGURATION

REVIEW

Provide the details of your cluster. Fields marked with * are required.

Cluster identifier*

vinodd

This is the unique key that identifies a cluster. This parameter is stored as a lowercase string. (e.g. my-dw-instance)

Database name

vinodd

Optional. A default database named dev is created for the cluster. Optionally, specify a custom database name (e.g. mydb) to create an additional database.

Database port*

5439

Port number on which the database accepts connections.

Master user name*

master

Name of master user for your cluster. (e.g. awsuser)

Master user password*

.....

Password must contain 8 to 64 printable ASCII characters excluding: /, ", ', \, and @. It must contain 1 uppercase letter, 1 lowercase letter, and 1 number.

Confirm password*

.....

Confirm master user password

Cancel

Continue

41

Launching Redshift Cluster



Services ▾

Resource Groups ▾



Redshift dashboard

Clusters

Snapshots

Security

Parameter groups

Reserved nodes

Events

Connect client

CLUSTER DETAILS

NODE CONFIGURATION

ADDITIONAL CONFIGURATION

REVIEW

Choose a number of nodes and node type below. Number of Compute Nodes is required for multi-node clusters.



The ds2 node types replace the deprecated ds1 node types. The newer ds2 node types provide higher performance than ds1 at no extra cost. [Learn more.](#)

Node type

Specifies the compute, memory, storage, and I/O capacity of the cluster's nodes.

CPU 7 EC2 Compute Units (2 virtual cores) per node

Memory 15 GiB per node

Storage 160GB SSD storage per node

I/O performance Moderate

Cluster type

Number of compute nodes*

Maximum 32

Minimum 2

Compute nodes store your data and execute your queries. In addition to your compute nodes, a leader node will be added to your cluster, free of charge. The leader node is the access point for ODBC/JDBC and generates the query plans executed on the compute nodes.

Cancel

Previous

Continue

Launching Redshift Cluster

Services

Resource Groups

Redshift dashboard

Clusters

Snapshots

Security

Parameter groups

Reserved nodes

Events

Connect client

CLUSTER DETAILS

NODE CONFIGURATION

ADDITIONAL CONFIGURATION

REVIEW

Provide the optional additional configuration details below.

Cluster parameter group

default.redshift-1.0

 Parameter group to associate with this cluster.

Encrypt database ☒ None ☐ KMS ☐ HSM [Learn more about database encryption](#)

Configure networking options:

Choose a VPC

Default VPC (vpc-e0b0c385)

 The identifier of the VPC in which you want to create your cluster

Cluster subnet group

default

 Selected Cluster Subnet Group may limit the choice of Availability Zones

Publicly accessible ☒ Yes ☐ No Select Yes if you want the cluster to be accessible from the public internet. Select No if you want it to be accessible only from within your private VPC network

Choose a public IP address ☐ Yes ☒ No Select Yes if you want the cluster to have a public IP address that can be accessed from the public Internet, select No if you want the cluster to have a private IP address that can only be accessed from within the VPC.

Enhanced VPC Routing ☐ Yes ☒ No Select Yes if you want to enable Enhanced VPC Routing. [Learn more](#)

Availability zone

No Preference

 The EC2 Availability Zone that the cluster will be created in.

Optionally, associate your cluster with one or more security groups.

VPC security groups

default (sg-5f2b9c3b)

 List of VPC security groups to associate with this cluster.

Optionally, create a basic alarm for this cluster.

Create CloudWatch Alarm ☐ Yes ☒ No Create a CloudWatch alarm to monitor the disk usage of your cluster.

Optionally, associate up to 10 IAM roles with this cluster.

Available roles

User vdubey is not authorized to list roles 1ebff588-0282-11e7-8d18-f3708c959775. Please check with your administrator. (Service: AmazonRedshift; Management: Status Code: 403; Error: GetRole: AccessDeniedException: IP: 64.64.64.64: 44-7.0.118.43309: 959775)

43

Launching Redshift Cluster

Services

Resource Groups

Redshift dashboard

Clusters

Snapshots

Security

Parameter groups

Reserved nodes

Events

Connect client

CLUSTER DETAILS

NODE CONFIGURATION

ADDITIONAL CONFIGURATION

REVIEW

You are about to launch a cluster with the following specifications:

Cluster properties

These attributes specify the name of your cluster, what type of virtual hardware it will run on, how many nodes it will contain, and the availability zone in which it will be located.

Cluster identifier: vinodd

Node type: dc1.large

Number of compute nodes: 2 (plus a free leader node)

Availability zone: No preference

Database configuration

These properties specify the database name, port, and username you will use to connect to the database. The parameter group contains configuration values used by the database.

Database name: vinodd

Database port: 5439

Master user name: master

Cluster parameter group: default.redshift-1.0

Security, access, and encryption

These settings control whether your cluster will be created in an existing VPC to allow for simpler integration with other AWS Services, and the security groups which define access rules to your cluster.

Virtual private cloud: vpc-e0b0c385

Cluster subnet group:

Publicly accessible: Yes

Elastic IP: Not used

VPC security groups: sg-5f2b9c3b

Enhanced VPC Routing: No

Encrypt database: No

CloudWatch alarms

CloudWatch alarms are used to notify if metrics for your cluster are within a certain threshold. All recipients under the SNS topic specified for your alarm will receive notifications once an alarm is triggered.

Basic alarms will not be created for this cluster

⚠ Unless you are eligible for the free trial, you will start accruing charges as soon as your cluster is active.

Applicable charges:

The on-demand hourly rate for this cluster will be \$0.50 , or \$0.25 /node.If you have purchased reserved nodes in this region for this node type that are active, your costs will be discounted. Additional nodes will be billed at the on-demand rate. There is no charge for the leader node.

If you are eligible for a free trial, you will receive 750 hours of free usage for each month of the trial, applied across all running dc1.large nodes across all regions. Regardless of when you start your trial, you will receive two full months of free usage. Once your trial expires or your usage exceeds 750 hours/month, you can shut down your cluster, avoiding any charges, or keep it running at our standard [On-demand rate](#) .

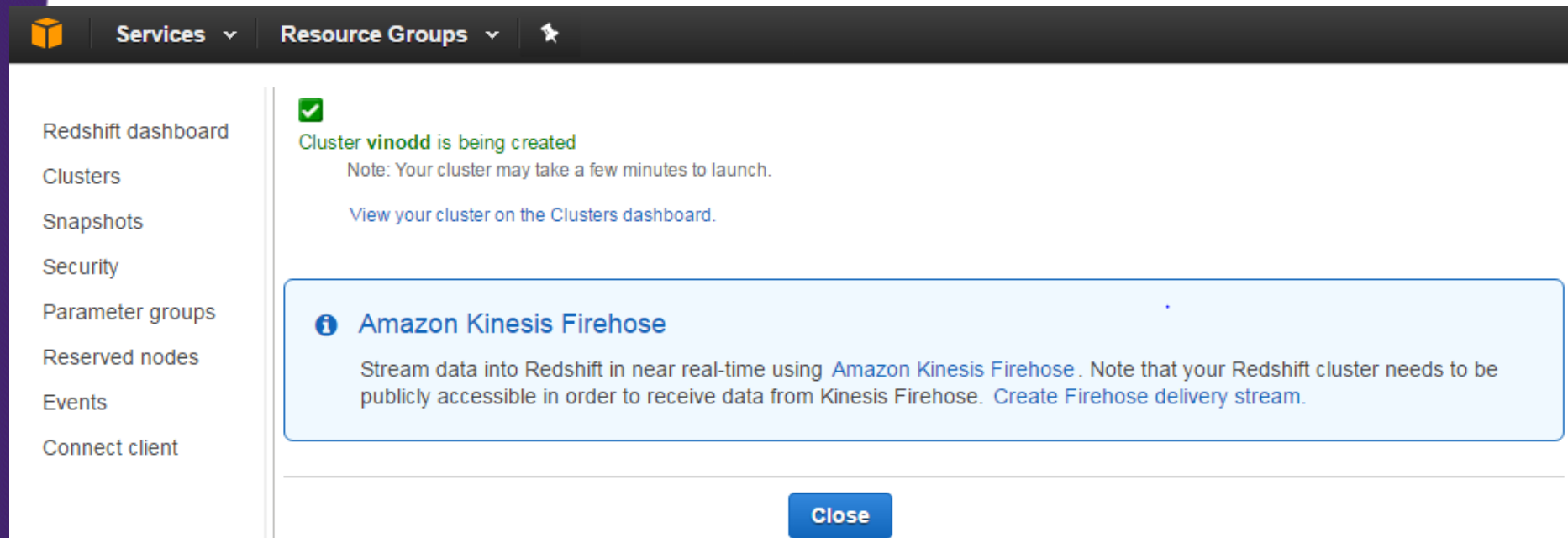
For more information, see [Amazon Redshift Free Trial FAQ](#) , [Amazon Redshift Pricing](#) , and [Reserved Nodes Documentation](#) .

Cancel

Previous

Launch cluster

Launching Redshift Cluster



The screenshot shows the AWS Management Console interface. At the top, there is a navigation bar with the AWS logo, 'Services' with a dropdown arrow, 'Resource Groups' with a dropdown arrow, and a star icon. On the left side, there is a sidebar menu with the following items: 'Redshift dashboard', 'Clusters', 'Snapshots', 'Security', 'Parameter groups', 'Reserved nodes', 'Events', and 'Connect client'. The main content area displays a green checkmark icon followed by the text 'Cluster **vinodd** is being created'. Below this, a note states: 'Note: Your cluster may take a few minutes to launch.' and a link says 'View your cluster on the Clusters dashboard.' Below this notification is a light blue box with a blue information icon and the title 'Amazon Kinesis Firehose'. The text inside the box reads: 'Stream data into Redshift in near real-time using [Amazon Kinesis Firehose](#). Note that your Redshift cluster needs to be publicly accessible in order to receive data from Kinesis Firehose. [Create Firehose delivery stream](#).' At the bottom right of the notification area, there is a blue button labeled 'Close'.

Services ▾ Resource Groups ▾ ☆

Redshift dashboard
Clusters
Snapshots
Security
Parameter groups
Reserved nodes
Events
Connect client

✓
Cluster **vinodd** is being created
Note: Your cluster may take a few minutes to launch.
[View your cluster on the Clusters dashboard.](#)

i Amazon Kinesis Firehose
Stream data into Redshift in near real-time using [Amazon Kinesis Firehose](#). Note that your Redshift cluster needs to be publicly accessible in order to receive data from Kinesis Firehose. [Create Firehose delivery stream](#).

Close

Launching a Redshift Cluster

The screenshot shows the AWS Redshift console interface. At the top, there's a navigation bar with 'Services' and 'Resource Groups' dropdowns. On the left, a sidebar lists navigation options: 'Redshift dashboard', 'Clusters' (highlighted with an orange bar), 'Snapshots', 'Security', 'Parameter groups', 'Reserved nodes', 'Events', and 'Connect client'. The main content area is titled 'Clusters' and contains three buttons: 'Launch Cluster' (blue), 'Manage Tags', and 'Manage IAM roles'. Below these buttons is a table with the following columns: a checkbox, a play button, a magnifying glass icon, 'Cluster', 'Cluster Status', and 'DB Health'. One cluster is listed with the name 'vinodd', a status of 'creating', and a health of 'unknown'.

			Cluster	Cluster Status	DB Health
<input type="checkbox"/>	▶	🔍	vinodd	creating	unknown

Cluster: **vinodd** ▾

Configuration

Status

Performance

Queries

Loads

Table restore

Cluster: vinodd

Cluster ▾

Database ▾

Backup ▾

Endpoint vinodd.cgmlfcjwrpzu.us-west-2.redshift.amazonaws.com:5439 (**No Inbound Permissions**) ⚠

Cluster Properties

Cluster Name	vinodd
Cluster Type	Multi Node
Node Type	dc1.large
Nodes	2
Zone	us-west-2c
Created Time	March 6, 2017 at 10:43:55 AM UTC-5
Cluster Version	1.0.1192
VPC ID	vpc-e0b0c385 (View VPCs)
Cluster Subnet Group	default
VPC security groups	default (sg-5f2b9c3b) (active)
Cluster Parameter Group	default.redshift-1.0 (in-sync)
Enhanced VPC Routing	No

Cluster Status

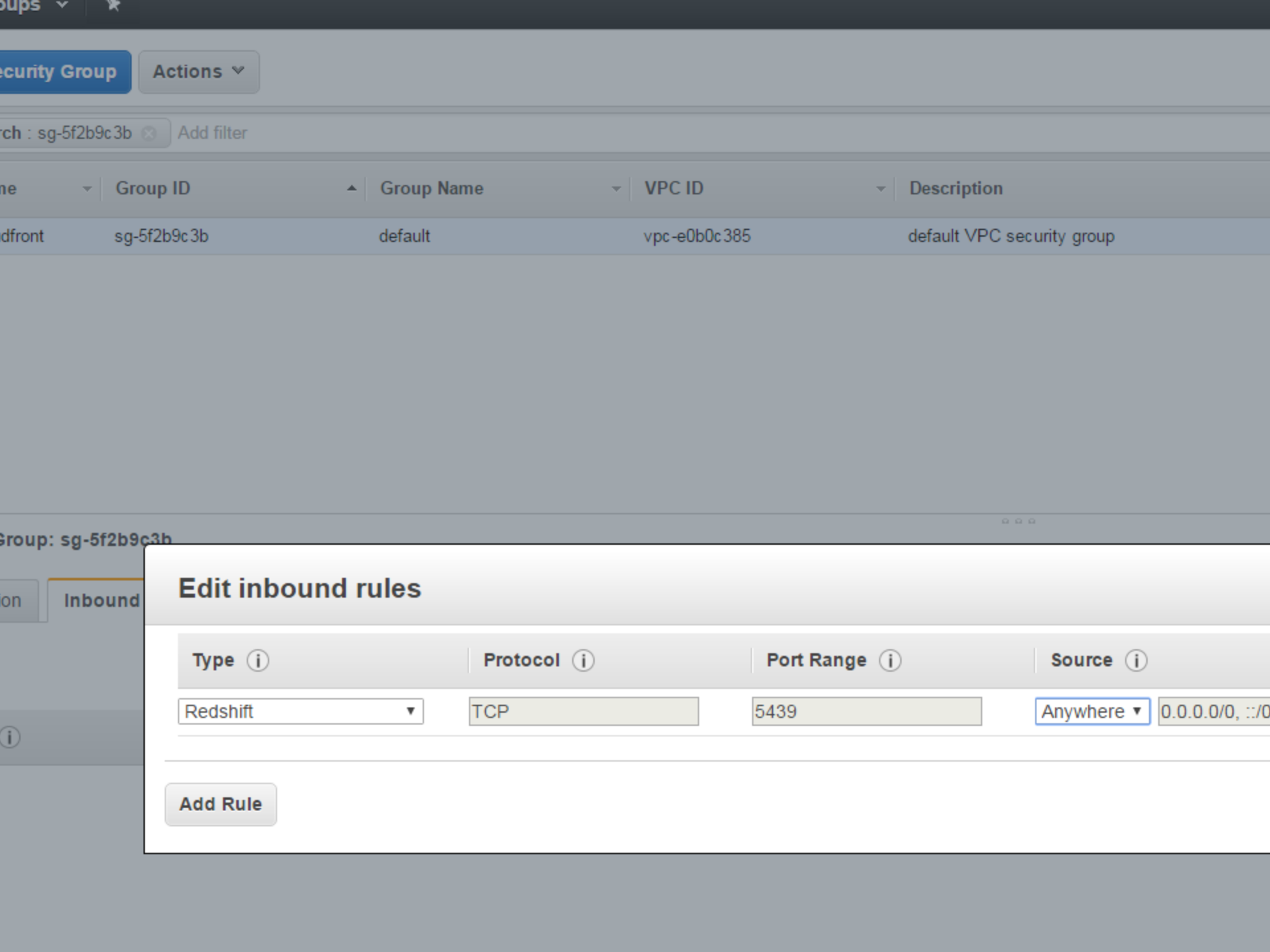
Cluster Status	available
Database Health	healthy
In Maintenance Mode	no
Parameter Group Apply Status	in-sync
Pending Modified Values	None

Cluster Database Properties

Port	5439
Publicly Accessible	Yes
Database Name	vinodd
Master Username	master
Encrypted	No

Backup, Audit Logging, and Maintenance

Automated Snapshot Retention Period	1
Cross-Region Snapshots Enabled	No
Audit Logging Enabled	No
Maintenance Window	wed:11:00-wed:11:00
Allow Version Upgrade	Yes



Security Group

Actions

Search : sg-5f2b9c3b Add filter

Group ID	Group Name	VPC ID	Description
sg-5f2b9c3b	default	vpc-e0b0c385	default VPC security group

Group: sg-5f2b9c3b

Inbound

Edit inbound rules

Type

Protocol

Port Range

Source

Redshift

TCP

5439

Anywhere

0.0.0.0/0, ::/0


Add Rule


Clusters

Launch Cluster

Manage Tags

Manage IAM roles

	Cluster	Cluster Status	DB Health
▼ 	vinodd	available	healthy

Endpoint vinodd.cgmlfcjwrpzu.us-west-2.redshift.amazonaws.com:5439 (**authorized**) 

Cluster Properties

Cluster Name	vinodd
Node Type	dc1.large
Nodes	2
Zone	us-west-2c
Cluster Parameter Group	default.redshift-1.0 (in-sync)
Enhanced VPC Routing	No

Cluster Database Properties

Port	5439
Database Name	vinodd
Master Username	master
Encrypted	No

Cluster Status

Cluster
Database
In Maintenance
Parameter Group Apply
Pending Modified V

Backup, Audit Logging, &
Automated Snapshot Re
Cross-Region Snap
Audit Log
Mainten
Allow Ver

Tags

You have not created any tags. Please add tags using the **Manage Tags** button above.

Launching a Redshift Cluster

Capacity Details

Current Node Type	dc1.large
CPU	7 EC2 Compute Units (2 virtual cores) per node
Memory	15 GiB per node
Storage	160GB SSD storage per node
I/O Performance	Moderate
Platform	64-bit

SSH ingestion settings

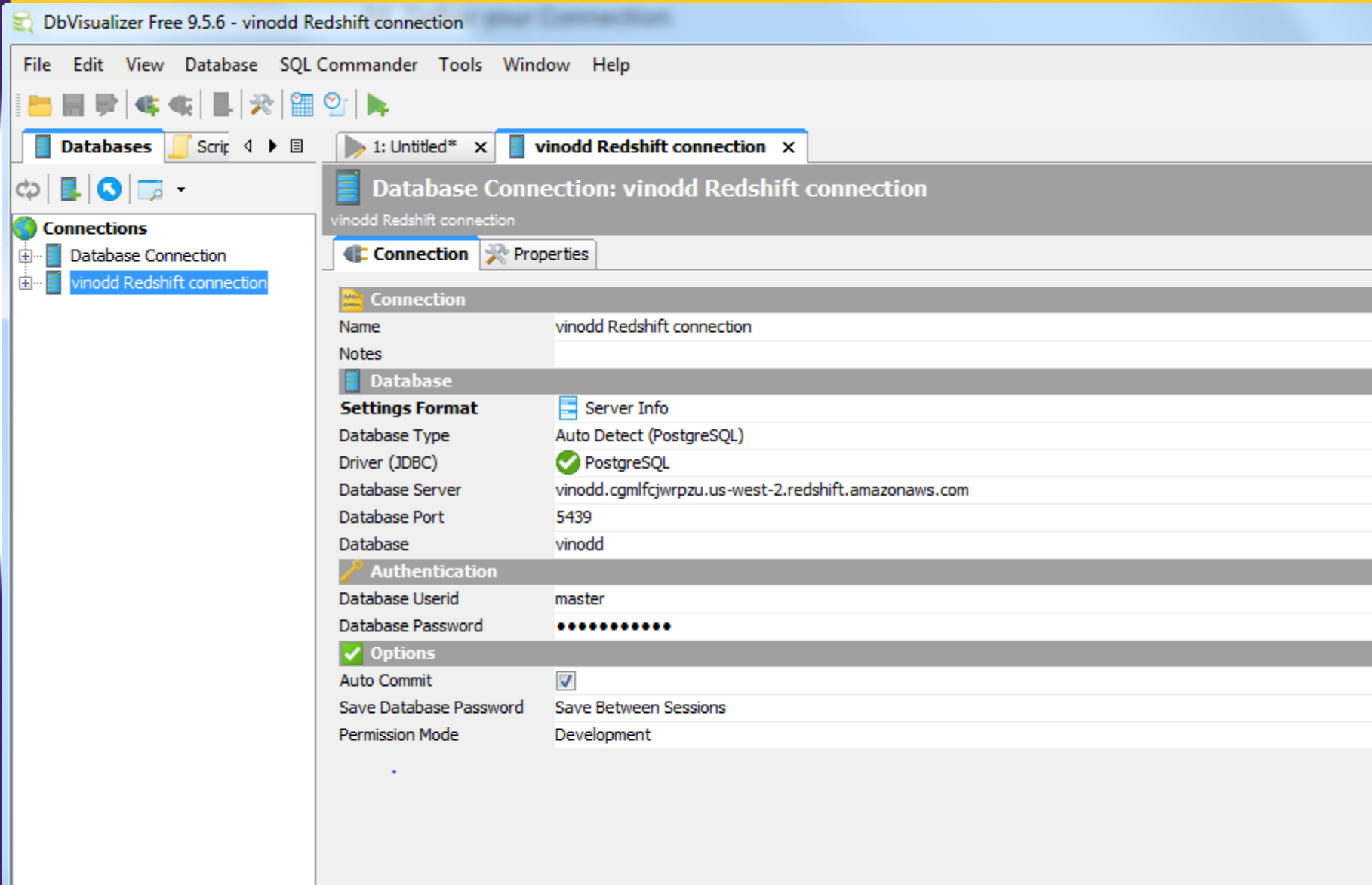
Cluster public key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQACdHxCsZGcwHySvJ/h32d
jXKqU41XYt1Z0ChTPvWYxe4TueOVmuydqiZcd9pbXr4Ag6p2Gw
8yrFjc+ig1gYK6CV6FWZxxxz/otNYZ286dTI7sOntK7VS8VHh6
Kozt6w1Qa4YQ0Z5F30D4e8bh6mluG0yLegFIPGl+P0l+IFAExf
2AIUi5EfdhACEIXPp0yo3+azBZnpgYZliDuOyx10q3LEdDat2G
SEw+rWdj4yee2arLBqK/5+pZZb1Ud03bvaYiD9ucnQP4Z5aumg
HAJ2+zkp5lX79QjVKew7InLkbRxe08eM1jfI+fn2kjibr0klHg
3F+Y4olBcjrjgSFQY3I/xz Amazon-Redshift
```

Node IP addresses:

Node	Public IP	Private IP
Compute-0	35.164.78.240	172.31.7.102
Leader	52.24.125.80	172.31.1.136
Compute-1	35.163.154.58	172.31.4.110

Connecting to Redshift Cluster Using DbVisualiser





Databases | Script | 1: Untitled* x



Connections

- Database Connection
- vinodd Redshift connection**
- vinodd (Default)



Database Connection

☐ Sticky Database

vinodd Redshift connection

vinodd

```

1 CREATE TABLE flights (
2   year          smallint,
3   month         smallint,
4   day           smallint,
5   carrier       varchar(80) DISTKEY,
6   origin        char(3),
7   dest          char(3),
8   aircraft_code char(3),
9   miles         int,
10  departures    int,
11  minutes       int,
12  seats         int,
13  passengers    int,
14  freight_pounds int
15 );

```

15:3 [358] INS



Log

☒ Preprocess script ☒ Log to GUI ☐ Log to File

... Physical database connection acquired for: vinodd Redshift connection/vinodd

11:30:24 [CREATE - 0 rows, 0.530 secs] Command processed. No rows were affected

... 1 statement(s) executed, 0 rows affected, exec/fetch time: 0.530/0.000 sec [1 successful, 0 errors]

Copying data from S3 to Redshift Cluster

DbVisualizer Free 9.5.6 - Untitled*

File Edit View Database SQL Commander Tools Window Help

Databases Script 1: Untitled* x

Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)

Database Connection: vinodd Redshift connection Sticky Database: vinodd Schema: public

```
1 COPY flights
2 FROM 's3://us-west-2-aws-training/awsu-spl/sp117-redshift/static/data/flights-usa'
3 CREDENTIALS 'aws_access_key_id=AKIAJQJX46P53YUBKKTQ;aws_secret_access_key=PPHEGmvITdNppXbMr/M5yGazPpqupa5qWjPRG0o'
4 GZIP
5 DELIMITER ','
6 REMOVEQUOTES;
```

6:14 [246] INS

Log

☒ Preprocess script ☒ Log to GUI ☐ Log to File

11:38:31 [COPY - 0 rows, 163.011 secs] Command processed. No rows were affected
SQL State: 00000 --- Load into table 'flights' completed, 96825753 record(s) loaded successfully.
... 1 statement(s) executed, 0 rows affected, exec/fetch time: 163.011/0.000 sec [1 successful, 1 SQL warnings, 0 errors]

Querying Redshift Cluster

The screenshot shows the DbVisualizer Free 9.5.6 interface. The 'Connections' tree on the left lists 'Database Connection', 'vinodd Redshift connection' (selected), and 'vinodd (Default)'. The main query editor displays the SQL statement: `1 SELECT COUNT(*) FROM flights;`. The 'Database Connection' dropdown is set to 'vinodd Redshift connection' and the 'Sticky Database' checkbox is unchecked. The status bar at the bottom indicates the query is executed, showing '1:30 [30] INS' and a 'Log' button. A results pane at the bottom right shows a table with one column 'count' and one row with the value '96825753'.

	count
1	96825753

DbVisualizer Free 9.5.6 - Untitled*

File Edit View Database SQL Commander Tools Window Help

Databases Script

Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)

1: Untitled* x

Database Connection: vinodd Redshift connection Sticky Database: vinodd Schema: public

```
1 SELECT carrier,SUM (departures) FROM flights GROUP BY carrier ORDER BY 2 DESC LIMIT 10;
```

1:88 [88] INS

Log 1: flights [10] x

*	carrier	sum
1	Southwest Airlines Co.	19846786
2	Delta Air Lines Inc.	17480331
3	American Airlines Inc.	15468460
4	United Air Lines Inc.	13402737
5	Northwest Airlines Inc.	9522027
6	Continental Air Lines Inc.	7929064
7	US Airways Inc.	7865177
8	American Eagle Airlines Inc.	6687299
9	ExpressJet Airlines Inc.	6270410
10	USAir	6052546

File Edit View Database SQL Commander Tools Window Help



Databases Script 1: Untitled* x



Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)



Database Connection

☐ Sticky Database

Schema

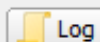
vinodd Redshift connection

vinodd

public

```
1 SELECT carrier,SUM (passengers) FROM flights GROUP BY carrier ORDER BY 2 DESC LIMIT 10;
```

1:31 [31] INS



Log

1: flights [10] x



*	carrier	sum
1	Delta Air Lines Inc.	1894041955
2	Southwest Airlines Co.	1812013508
3	American Airlines Inc.	1599088928
4	United Air Lines Inc.	1396240286
5	Northwest Airlines Inc.	846994753
6	Continental Air Lines Inc.	751752616
7	US Airways Inc.	738513678
8	USAir	406251028
9	America West Airlines Inc.	327787337
10	Alaska Airlines Inc.	290615980

File Edit View Database SQL Commander Tools Window Help



Databases Scrip < > □



Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)

1: Untitled* x



Database Connection

☐ Sticky Database

Schema

vinodd Redshift connection

vinodd

public

1 SELECT carrier,SUM (miles) FROM flights GROUP BY carrier ORDER BY 2 DESC LIMIT 10;

1:83 [83] INS



Log

1: flights [10] x



*	carrier	sum
1	United Air Lines Inc.	7478582456
2	Delta Air Lines Inc.	7356255471
3	American Airlines Inc.	6033117399
4	Southwest Airlines Co.	5326561689
5	Continental Air Lines Inc.	4454443574
6	Northwest Airlines Inc.	4277614432
7	US Airways Inc.	3193773517
8	USAir	2178906790
9	America West Airlines Inc.	1713042483
10	Federal Express Corporation	1514604035



Databases Script



Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)

1: Untitled* x



Database Connection

Sticky Database

Schema

vinodd Redshift connection

vinodd

public

```

1 SELECT
2   aircraft,
3   SUM(departures) AS trips
4 FROM flights
5 JOIN aircraft using (aircraft_code)
6 GROUP BY aircraft
7 ORDER BY trips DESC
8 LIMIT 10;

```

8:10 [143] INS



1: flights [10] x



*	aircraft	trips
1	Boeing 737-300	19632153
2	McDonnell Douglas DC9 Super 80/MD81/82/83/88	18608868
3	Canadair RJ-200ER /RJ-440	12062704
4	Boeing 757-200	10768257
5	Boeing 727-200/231A	9188041
6	Embraer-145	9184729
7	Boeing 737-100/200	8567467
8	Boeing 737-700/700LR	7550737
9	McDonnell Douglas DC-9-30	7105295
10	Airbus Industrie A320-100/200	6664803

DbVisualizer Free 9.5.6 - Untitled*

File Edit View Database SQL Commander Tools Window Help

Databases Script 1: Untitled* x

Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)

Database Connection vinodd Redshift connection Sticky Database Schema public

1 ANALYZE COMPRESSION flights;

1:29 [29] INS

Log 1: ANALYZE COMPRESSION flights [13] x

*	Table	Column	Encoding	Est_reduction_pct
1	flights	year	lzo	0.00
2	flights	month	lzo	0.00
3	flights	day	lzo	0.00
4	flights	carrier	lzo	0.00
5	flights	origin	lzo	0.00
6	flights	dest	lzo	0.00
7	flights	aircraft_code	lzo	0.00
8	flights	miles	lzo	0.00
9	flights	departures	lzo	0.00
10	flights	minutes	lzo	0.00
11	flights	seats	lzo	0.00
12	flights	passengers	delta	0.00
13	flights	freight_pounds	delta	0.00



Connections

- Database Connection
- vinodd Redshift connection
- vinodd (Default)

Database Connection

☐ Sticky Database

vinodd Redshift connection

vinodd

```
1 SELECT
2   owner AS node,
3   diskno,
4   used,
5   capacity,
6   used/capacity::numeric * 100 as percent_used
7 FROM stv_partitions
8 WHERE host = node
9 ORDER BY 1, 2;
```

9:15 [156] INS

Log 1: stv_partitions [2] x



*	node	diskno	used	capacity	percent_used
1	0	0	1016	190633	0.5329612396594503500
2	1	0	799	190633	0.4191299512676189300

▼ SQL

```
COPY flights FROM  
  's3://us-west-2-aws-training/awsu-spl/spl17-redshift/static/data/flights-usa' CREDENTIALS '' GZIP  
DELIMITER ',' REMOVEQUOTES
```

▼ Query Execution Details

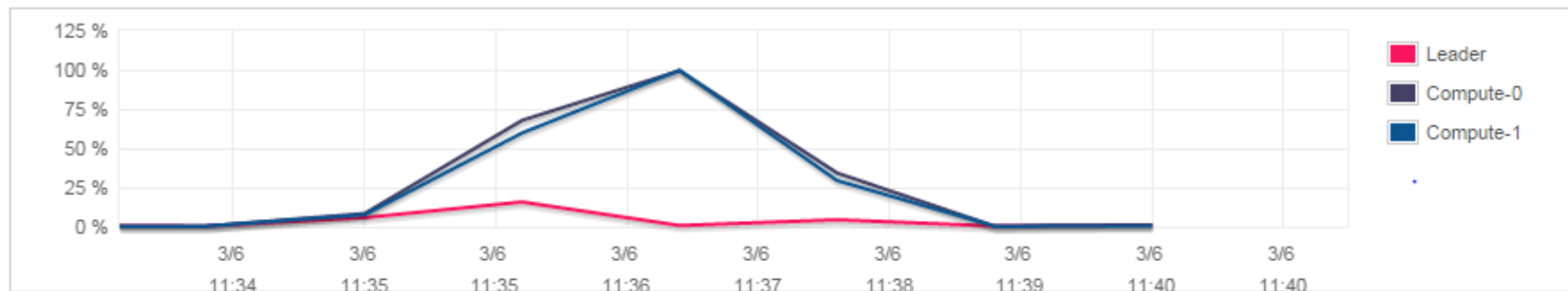
▼ Loaded Files

Filename

Cluster Performance During Query Execution

Cluster performance is shown from 3 minutes prior to query execution through 3 minutes after query completion.

CPU utilization



Cluster: **vinodd** ▾

Configuration

Status

Performance

Queries

Loads

Table restore

Terminate Query

Filter: Last 24 Hours ▾



Search...



<input type="checkbox"/>		Query ▾	Run time ▾	Start time ▾	Status ▾	User ▾	SQL
<input type="checkbox"/>		321	8.64s	March 6, 2017 at 12:10:17 PM UTC-5	completed	master	SELECT name, count(*) FROM stv_blocklist JOIN (SE
<input type="checkbox"/>		318	3.96s	March 6, 2017 at 12:08:42 PM UTC-5	completed	master	SELECT owner AS node, diskno, used, capacity, use
<input type="checkbox"/>		311	9.17s	March 6, 2017 at 12:07:16 PM UTC-5	completed	master	CREATE TABLE vegas_flights DISTKEY (origin) SORTK
<input type="checkbox"/>		308	1.02s	March 6, 2017 at 12:06:26 PM UTC-5	completed	master	padb_fetch_sample: select * from airports
<input type="checkbox"/>		307	4ms	March 6, 2017 at 12:06:26 PM UTC-5	completed	master	padb_fetch_sample: select count(*) from airports
<input type="checkbox"/>		306	1.58s	March 6, 2017 at 12:06:25 PM UTC-5	completed	master	COPY airports FROM 's3://us-west-2-aws-training/a
<input type="checkbox"/>		305	40ms	March 6, 2017 at 12:06:25 PM UTC-5	completed	master	analyze compression phase 2
<input type="checkbox"/>		304	11ms	March 6, 2017 at 12:06:25 PM UTC-5	completed	master	analyze compression phase 1
<input type="checkbox"/>		303	37ms	March 6, 2017 at 12:06:25 PM UTC-5	completed	master	analyze compression phase 2
<input type="checkbox"/>		302	6ms	March 6, 2017 at 12:06:25 PM UTC-5	completed	master	analyze compression phase 1
<input type="checkbox"/>		301	1.08s	March 6, 2017 at 12:06:24 PM UTC-5	completed	master	COPY ANALYZE airports
<input type="checkbox"/>		290	39ms	March 6, 2017 at 12:02:06 PM UTC-5	completed	master	analyze compression phase 2
<input type="checkbox"/>		289	2.07s	March 6, 2017 at 12:02:04 PM UTC-5	completed	master	analyze compression phase 1
<input type="checkbox"/>		288	20ms	March 6, 2017 at 12:02:04 PM UTC-5	completed	master	analyze compression phase 1



Redshift dashboard

Clusters

Snapshots

Security

Parameter groups

Reserved nodes

Events

Connect client

Cluster: **vinodd** ▾

Configuration

Status

Performance

Queries

Loads

Table restore

Time Range: **Last Hour** ▾Period: **5 Minutes** ▾Statistic: **Average** ▾

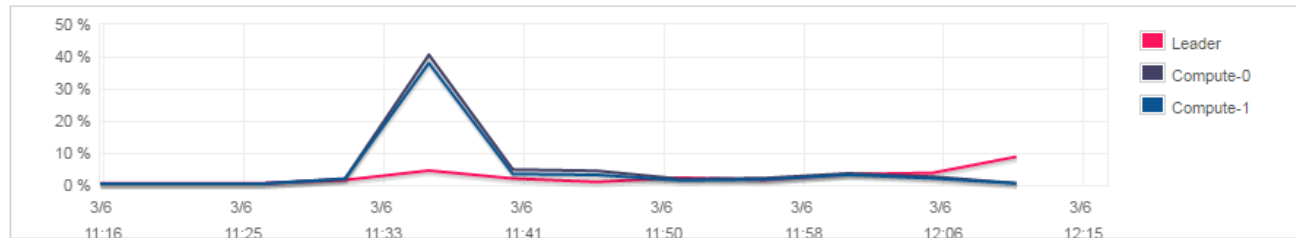
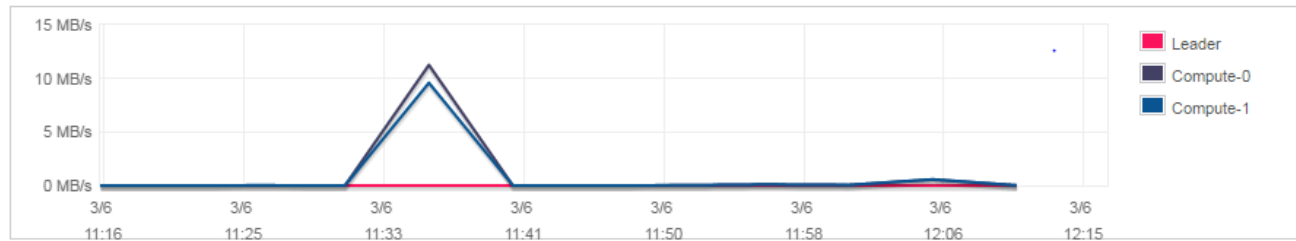
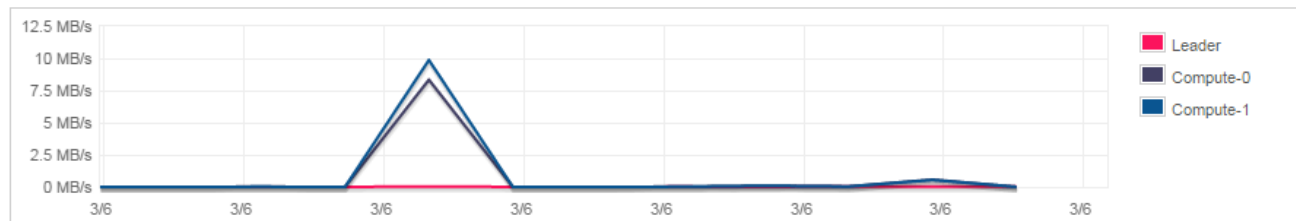
All Metrics ▾

Nodes

Refresh

Queries

Hover over the queries graph or click on a query ID in the legend to inspect queries. Click and drag on any graph to zoom in.

**CPU utilization****Network receive Throughput****Network transmit Throughput**

- Redshift dashboard
- Clusters**
- Snapshots
- Security
- Parameter groups
- Reserved nodes
- Events
- Connect client

Cluster: **vinodd ▾**

Configuration

Status

Performance

Queries

Loads

Table restore


Cluster Status

Cluster Status	available
Database Health	healthy
In Maintenance Mode	no
Parameter Group Apply Status	in-sync
Pending Modified Values	None

Recent Events

Time ▾	Event ▾	Source ID ▾	Source type ▾
Mar 6 10:43 AM	Amazon Redshift cluster 'vinodd' has been created at 2017-03-06 15:43 UTC and is ready for use.	vinodd	cluster
Mar 6 10:43 AM	Cluster restart is complete.	vinodd	cluster
Mar 6 10:43 AM	Cluster is being restarted.	vinodd	cluster

▾ CloudWatch Alarms

▾  No alarms configured

No alarms created. You can create an alarm using the Create Alarm button above