



Data Science on Redshift

Data Science Connect 2020

John Wyant, Senior Analytics Solutions Architect
October 7th, 2020



Agenda

- Analytical Landscape Overview
- Redshift Overview
- Redshift Data Science Enabling Features
- Accessing Redshift
 - Amazon Console
 - JDBC
 - Notebook
- Sagemaker Python (scikit-learn) Demo – Sales Forecasting
- Sagemaker Training Demo – Smart Meter Prediction

Data warehousing trends



Migrations to
the cloud



Exponential growth of
event data

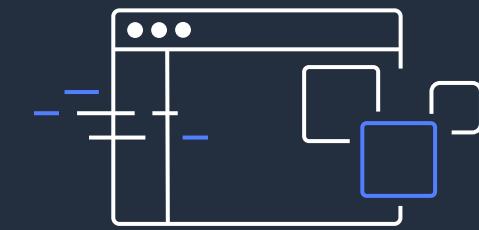


End-to-end insights from
analyzing all your data

Challenges of data analytics at scale



Data volume,
variety, velocity



Performance,
concurrency



Multiple
analytics needs



Security,
governance



Increasingly
costly, inflexible

Data lake architectures

Bringing together the best of both worlds



Extends or evolves DW architectures

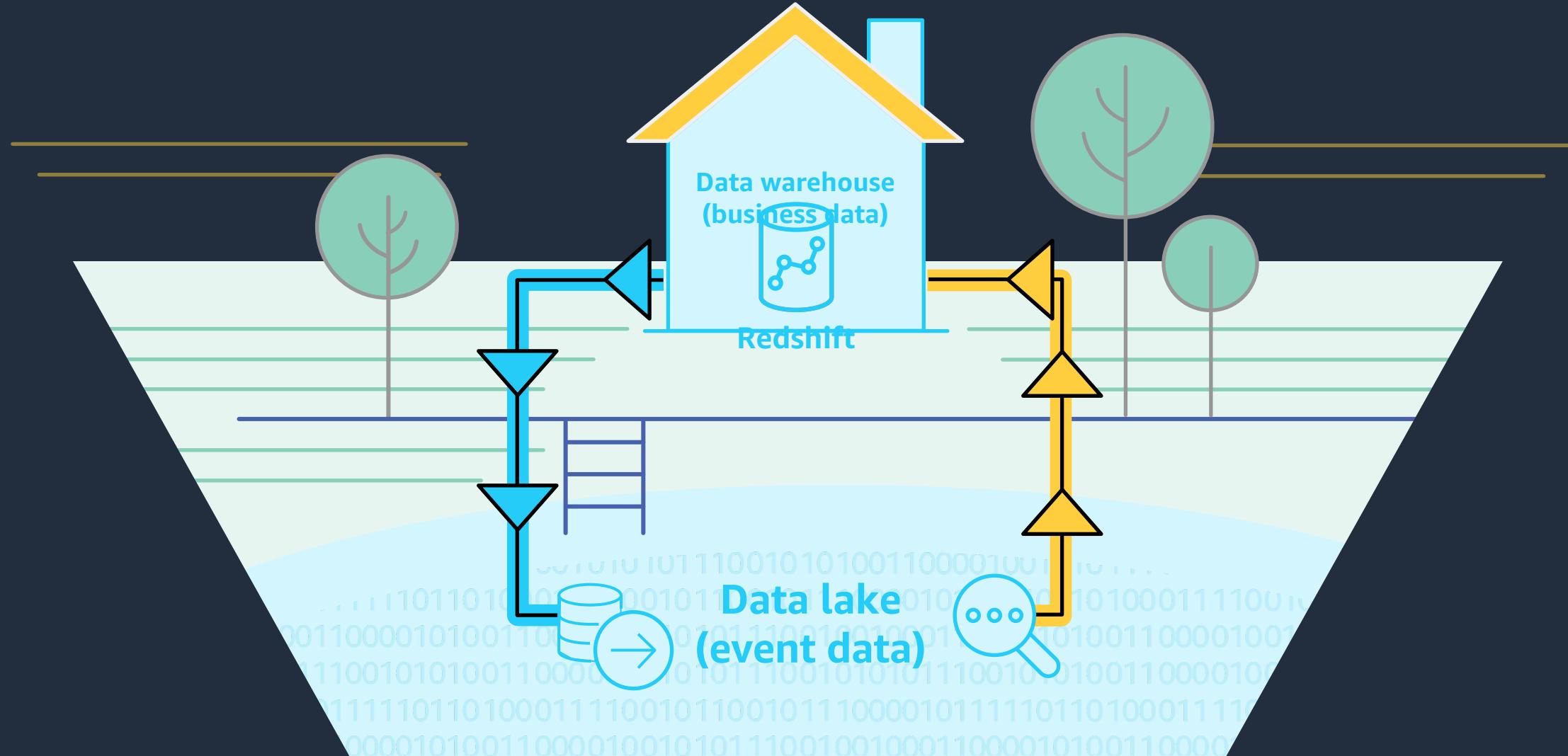
Store any data in any format

Durable, available, and exabyte scale

Secure, compliant, auditable

Run any type of analytics from DW to Predictive

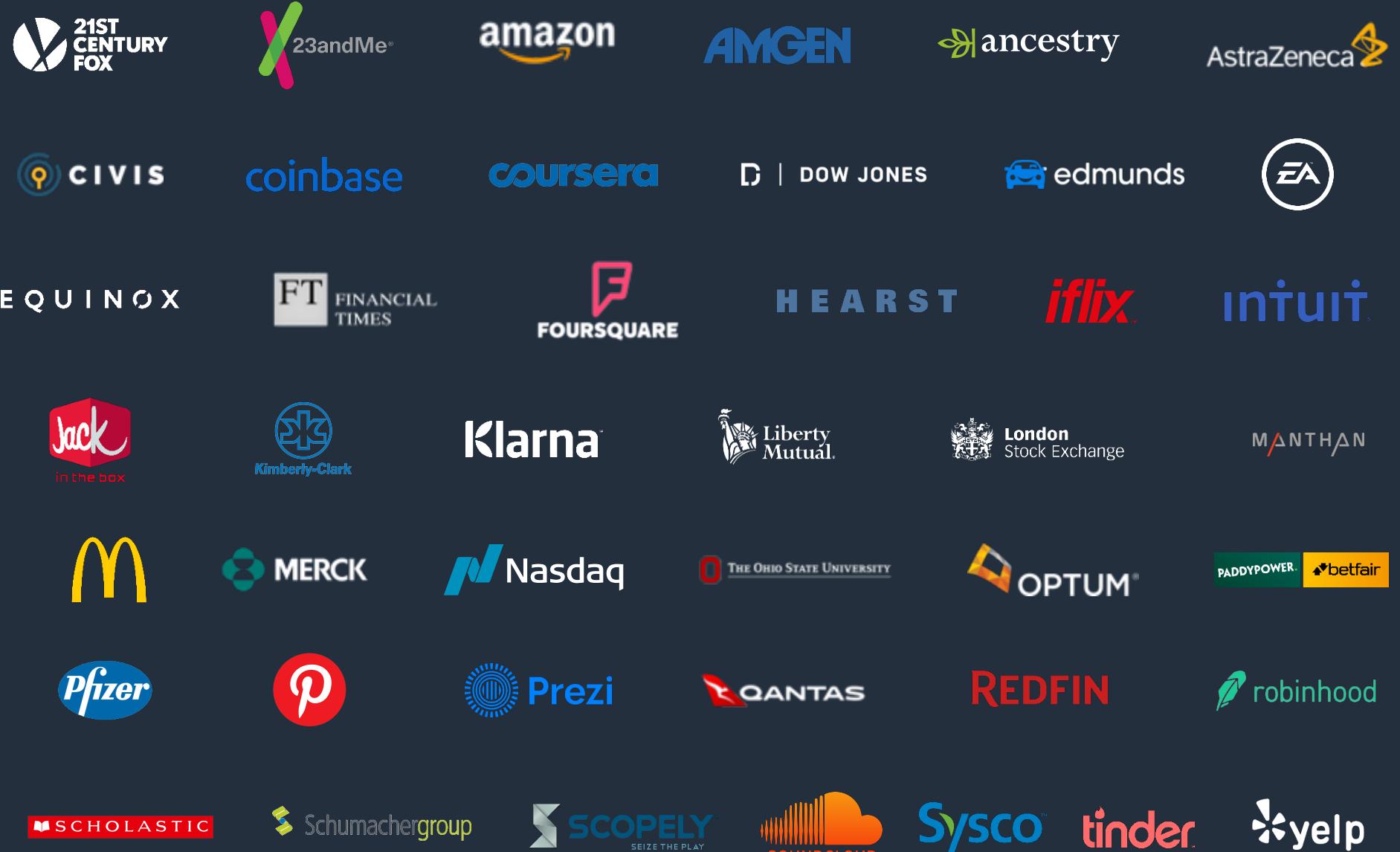
Cost-effectively scale compute and storage



Customers moving to **data lake** architectures
Redshift enables you to have a **lake house** approach



Amazon
Redshift
is the most popular
cloud data warehouse



3M Health Information Systems modernized their data warehouse with Redshift



Challenge

3M HIS needed to deliver customer value while scaling to support expected business growth. 3M HIS was already processing complex healthcare data for many customers, requiring a lot of complex transformations to get data into a format useful for analytics or machine learning.

Solution

They migrated applications installed on-premises or on other cloud hosting providers to AWS and chose Amazon Redshift for their data warehouse.

Benefits

- Ability to load data in near real-time
- Scalability to store 10x the data of the existing solution
- Cost-effective, with the ability to integrate with other analytics solutions

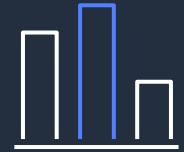
Amazon Redshift benefits

10's of thousands of customers use Redshift & process exabytes of data per day



Data lake & AWS integrated

Lake Formation catalog & security,
Exabyte scale query (spectrum & federated),
AWS integrated (DMS, CloudWatch)



Best performance

Up to 3x faster than other
cloud data warehouses



Lowest cost

Up to 75% less than other cloud data
warehouses & predictable costs



Most scalable

Virtually unlimited
elastic linear scaling



Most secure & compliant

AWS-grade security, (e.g. VPC, encryption
with KMS, Cloud Trail), Certifications such
as SOC, PCI, DSS, ISO, FedRAMP, HIPAA



Easy to manage

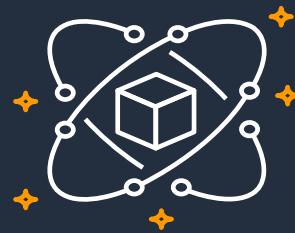
Easy to provision & manage, automated
backups, AWS support, 99.9% SLAs

Self-optimizing and easy to maintain



Easy to manage

Easy to provision & manage, automated backups, AWS support, 99.9% SLAs



Automatic Optimizations

Self-learning and self-optimizing to turbo charge performance based on usage patterns



Easy to customize

Get the best out of the cluster hardware and spend where it matters to the business

Amazon Redshift architecture

Massively parallel,
shared-nothing architecture

Leader node

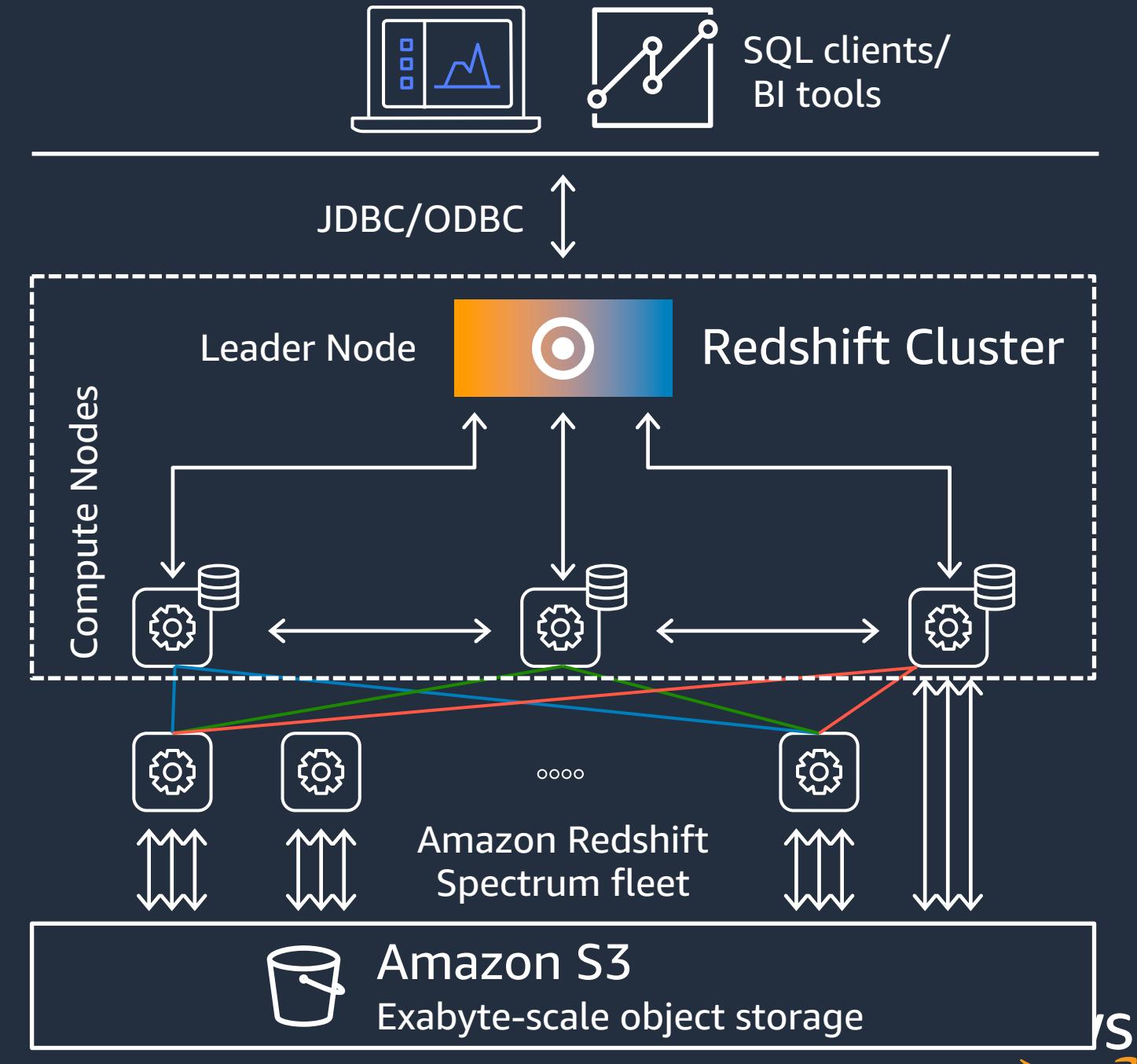
- SQL endpoint, stores metadata
- Coordinates parallel SQL processing
- Free for any cluster with two or more nodes

Compute nodes

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

Amazon Redshift Spectrum nodes

- Serverless, not managed by customer, bring power proportional to cluster slices
- Execute queries directly against data lake



Very Simple to Create and Manage Redshift Warehouse

Okay, you have your VPC and IAM roles already defined from other AWS services you've been using, and now you want to launch a Redshift cluster? Easy-peasy!

1. Select Redshift from the AWS Service Console
2. Use the Create Cluster wizard to select the size of your dataset and enter basic information such as instance type, number of nodes, IAM role to use, etc.
3. Click the “Create cluster” button

Amazon Redshift > Clusters > Create cluster

Create cluster

Find the best cluster configuration for your needs [Learn more](#)

What is your uncompressed data size?

324 TB

ra3.16xlarge

Compute: \$13.04/node/hour Storage: \$0.024/GB/month

On-demand: USD 104.32/hour Nodes: 8
Reserved (1 yr): USD 68.88/hour Compressed Storage: 227.68 TB
Reserved (3 yr): USD 39.12/hour

Select this configuration

If you're doing a proof of concept on Amazon Redshift, [follow this guide](#) or reach the [Amazon Redshift team](#) for help.

Cluster details

Cluster identifier
This is the unique key that identifies a cluster.
 The identifier must be from 1 to 63 characters. Valid characters are a-z (lowercase only) and - (hyphen).

Database port (optional)
Port number of the port where the database accepts inbound connections.
 The port must be numeric (1150-65535).

Master user name
Enter a login ID for the master user of your DB instance.
 The name must be 1-128 alphanumeric characters, and it can't be a [reserved word](#).

Master user password

 Show password

● The value must be 8-64 characters. ● The value must contain at least one uppercase letter.
● The value must contain at least one lowercase letter. ● The value must contain at least one number.

Amazon Redshift has been innovating quickly

Robust result set caching

Large # of tables support ~20000

Copy command support for ORC, Parquet

IAM role chaining

Elastic resize

Groups

Redshift Spectrum: date formats, scalar json and ION file formats support, region expansion, predicate filtering

Auto analyze

Health and performance monitoring w/Amazon Cloud watch

Automatic table distribution style

Cloud watch support for WLM queues

Performance enhancements—hash join, vacuum, window functions, resize ops, aggregations, console, union all, efficient compile code cache

Auto WLM

~25 Query Monitoring Rules (QMR) support

Concurrency Scaling

Auto analyze for incremental changes on table

200+

new features & enhancement in the past 18 months

DC1 migration to DC2

Spectrum Request Accelerator

Performance: Bloom filters in joins, complex queries that create internal table, communication layer

Amazon Lake Formation integration

Auto-Vacuum sort, Auto-Analyze and Auto Table Sort

Console redesign

Manage multi-part query in AWS console

Redshift Spectrum: Row group filtering in Parquet and ORC, Nested data support, Enhanced VPC Routing, Multiple partitions

Faster Classic resize with optimized data transfer protocol

Auto WLM with query priorities

Snapshot scheduler

Performance: join pushdowns to subquery, mixed workloads temporary tables, rank functions, null handling in join, single row insert

Spatial Processing

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Column level access control with AWS Lake formation

RA3

Advisor recommendations for distribution keys

Performance of Inter-Region Snapshot Transfers

Federated Query

AZ64 compression encoding

Materialized Views

Manual Pause and Resume

Amazon Redshift: Key 2019 innovations



Performance & scalability

NEW!



RA3 with Redshift managed storage

PREVIEW!



AQUA

NEW!



AZ64

PREVIEW!



Materialized views

NEW!



Spatial data support

NEW!



Concurrency Scaling



Data lake & AWS integration

PREVIEW!



Federated query across Redshift & RDS/Aurora

NEW!



Data lake export in Parquet

NEW!



Spectrum Request Accelerator

NEW!



Amazon Lake Formation integration



Fully managed

NEW!



New management console

NEW!



Auto WLM: Query priorities

NEW!



Elastic resize scheduler

NEW!



Auto-Vacuum, Auto-Analyze & Auto Table Sort

NEW!



Stored procedures

NEW!



Distribution and sort key advisor

NEW!



Cross-instance restore

NEW!



Faster cross regional copy and change

NEW!



Auto Data Distribution

NEW!



Deferred Maintenance

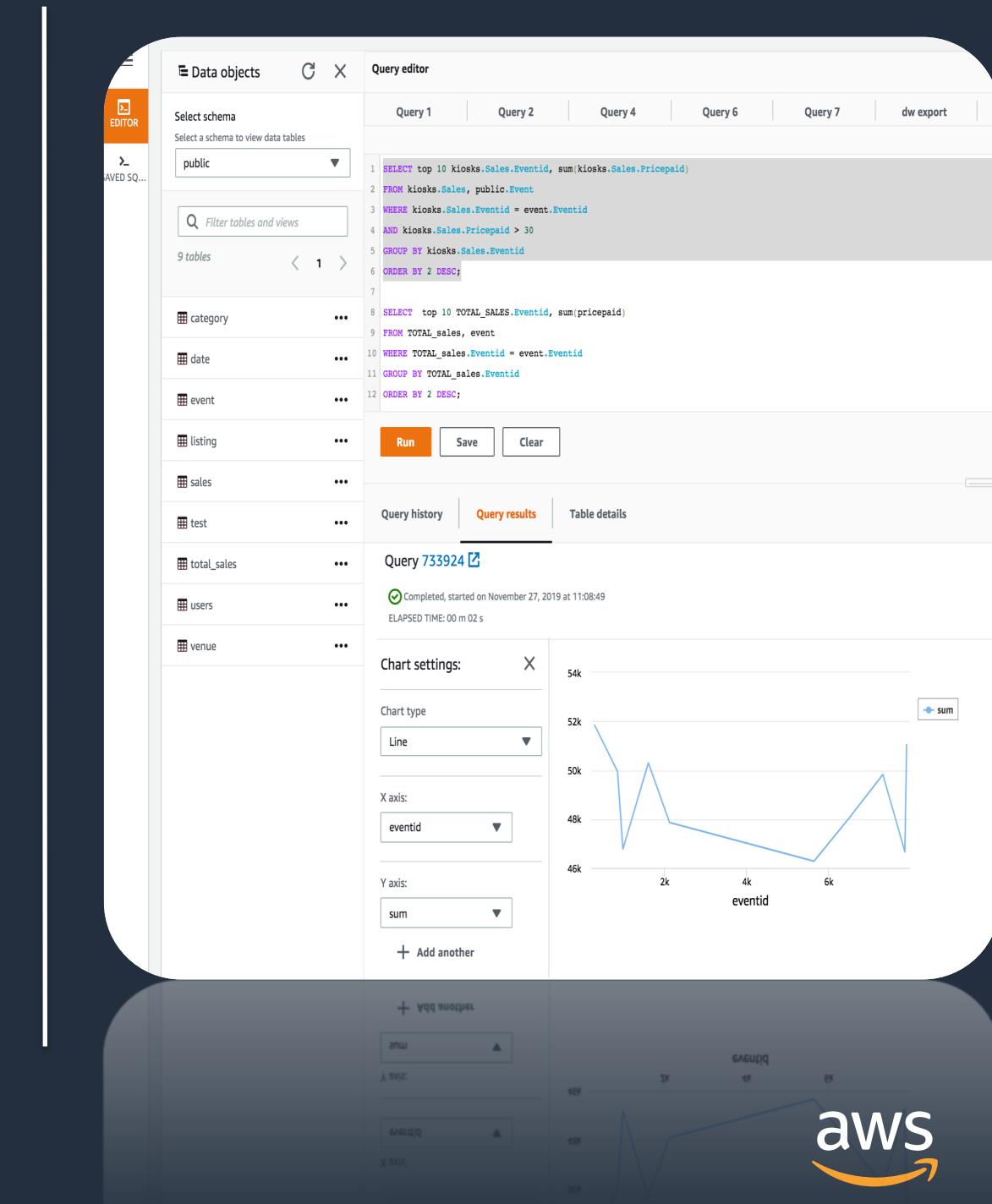
NEW!



Elastic resize

Query Editor

- Author & Execute
- Visually analyze your query results
- Results viewable & downloadable to CSV file
- Queries can be Saved for repeat execution
- In-Place analysis of Query Plan
- Considerations
 - Max 50 Query Editor users
 - Result paginated (100 rows/page)
 - Not intended for transactions (begin/commit)
 - Enhanced VPC Routing is not supported
 - Access requires IAM permissions



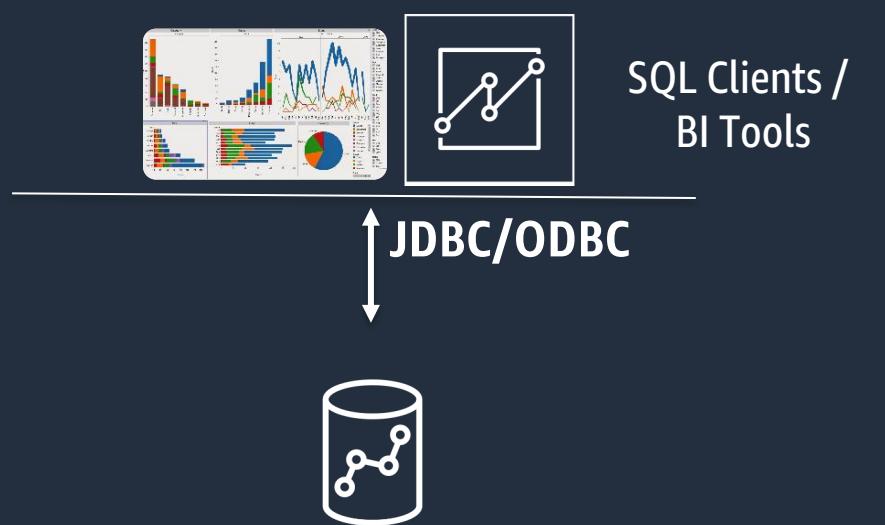
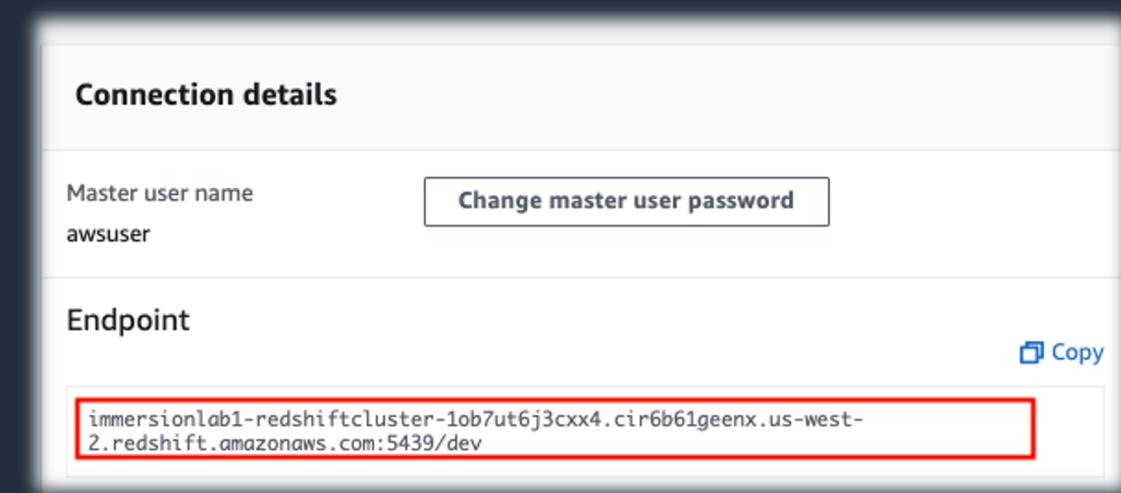
JDBC Client

- Two driver options for connecting to Redshift
 - Redshift driver (recommended)
 - 35% faster than Postgres Driver
 - Support for IAM SSO / Active Directory
 - Download from console
 - JDBC/ODBC Postgres driver (Not recommended)
- JDBC URL (Copy from Redshift console)

Examples

`jdbc:[redshift|postgresql]://endpoint:port/databaseName`

- `jdbc:redshift://demo.dsi9zn4ccku4.us-east-1.redshift.amazonaws.com:8192/pocdb`



Amazon Redshift

Connecting from Jupyter Notebook

- You can use *pyscopg2* or *pygresql* driver with *sqlalchemy*
- You can use *boto3* package and *Secrets Manager* with *SageMaker* notebook to retrieve credentials

The screenshot shows a Jupyter Notebook interface with a sidebar menu and two code cells.

File Edit View Run Kernel Tabs Settings Help

CONSOLE

- Change Kernel...
- Clear Console Cells
- Close and Shut Down...
- Insert Line Break
- Interrupt Kernel
- New Console
- Restart Kernel...
- Run Cell (forced)
- Run Cell (unforced)
- Show All Kernel Activity

EXTENSION MANAGER

- Enable Extension Manager (experimental)

FILE OPERATIONS

- ✓ Autosave Documents
- Open from Path...
- Reload Notebook from Disk
- Revert Notebook to Checkpoint
- Save Notebook
- Save Notebook As...
- Show Active File in File Browser
- Trust HTML File

HELP

- Jupyter Reference
- JupyterLab FAQ
- JupyterLab Reference
- Launch Classic Notebook
- Markdown Reference
- Reset Application State

IMAGE VIEWER

- Flip image horizontally
- Flip image vertically
- Invert Colors
- Reset Image
- Rotate Clockwise
- Rotate Counterclockwise
- Zoom In
- Zoom Out

KERNEL OPERATIONS

- Shutdown All Kernels...

LAUNCHER

- New Launcher

MAIN AREA

- Activate Next Tab
- Activate Previous Tab
- Activate Previously Used Tab
- Close All Other Tabs

0 2 Python 3 | Idle

salesanalysis.ipynb Untitled.ipynb

[25]:

```
import psycopg2
import os
import pandas as pd
rshost=os.environ['RS_HOST']
rsport=os.environ['RS_PORT']
dbname=os.environ['RS_DBNAME']
dbuser=os.environ['RS_USER']
clusterid=os.environ['CLUSTER_ID']
dbpassword = os.environ['RS_PASSWORD']

def db_connection():

    try:
        con = psycopg2.connect(
            host=rshost,
            port=rsport,
            user=dbuser,
            password=dbpassword,
            database=dbname
        )
        return con
    except psycopg2.Error:
        print("Failed to connect")

conn = db_connection()

#conn = psycopg2.connect(**db_connect_params)

#print("Connection Successful to Redshift")
query = """select s3.sales.eventid eventid, sum(s3.sales.pricepaid)
from s3.sales
group by s3.sales.eventid order by 2 desc;"""
df = pd.read_sql_query(query, conn)
df
```

[25]:

	eventid	sum
0	NaN	NaN
1	289.0	51846.0
2	7895.0	51049.0
3	1602.0	50301.0
4	851.0	49956.0
...
8508	1756.0	58.0
8509	8016.0	40.0
8510	4565.0	40.0
8511	5754.0	39.0
8512	3662.0	21.0

8513 rows × 2 columns

[26]: df.set_index('eventid',inplace=True)
df.plot()

[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1108d4ac0>

Amazon Redshift Data API

Simplifies data access from web services based applications

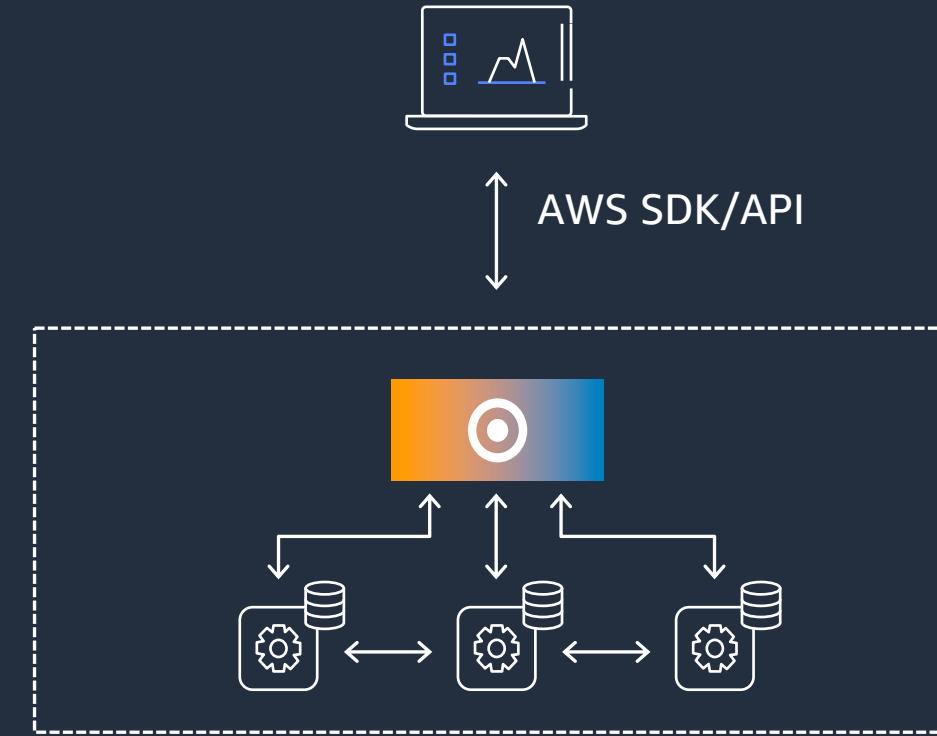
Simplifies data access from languages such as Python, Go, Java, Node.js and other languages

Query, Load and Unload data from CLI/SDK

Do not have to worry about configuring drivers, connection pools

Leverages IAM credentials or Secrets manager

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



```
response = redshift.execute_statement(  
    ClusterIdentifier = 'data_science',  
    Database = 'churn_model',  
    ...  
    Statement = 'select * from customer_profile'  
)
```

Seamless Integration with AWS Services

— Migration | ETL | Machine Learning | Data Visualization —

AWS Database Migration Service (DMS)

Fully-managed migration service

AWS Glue

Fully managed extract, transform, and load (ETL) service

Amazon EMR

Cloud-native big data platform

Amazon Sagemaker

Fully managed service to build, train, and deploy machine learning (ML) models

Amazon Quicksight

Fast, cloud-powered business intelligence service

and more...

Sagemaker / Jupyter Notebook

The screenshot shows a Jupyter Notebook interface with two code cells and one output cell. The first code cell contains Python code for connecting to a PostgreSQL database using psycopg2, with environment variables for host, port, user, password, and database. The second code cell runs a SQL query to sum sales by event ID from an S3 bucket. The output cell displays the resulting DataFrame with 8513 rows and 2 columns (eventid and sum), followed by a histogram plot showing the distribution of event IDs.

```
[25]: import psycopg2
import os
import pandas as pd
rhost=os.environ['RS_HOST']
rport=os.environ['RS_PORT']
dbname=os.environ['RS_DBNAME']
dbuser=os.environ['RS_USER']
clusterid=os.environ['CLUSTER_ID']
dbpassword = os.environ['RS_PASSWORD']

def db_connection():

    try:
        con = psycopg2.connect(
            host=rhost,
            port=rport,
            user=dbuser,
            password=dbpassword,
            database=dbname
        )
        return con
    except psycopg2.Error:
        print("Failed to connect")

conn = db_connection()

#conn = psycopg2.connect(**db_connect_params)

#print("Connection Successful to Redshift")
query = """select s3.sales.eventid, sum(s3.sales.pricepaid)
from s3.sales
group by s3.sales.eventid order by 2 desc;"""
df = pd.read_sql_query(query, conn)
df
```

eventid	sum
0	NaN
1	289.0
2	7895.0
3	1602.0
4	851.0
...	...
8508	1756.0
8509	8016.0
8510	4565.0
8511	5754.0
8512	3662.0

[26]: df.set_index('eventid', inplace=True)
[26]: df.plot()
[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1108d4ac0>

50000
40000
30000

Unloading Data: UNLOAD Command

UNLOAD command is the reverse of COPY, in that it outputs data from Amazon Redshift to S3

Runs from a SELECT statement. Order By clause respected by UNLOAD if PARALLEL=OFF

Encryption & compression handled automatically

Runs in parallel on all compute nodes

UNLOAD output

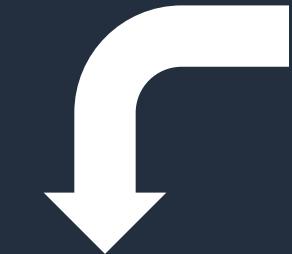
CSV or Parquet (**Data Lake Export**) file formats

Generates > 1 file per slice for all compute nodes

Max file size written on S3 can be controlled (max internal limit 6.2GB)

Generates a manifest for all unloaded files (useful for COPY into another cluster)

Control if files can overwrite existing locations or not



```
UNLOAD ('select-statement')  
TO 's3://object-path/name-prefix'  
iam_role "arn" [ option [ ... ] ]
```

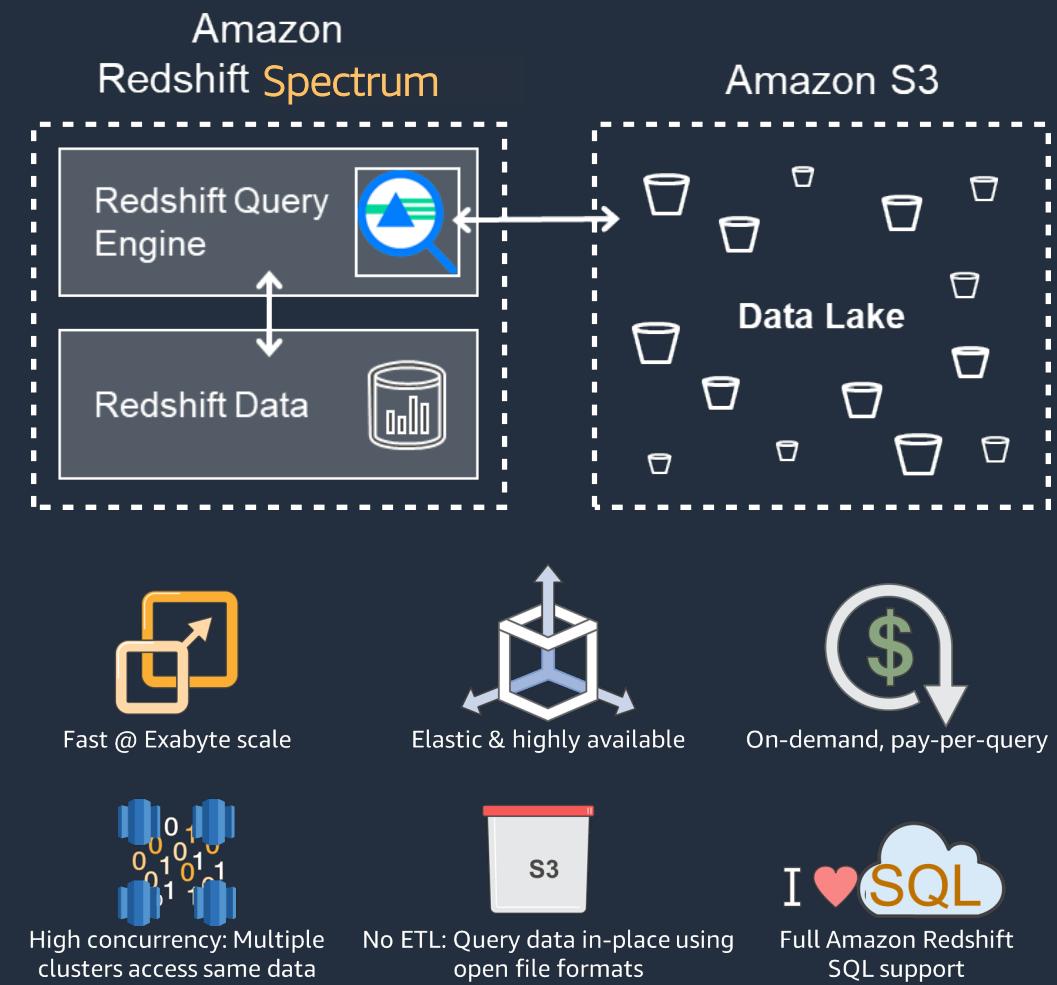
Redshift Spectrum Overview

Redshift *Spectrum* is a feature of Redshift that allows Redshift SQL queries to reference external data on Amazon S3 as they would any other table in Amazon Redshift

Benefits

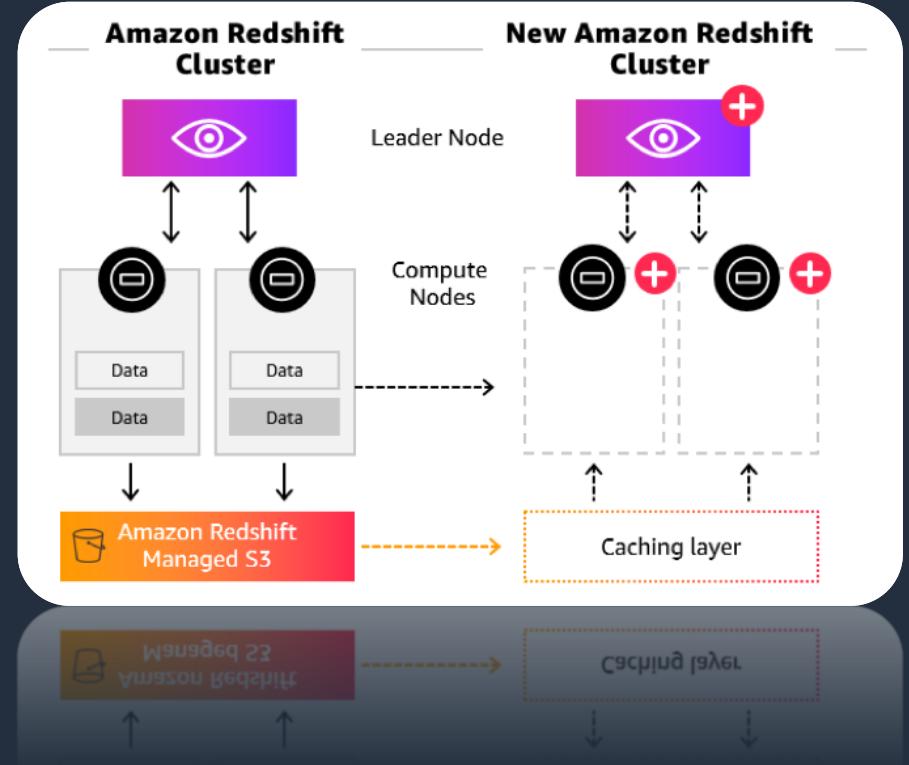
- Enables the *Lake House* pattern out-of-the-box
- Allows for querying of potentially exabytes of data in an S3 data lake from within Amazon Redshift
- Data is queried *in-place*, so no loading of data into your Redshift cluster is required
- Keeps your data *warehouse lean* by ingesting warm data locally while keeping other data in the data lake within reach
- Powered by a *separate fleet* of powerful Amazon Redshift Spectrum nodes

Run SQL queries directly against data in S3 using thousands of nodes



Redshift Concurrency Scaling Overview

- *Concurrency Scaling* is a Redshift feature that automatically adds transient clusters to your cluster within seconds to handle concurrent requests with consistently fast performance
- Free for over 97% of Redshift customers
- For every 24 hours that your main cluster is in use, you accrue a one-hour credit for Concurrency Scaling. Beyond that, customers are billed on a per-second basis per transient cluster
- Applies to Redshift local & spectrum queries
- Email notifications are issued when concurrency scaling occurs



Machine learning based automatic optimizations

To simplify maintenance and turbo charge performance

Automates table maintenance

Optimizes for peak performance
as data and workloads scale

Leverages machine learning

Offers prescriptive recommendations
with ability to apply changes dynamically



Automatic
Analyze



Automatic Table
Distribution Style



Distribution/Sort
Key Advisors



Automatic
Vacuum Delete



Automatic
Table Sort

The Amazon ML Stack

AI Services

VISION	SPEECH	LANGUAGE	CHATBOTS	FORECASTING	RECOMMENDATIONS				
 REKOGNITION IMAGE	 REKOGNITION VIDEO	 TTEXTRACT	 POLLY	 TRANSCRIBE	 TRANSLATE	 COMPREHEND	 LEX	 FORECAST	 PERSONALIZE

ML Services

 Amazon SageMaker	Ground Truth	Notebooks	Algorithms + Marketplace	Reinforcement Learning	Training	Optimization	Deployment	Hosting
--	--------------	-----------	--------------------------	------------------------	----------	--------------	------------	---------

ML Frameworks + Infrastructure

FRAMEWORKS	INTERFACES	INFRASTRUCTURE
 TensorFlow  PYTORCH	 GLUON  Keras	 EC2 P3 & P3DN  EC2 G4  EC2 C5  FPGAS  GREENGASS  ELASTIC INFERENCE  INFERENTIA

Demo Time!





Thank You!

jwyant@amazon.com

<https://aws.amazon.com/big-data/datalakes-and-analytics/>

Try it on your own: https://github.com/aws/amazon-sagemaker-examples/blob/master/advanced_functionality/working_with_redshift_data/working_with_redshift_data.ipynb