Day - 15 (10th June)

10 June 2024 10:08

TechSavvy, an e-commerce company, has decided to migrate their data warehouse to AWS Redshift to handle their growing data analytics needs. Given their requirements for scalability, performance, cost-effectiveness, and integration, which of the following steps should TechSavvy take to ensure a successful migration and optimal use of AWS Redshift?

2.Data Migration:

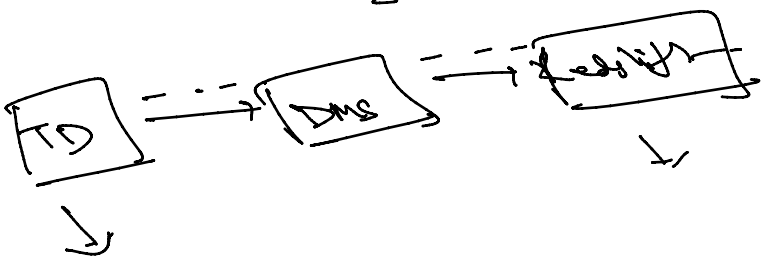
To minimize downtime and ensure data integrity during migration from their traditional databases to Redshift, TechSavvy should:

Use AWS Database Migration Service (DMS) for continuous data replication.

Perform a full data dump and load it manually during a scheduled downtime.

Use a custom ETL script to transfer data intermittently as needed.

Defer the migration to non-peak business hours without any particular strategy.



Data Ingestion:
For ongoing data ingestion from various sources into Redshift, TechSavvy should leverage:

Amazon S3 for batch uploads and manual intervention for real-time data.

AWS Glue for scheduled ETL jobs and Amazon Kinesis for streaming data.

Amazon EC2 instances to run custom ingestion scripts.

Amazon RDS to directly replicate data into Redshift.

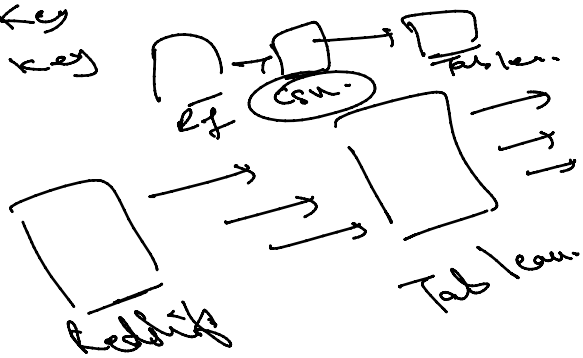
4.Query Optimization:
To achieve optimal query performance in Redshift, TechSavvy should:

Disable automatic sorting and compression to reduce processing time.

Utilize Redshift's columnar storage and distribution keys effectively.

Perform all queries without considering data distribution and sorting.

Use only default settings and avoid query tuning for simplicity.



5.Integration with BI Tools:
When integrating AWS Redshift with Tableau for seamless reporting, TechSavvy should consider:

Directly accessing the raw data without any aggregation to reduce complexity.

Using Amazon QuickSight as an intermediate step before Tableau for better performance.

Establishing a live connection from Tableau to Redshift and leveraging Redshift's JDBC/ODBC connectors.

Exporting data from Redshift to CSV files for Tableau import.

6.Security and Compliance:
To ensure data security and regulatory compliance, TechSavvy should utilize Redshift's capabilities by:

Storing all data in Redshift without encryption to simplify access.

Applying Redshift's encryption features (such as encryption at rest and in transit) and managing IAM roles for access control.

Allowing unrestricted access to the Redshift cluster for easier data access.

Ignoring VPC and network security configurations as Redshift is already secure.

7.Cost Management:
To effectively manage and optimize costs while using AWS Redshift, TechSavvy should:

Scale up to the highest available node type regardless of workload needs.

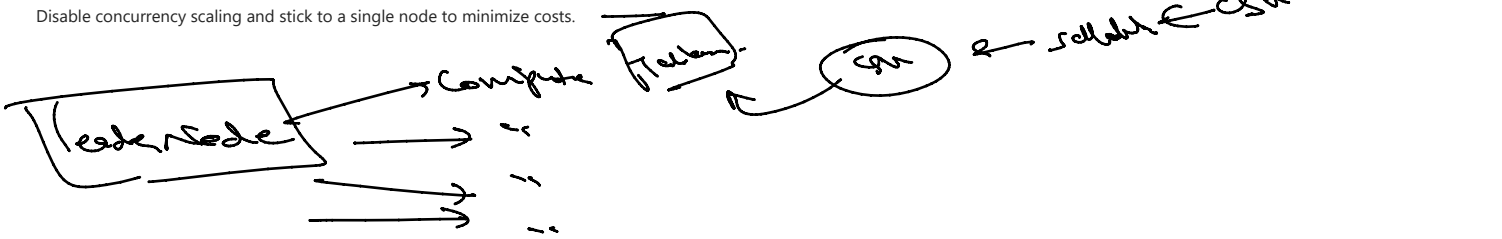
Use Reserved Instances and Spectrum for cost-effective and scalable data storage and querying.

Always run their Redshift clusters 24/7 to avoid the overhead of restarting them.

Use Reserved Instances and Spectrum for cost-effective and scalable data storage and querying.

Always run their Redshift clusters 24/7 to avoid the overhead of restarting them.

Disable concurrency scaling and stick to a single node to minimize costs.



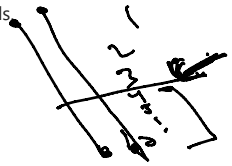
TechSavvy, an e-commerce company, is using AWS Glue as part of their data ingestion and transformation pipeline into AWS Redshift. They need to handle data from various sources efficiently and transform it for analysis. Considering their requirements, which of the following approaches should TechSavvy take to best utilize AWS Glue?

8.Data Ingestion:

To efficiently ingest and catalog data from multiple sources into AWS Redshift using AWS Glue, TechSavvy should:

Use AWS Glue Crawlers to automatically detect and catalog data stored in Amazon S3 and other sources, then schedule ETL jobs to transform and load data into Redshift.

Manually write and schedule custom scripts to extract data from sources and load it into Redshift without using AWS Glue's built-in tools.



Upload all data directly to Redshift without pre-processing, relying on Redshift to handle any necessary transformations.

Use AWS Glue solely for batch processing, and handle real-time data ingestion with separate custom applications.

9.Data Transformation:

For transforming and preparing data for analysis in AWS Redshift, TechSavvy should:

Rely solely on Redshift's SQL capabilities to perform all data transformations after ingestion.

Use AWS Glue to create ETL (Extract, Transform, Load) jobs that automate the process of transforming data before it is loaded into Redshift.

Perform all data transformations manually before loading them into AWS Glue to avoid using ETL jobs.

Use AWS Glue only for simple file format conversions, and depend on third-party tools for complex data transformations.

10.Scheduling and Automation:

To automate their data processing workflows and ensure timely data updates in Redshift, TechSavvy should:

Manually run AWS Glue ETL jobs whenever new data needs to be processed.

Use AWS Glue's scheduling capabilities to automate the execution of ETL jobs based on a regular schedule or event triggers.

Write custom code to trigger ETL jobs and manage scheduling outside of AWS Glue.

Schedule Redshift queries directly for data updates, bypassing the need for ETL automation in AWS Glue.

11.Cost Management:

To manage and optimize costs while using AWS Glue for their data processing needs, TechSavvy should:

Continuously run AWS Glue ETL jobs to process data in real-time, regardless of actual data arrival patterns.

Optimize ETL job scripts to minimize execution time and use AWS Glue's "on-demand" mode to run jobs only when needed.

Use AWS Glue exclusively for large-scale data processing tasks and avoid it for smaller, more frequent updates.

Provision and maintain a dedicated fleet of EC2 instances to run AWS Glue ETL jobs to control costs better.

TechSavvy is expanding its data pipeline to handle real-time event processing and notifications. They plan to use AWS SQS and SNS to manage their messaging and notification requirements. Given their needs for scalable, reliable, and decoupled services, which of the following approaches should TechSavvy take to best leverage AWS SQS and SNS?

12.Real-Time Order Processing:
TechSavvy wants to decouple their order processing system to handle high volumes of incoming orders efficiently. They decide to use AWS SQS. Which strategy best aligns with their needs?

- Use a single SQS queue to receive all incoming order messages and process them sequentially by a single consumer.
- Use multiple SQS queues to distribute incoming order messages, with each queue dedicated to a specific type of order, allowing parallel processing by multiple consumers.
- Send order messages directly to their processing microservices without using SQS to avoid the complexity of queue management.
- Use SQS for batch processing of orders, accumulating a large number of orders before processing them together.

13.Notifications for Inventory Updates:

TechSavvy needs to notify multiple services whenever there is an inventory update. To achieve this using AWS SNS, they should:

- Use a single SNS topic to publish inventory updates and subscribe all relevant services to this topic to receive notifications.
- Publish inventory updates directly to each service endpoint to ensure they receive the messages.
- Use individual SNS topics for each service to segregate notifications and reduce the complexity of message delivery.
- Rely on periodic polling of inventory status by services instead of using SNS for real-time notifications.

14.Ensuring Message Delivery and Durability:

To ensure that critical messages (such as order confirmations) are reliably delivered and retained until processed, TechSavvy should:

- Use SNS with no SQS integration, assuming that SNS will handle all message delivery requirements.
- Integrate SNS with SQS by subscribing an SQS queue to an SNS topic to decouple message publishing and ensure messages are stored until they are processed.
- Use SQS with no SNS, relying solely on SQS's message retention and durability features.
- Implement a custom messaging solution to guarantee message delivery and durability.

15.Scaling and Handling High Throughput:

To handle peaks in traffic and ensure that their system scales efficiently, TechSavvy should:

- Use SQS with a single consumer and scale vertically by upgrading the consumer's hardware during peak times.
- Use SNS to broadcast all messages and rely on downstream services to manage message queuing and processing.
- Use a combination of SNS and SQS, where SNS handles message broadcasting and SQS queues distribute the load across multiple consumers to scale horizontally.
- Avoid using SNS and SQS and implement an in-house message queuing system tailored to their peak traffic requirements.

16.Cost Management:

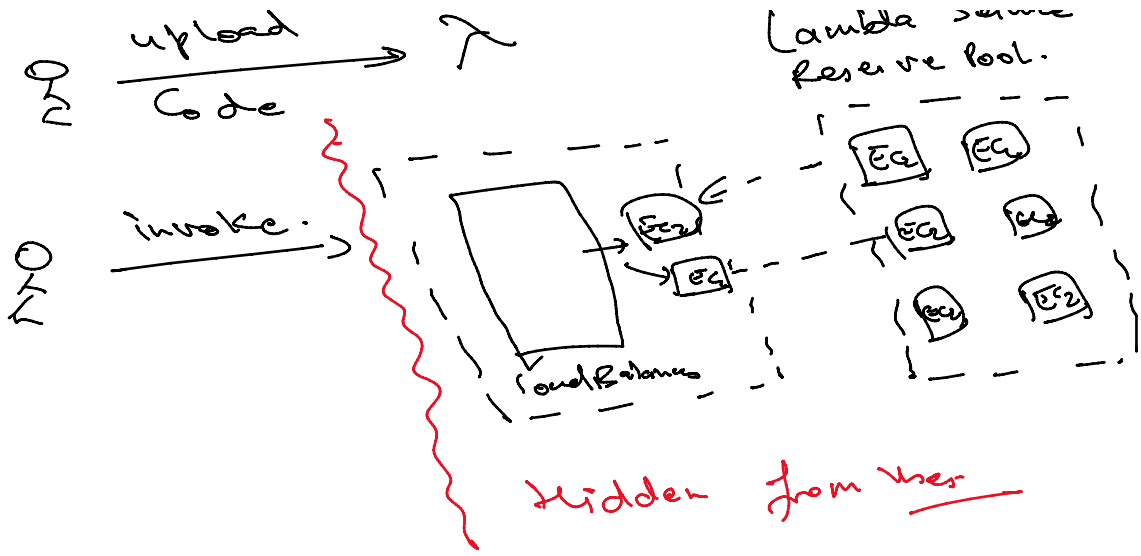
To manage and optimize costs while using AWS SQS and SNS, TechSavvy should:

- Use SNS exclusively since it's cheaper and easier to implement than SQS.
- Use SQS for message queuing but limit the number of queues to minimize costs.
- Combine SQS and SNS, using SNS to reduce the number of direct service calls and SQS to ensure cost-effective message processing and retention.
- Implement cost control by throttling message sending and receiving rates to minimize usage of both services.

Lambda :-

1. Run on-demand or schedule.
2. Support (Python, C#, go, java, node, ruby)
3. Serverless service
4. Pay per invocation, duration & memory.
5. Built in Metrics, cloudwatch
6. execution is 15 mins

Lambda Service Reserve pool.



Why popular?

(1) Integrate with other AWS Services

Step function
DynamoDB
API
S3
SQS
SNS

New features:-

(1) Lambda Edge:- Deploy lambda functions across different Region

(2) Lambda destination:- pipe o/p of lambda o/p directly to other AWS service.

(3) Layers:- Easily include libraries

(4) Provisioned Concurrency:- Cold start → Reduce latency