

Disclaimer: These slides are copyrighted and strictly for personal use only

- This document is reserved for people enrolled into the [AWS Certified Database Specialty course by Stephane Maarek and Riyaz Sayyad.](#)
- Please do not share this document, it is intended for personal use and exam preparation only, thank you.
- If you've obtained these slides for free on a website that is not the course's website, please reach out to piracy@datacumulus.com. Thanks!
- Best of luck for the exam and happy learning!

AWS Certified Database Specialty Course

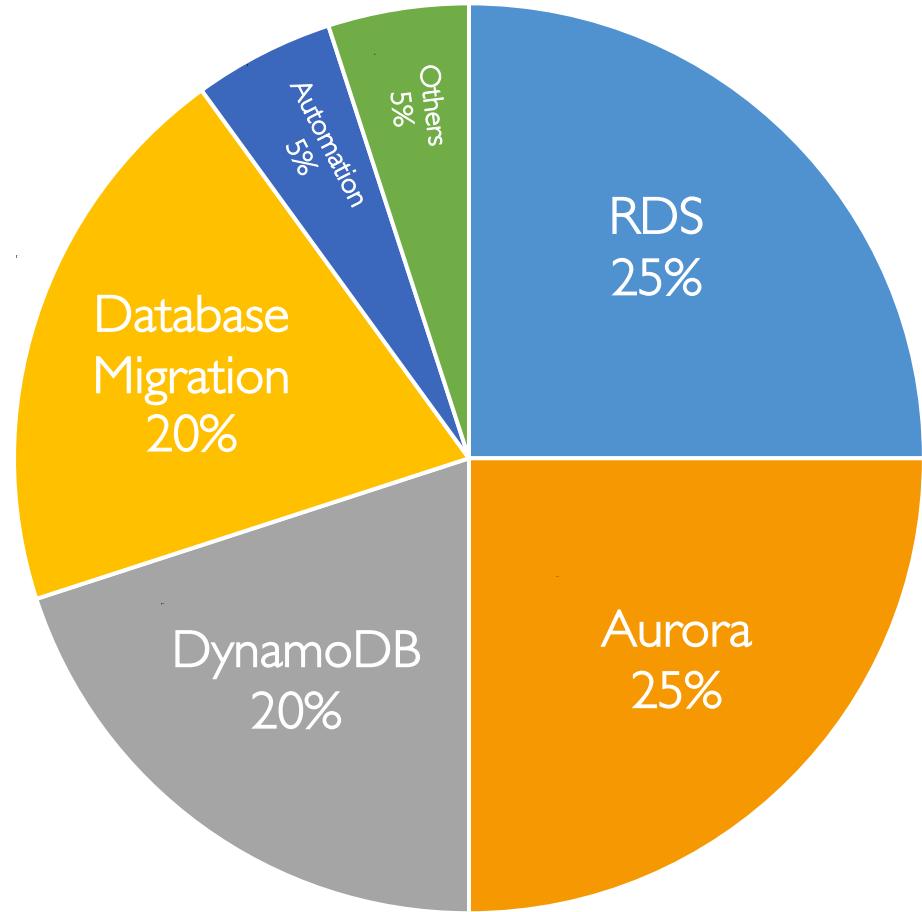
DBS-C01

Welcome!

- We're going to prepare for the Database Specialty exam – DBS-C01
- It's a challenging certification
- This course will be intense and fast-paced, yet interesting and fulfilling!
- Recommended to have previous AWS knowledge (Networking, Storage, etc.). An associate level AWS Certification is good to have.
- Preferred to have some database background
- We will cover all the AWS Database services related to the exam
- Take your time, it's not a race!

Key Focus Areas

- RDS
 - Aurora
 - DynamoDB
 - Database Migration
 - Automation
 - Others
-
- This is a ballpark distribution for general reference, and you might see a different distribution at your exam.



Services we'll learn

RELATIONAL



Amazon RDS



Amazon Aurora



Amazon Redshift

KEY-VALUE



DynamoDB



DAX



Keyspaces

DOCUMENT



DocumentDB

IN-MEMORY



ElastiCache



ElastiCache for
Redis



ElastiCache for
Memcached

GRAPH



Neptune

SEARCH



Elasticsearch
Service

TIME-SERIES



Timestream

LEDGER



QLDB

Migration & Automation



AWS DMS



Snowball Edge



CloudFormation

SECURITY AND MONITORING



AWS IAM



AWS KMS



Systems Manager



Secrets Manager



AWS CloudTrail



CloudWatch

What to expect ahead?

- We'll learn each of these database services one by one
- We'll cover all five exam domains for each database service
- We'll also learn the supporting AWS services along the way
- We'll have hands-on demos to help you reinforce your learnings
- We'll discuss exam-like questions and scenarios
- By the end of this course, you'll have learned everything you need to pass and ace this certification exam
- Trust the process

Some tips to take this course

- Take notes!
- Practice what we teach at your work if possible
- Set a target date for when you want to pass the exam
- If you feel overwhelmed, take breaks
- Re-watch lectures if we went too fast
- You can speed up / slow down the videos on the video player
- Take as much time as you need. Remember, mastery takes time!

About me – Stephane Maarek

- I'm Stephane!
- 9x AWS Certified
- Worked with AWS many years: built websites, apps, streaming platforms
- Veteran Instructor on AWS (Certifications, CloudFormation, Lambda, EC2...)
- You can find me on
 - GitHub: <https://github.com/simplesteph>
 - LinkedIn: <https://www.linkedin.com/in/stephanemaarek>
 - Medium: <https://medium.com/@stephane.maarek>
 - Twitter: <https://twitter.com/stephanemaarek>



- ★ 4.7 Instructor Rating
- ⭐ 130,318 Reviews
- 👤 471,618 Students
- ▶ 34 Courses

About me – Riyaz Sayyad

- I'm Riyaz!
- 3x AWS Certified
- Worked with AWS many years: built websites, apps, SaaS products
- Best-selling Instructor on AWS (Serverless, Lambda, DynamoDB, ...)
- You can find me on
 - YouTube: youtube.com/rizmaxed
 - LinkedIn: linkedin.com/in/riyazsayyad
 - Twitter: twitter.com/riyaznet
 - Rizmaxed: rizmaxed.com



4.5 Instructor Rating
 3,041 Reviews
 21,561 Students
 2 Courses

Purpose-built databases

Finding the right database for the right job!

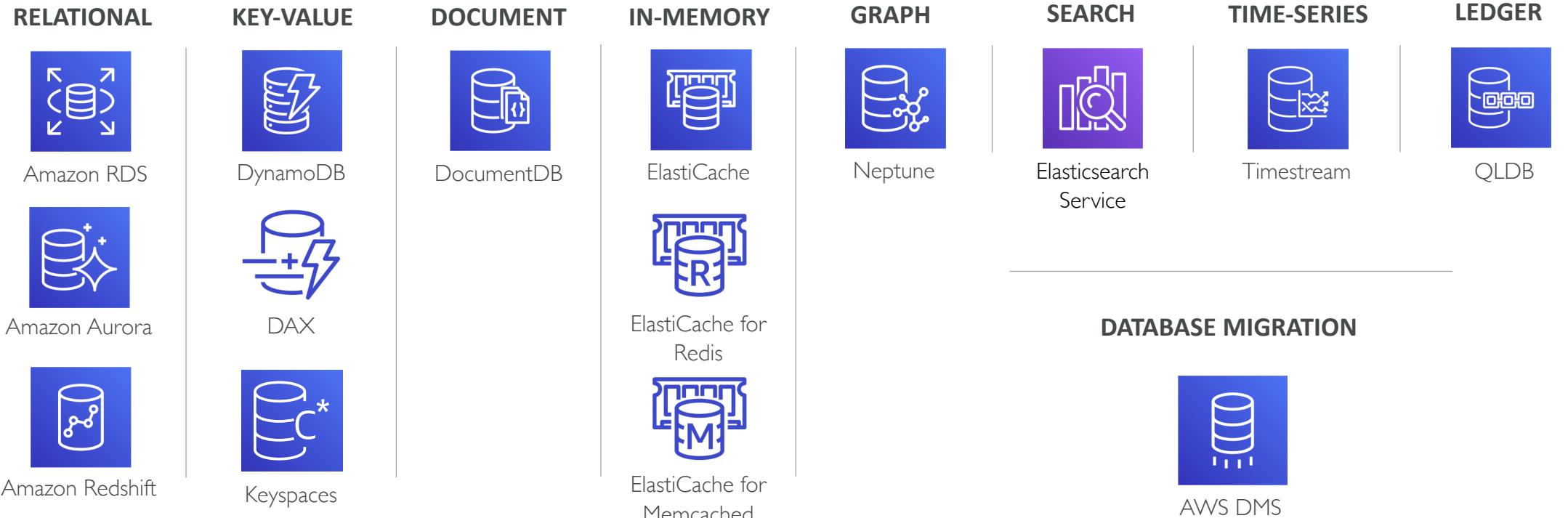
Purpose-built databases

- Databases used to be a “one-size-fits-all”: use a relational DB and off you go!
- New challenges:
 - With advent of big data, demands on volume, velocity and variety of data have increased
 - We store TBs to PBs data
 - Millions of requests per second
 - Sub-millisecond latencies and global users
- Although relational databases are here to stay (**and are evolving!**), they alone aren’t enough to cater to our data needs today
- In modern applications, it’s common to see a mix of databases
- Finding and using databases fit for purpose – right database for the right job

Choosing the Right Database

- We have a lot of managed databases on AWS to choose from
- Questions to choose the right database based on your architecture:
 - Read-heavy, write-heavy, or balanced workload? Throughput needs? Will it change, does it need to scale or fluctuate during the day?
 - How much data to store and for how long? Will it grow? Average object size? How are they accessed?
 - Data durability? Source of truth for the data ?
 - Latency requirements? Concurrent users?
 - Data model? How will you query the data? Joins? Structured? Semi-Structured?
 - Strong schema? More flexibility? Reporting? Search? RDBMS / NoSQL?
 - License costs? Switch to Cloud Native DB such as Aurora?

Purpose-built Databases from AWS



The Basics

Mastering the basics is the secret key to your success!

Data

- Data: a collection of values or information
- Database: an organized collection of this data
- Three types of data
 - Structured
 - Semi-structured
 - Unstructured
- Hence the need for different databases to suit different types of data
- We often end up working with a mix of these types

Structured Data

- Typically stored in tables with predefined structures (a.k.a. schema)
- Suitable for OLTP (transactional) and OLAP (analytical) workloads
- Typical of relational databases (indexed tables linked with foreign key relationships)
- Suited for complex queries and analytics e.g. complex table join operations

Student ID	First Name	Last Name	Birth Date
1	Amanda	Smith	05/02/1980
2	Justin	Kingsley	12/04/1985
3	Dennis	Green	11/09/1982

Semi-structured Data

- Organized data, but not constrained by a fixed schema
- Schema accommodates variable data structures
- Typical of non-relational databases
- Well suited for big data and low-latency applications
- Examples: XML and JSON

```
{  
  "trace_id": "daeae4ef-5495-4643-81ca-911d0615d5b8",  
  "request_id": "daeae4ef-5495-4643-81ca-911d0615d5b8",  
  "request_type": "NEW",  
  "request_timestamp": 1581681322024,  
  "request_status": "queued"  
},  
{  
  "trace_id": "fcdse4ef-4565-4677-888a-876d0615d888",  
  "request_id": "fcdse4ef-4565-4677-888a-876d0615d888",  
  "request_type": "NEW",  
  "request_timestamp": 1581681323182,  
  "request_status": "processing",  
  "message": "Your request is being processed.",  
  "assigned_to": "Jason Root"  
}
```

Unstructured Data

- Unorganized data
- No defined schema e.g. documents, photos, videos, music files, text messages etc
- Typical of non-relational databases, file systems, data stores or data lakes like Amazon S3

<input type="checkbox"/> Name ▾	Last modified ▾	Size ▾
<input type="checkbox"/> EG4XQRMFC0YP0.2018-10-03-06.d9b87078.gz	Oct 3, 2018 12:14:48 PM GMT+0530	1.3 KB
<input type="checkbox"/> EG4XQRMFC0YP0.2018-10-03-07.cc21b812.gz	Oct 3, 2018 12:49:48 PM GMT+0530	525.0 B
<input type="checkbox"/> EG4XQRMFC0YP0.2018-10-03-07.f50d6d2f.gz	Oct 3, 2018 12:44:48 PM GMT+0530	703.0 B
<input type="checkbox"/> EG4XQRMFC0YP0.2018-10-03-09.d0a60443.gz	Oct 3, 2018 2:59:49 PM GMT+0530	1.1 KB
<input type="checkbox"/> EG4XQRMFC0YP0.2018-10-03-12.218724e7.gz	Oct 3, 2018 6:19:49 PM GMT+0530	934.0 B
<input type="checkbox"/> EG4XQRMFC0YP0.2018-10-03-12.664590e9.gz	Oct 3, 2018 5:49:50 PM GMT+0530	851.0 B



Relational Databases

- Have a predefined schema
- Enforce strict ACID compliance and supports “joins”
- Typical use cases – OLTP (transactional) and OLAP (analytical) applications
- Examples: MySQL, PostgreSQL, MariaDB, Oracle, Microsoft SQL Server
- AWS Services:
 - Amazon RDS and Aurora for OLTP
 - Redshift for OLAP



Amazon RDS



Amazon Aurora



Amazon Redshift

Relational Databases

- Characterized by multiple tables interconnected through foreign key relationships

Students			
Student ID	Dept ID	Name	Email
1	M01	Joe Miller	joe@abc.com
2	B01	Sarah T	sarah@abc.com
Departments			
Dept ID	SPOC	Email	Phone
M01	Kelly Jones	kelly@abc.com	+1234567890
B01	Satish Kumar	satish@abc.com	+1234567891

Subjects	
Student ID	Subject
1	Physics
1	Chemistry
1	Math
2	History
2	Geography
2	Economics

Table Indexes

- Relational databases are written to and queried using SQL (Structured Query Language)
- For efficient query performance, we add indexes to the table
- Tables are indexed by their primary key by default
- Secondary indexed can be added on non-key fields

What is ACID compliance

- ACID = Atomicity, Consistency, Isolation, and Durability
 - Atomicity means “all or nothing” – a transaction executes completely or not at all
 - Consistency means once a transaction has been committed, the data must conform to the given schema
 - Isolation requires that concurrent transactions execute separately from one another
 - Durability is the ability to recover from an unexpected system failure or outage
- ACID compliance refers only to the compliance enforced by the DB
- ACID behavior can also be enforced by your application
 - e.g. your application can maintain strict foreign key relationship between two DynamoDB tables. However, DynamoDB by itself will not enforce this relationship. Another application can always override this behavior.
 - if it were MySQL tables with relationships defined on the DB, no application would be able to override the behavior.

Non-Relational Databases

- Sometimes called as NoSQL (= Not only SQL)
- Suitable for semi-structured and unstructured data
- Data stored in denormalized form
- Suited for big data applications
 - Big data = High Volume, High Velocity, High Variety
- Also suited for low-latency (high-speed) applications
(why? Simple data formats)
- Flexible data model (i.e. trades some ACID properties)
- Not suited for OLAP (analytical) applications

TYPES OF NON-RELATIONAL DATABASES

KEY-VALUE



DynamoDB

DOCUMENT



DocumentDB

IN-MEMORY



ElastiCache

GRAPH



Neptune

What is BASE compliance

- BASE = Basically Available Soft-state Eventually-consistent
- Basically available – basic R/W operations are available
- Soft-state – no consistency guarantees, data state may change
- Eventually consistent – data is expected to be consistent over time
- Use cases that prefer high performance over strong consistency

Comparison of Relational and Non-Relational Databases

Relational (SQL)	Non-Relational (NoSQL)
Data stored as tables linked through foreign key relationships	Data stored as collections or key-value pairs
Strict ACID Compliance	Not so strict ACID Compliance
Fixed Schema (structured data)	Flexible schema (Semi-structured and unstructured data)
Vertical Scaling	Horizontal Scaling
Uses SQL	Uses Object based APIs
Suited for OLTP and OLAP	Suited for OLTP (web/mobile apps)

Amazon RDS and Aurora

Relational databases ain't going away!

Amazon RDS – Overview

- RDS stands for Relational Database Service
- Uses SQL (Structured Query Language)
- Supported Engines: PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server, Aurora
- Managed DB service
- Launched within a VPC, usually in private subnet, control network access using security groups (important when using Lambda)
- Storage by EBS (gp2 or io1), can increase volume size with auto-scaling
- Backups: automated with point-in-time recovery. Backups expire
- Snapshots: manual, can make copies of snapshots cross region
- Monitoring through CloudWatch
- RDS Events: get notified via SNS for events (operations, outages...)
- Supports Multi-AZ deployments



Amazon RDS

Why use RDS?

- When you self-host your DB, you manage everything
 - You manage hardware (physical infrastructure)
 - You manage software (OS)
 - You manage application (DB)
- When you host on AWS Cloud (on EC2, but not on RDS)
 - AWS manages hardware
 - You manage software (OS)
 - You manage application (DB)
- When you use RDS, AWS manages everything
 - AWS manages hardware
 - AWS manages software (OS)
 - AWS manages application (DB)
- See the AWS Shared Responsibility Model



Amazon RDS

Using RDS vs deploying DB on EC2

- RDS is a managed service:
 - Automated provisioning, OS patching
 - Continuous backups and restore to specific timestamp (Point in Time Restore)!
 - Monitoring dashboards
 - Read replicas for improved read performance
 - Multi AZ setup for DR (Disaster Recovery)
 - Maintenance windows for upgrades
 - Scaling capability (vertical and horizontal)
 - Storage backed by EBS (gp2 or io1)
- BUT you can't SSH into your underlying DB instances



Amazon RDS

RDS Pricing Model

- When creating an RDS database, you choose:
 - Instance type (on-demand and reserved)
 - Engine type (PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server, and Aurora)
 - DB instance class (based on memory, CPU and I/O capacity requirements)
- Uses pay as you go pricing model
- Instance types
 - On-demand (Pay for compute capacity per hour)
 - Reserved (deeply discounted, 1-year or 3-year term contract)
- Storage (GB/month) / Backups / Snapshot Export to S3
- I/O (per million requests)
- Data transfer



Amazon RDS

Choosing an Instance Class

- Instance class types
 - Standard
 - Memory-optimized (memory-intensive, high performance workloads)
 - Burstable performance
- Burstable performance instances
 - Can burst to higher level of CPU performance on demand depending on the workload while providing baseline level performance at other times
 - Managed through CPU credits (1 credit = 100% CPU Core utilization for one min)
 - You get CPU credits when you use underutilize the CPU



Amazon RDS

Choosing Storage Type

- General Purpose Storage (=cost-effective SSD storage)
 - You choose the storage size
 - You get a baseline of 3 IOPS/GB
 - Volumes below 1 TiB can burst to 3,000 IOPS (uses I/O credits)
 - Use with variable workloads
 - Typically used for small to medium sized DBs and in DEV/TEST environments
- Provisioned IOPS (=high-performance storage, recommended for production)
 - You choose storage size and required IOPS
 - Fast and predictable performance
 - Up to 32,000 IOPS max per DB instance
 - Use when consistent high IOPS are required (I/O-intensive workloads)
 - Well-suited for write-heavy workloads
- If instance runs out of storage, it may no longer be available until you allocate more storage (=> use storage autoscaling)



Amazon RDS

Storage Auto Scaling in RDS

- Both storage types (gp2 & io1) support storage autoscaling
- Dynamically scales up storage capacity based on workload
- Storage is scaled up automatically when the utilization nears the provisioned capacity
- Zero downtime
- You save costs due to over-provisioning and prevent downtime due to under-provisioning
- Use when workload is unpredictable
- You must set the max storage limit for storage autoscaling



Amazon RDS

Storage autoscaling [Info](#)

Provides dynamic scaling support for your database's storage based on your application's needs.

Enable storage autoscaling

Enabling this feature will allow the storage to increase once the specified threshold is exceeded.

Maximum storage threshold [Info](#)

Charges will apply when your database autoscales to the specified threshold

1000

GiB

Minimum: 101 GiB, Maximum: 65536 GiB

Parameter groups

- Define configuration values specific to the selected DB engine
- Default parameter group cannot be edited
- To make config changes, you must create a new parameter group
- New parameter group inherits settings from the default parameter group
- Can be applied to any DB instances within the AWS region
- Examples (vary by DB engine):
 - autocommit
 - time_zone
 - force_ssl
 - default_storage_engine
 - max_connections



RDS > Parameter groups > mysql-parameter-group

mysql-parameter-group

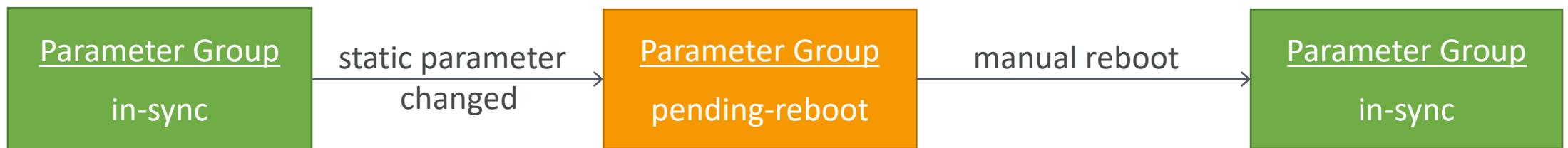
Parameters

	Name	Values	Apply type	Data type	Description
<input type="checkbox"/>	allow-suspicious-udfs		static	boolean	Controls whether user-defined functions that have only an xxx symbol for the main function can be loaded
<input type="checkbox"/>	autocommit		dynamic	boolean	Sets the autocommit mode
<input type="checkbox"/>	auto_generate_certs		static	boolean	Controls whether the server autogenerated SSL key and certificate files in the data directory, if they do not already exist.
<input type="checkbox"/>	auto_increment_increment		dynamic	integer	Intended for use with master-to-master replication, and can be used to control the operation of AUTO_INCREMENT columns
<input type="checkbox"/>	auto_increment_offset		dynamic	integer	Determines the starting point for the AUTO_INCREMENT column value
<input type="checkbox"/>	automatic_sp_privileges		dynamic	boolean	When this variable has a value of 1 (the default), the server automatically grants the EXECUTE and ALTER ROUTINE privileges to the creator of a stored routine, if the user cannot already execute and alter or drop the routine.
<input type="checkbox"/>	avoid_temporal_upgrade		dynamic	boolean	This variable controls whether ALTER TABLE implicitly upgrades temporal columns found to be in pre-5.6.4 format.
<input type="checkbox"/>	back_log		static	integer	The number of outstanding connection requests MySQL can have
<input type="checkbox"/>	basedir	/rdsdbbin/mysql	static	string	The MySQL installation base directory.
<input type="checkbox"/>	binlog_cache_size	32768	dynamic	integer	The size of the cache to hold the SQL statements for the binary log during a transaction.

MySQL Parameter Group

Changes to parameters in parameter groups

- Changes to dynamic parameters always get applied immediately (irrespective of *Apply Immediately* setting)
- Changes to static parameters require a **manual reboot**
- DB parameter group shows a status of **pending-reboot** after applying change
- Status of **pending-reboot** will not trigger automatic reboot during the next maintenance window
- You must manually reboot the DB instance to apply parameter changes
- Status will change from **pending-reboot** to **in-sync** post manual reboot



Option groups

- For configuration of optional features offered by DB engines (not covered by parameter groups)
- Default option group is empty and cannot be modified
- To make config changes, you must create a new option group
- New OG derives its settings from the default OG
- New OG is empty when created and you can add options to it
- Examples:
 - SQLSERVER_BACKUP_RESTORE in SQL Server
 - NATIVE_NETWORK_ENCRYPTION in Oracle
 - MARIADB_AUDIT_PLUGIN in MariaDB and MySQL
 - MEMCACHED to enable Memcached support in MySQL

Options						Add option
Name	Persistent	Permanent	Port	Security Groups	Version	Settings
MEMCACHED	No	No	11211	default	-	BACKLOG_QUEUE_LIMIT 1024 DAEMON_MEMCACHED_R_BATCH_SIZE 1 INNODB_API_ENABLE_MDL 0 DAEMON_MEMCACHED_W_BATCH_SIZE 1 CAS_DISABLED 0 INNODB_API_TRX_LEVEL 0 CHUNK_SIZE 48 BINDING_PROTOCOL auto INNODB_API_DISABLE_ROWLOCK 0 VERBOSITY v CHUNK_SIZE_GROWTH_FACTOR 1.25 INNODB_API_BK_COMMIT_INTERVAL 5 MAX_SIMULTANEOUS_CONNECTIONS 1024 ERROR_ON_MEMORY_EXHAUSTED 0

MySQL Option Group

Creating a database in RDS (MySQL)



Demo



DataCumulus | RIZMAXed

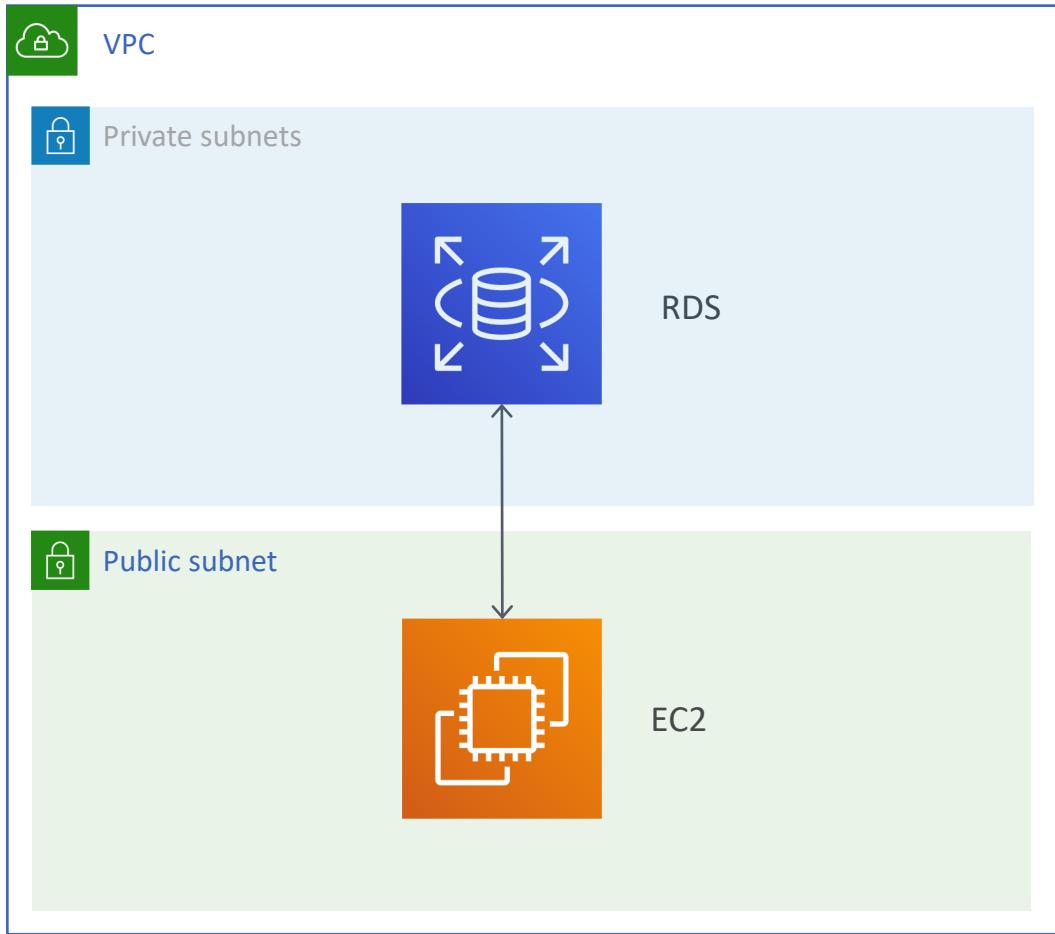
Working with parameter group and option group



Demo

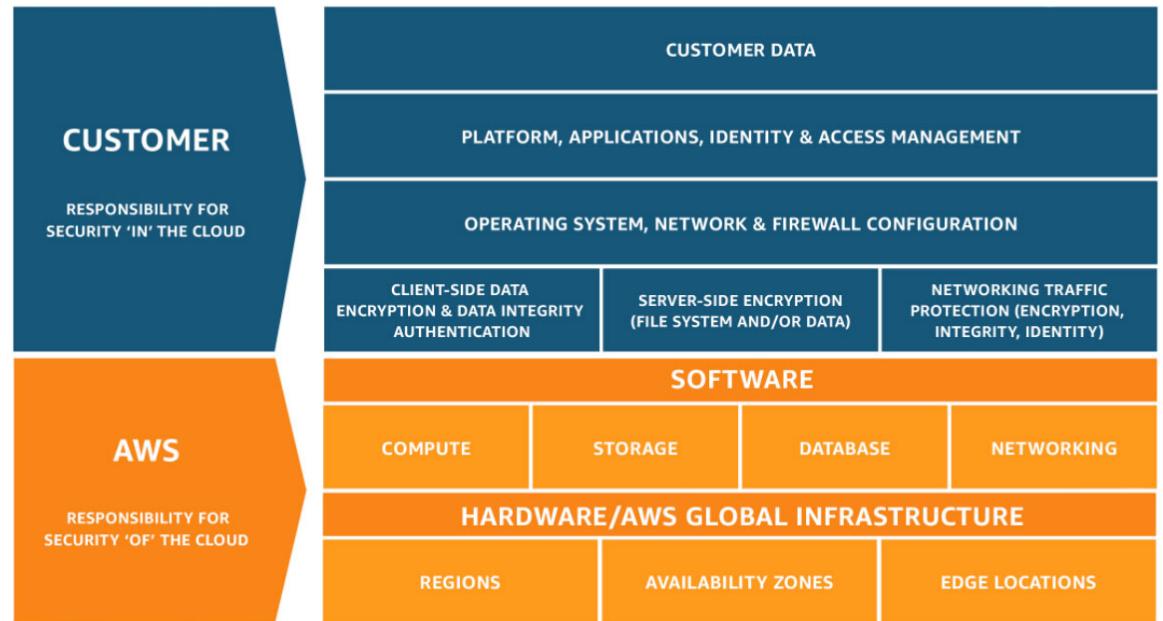
RDS Security – Network

- Launch within VPC to restrict internet access to the RDS instance
- You can't change the VPC after creating the DB
- RDS databases are usually deployed within a private subnet, not in a public one
- RDS security works by leveraging security groups (the same concept as for EC2 instances) – it controls which IP / security group can **communicate** with RDS
- Use security groups to control access at DB, EC2 and VPC level



RDS Security – Shared Responsibility

- Your responsibility:
 - Check the ports / IP / security group inbound rules in DB's SG
 - In-database user creation and permissions or manage through IAM
 - Creating a database with or without public access
 - Ensure parameter groups or DB is configured to only allow SSL connections
- AWS responsibility:
 - No SSH access
 - No manual DB patching
 - No manual OS patching
 - No way to audit the underlying instance



<https://aws.amazon.com/compliance/shared-responsibility-model/>

RDS Security – IAM

- Use IAM to secure access to the RDS DB resources
- IAM policies help control who can manage AWS RDS (through the RDS API)
- Traditional Username and Password can be used to log in to the database
- IAM-based authentication can be used to login into RDS MySQL & PostgreSQL



IAM



SSL

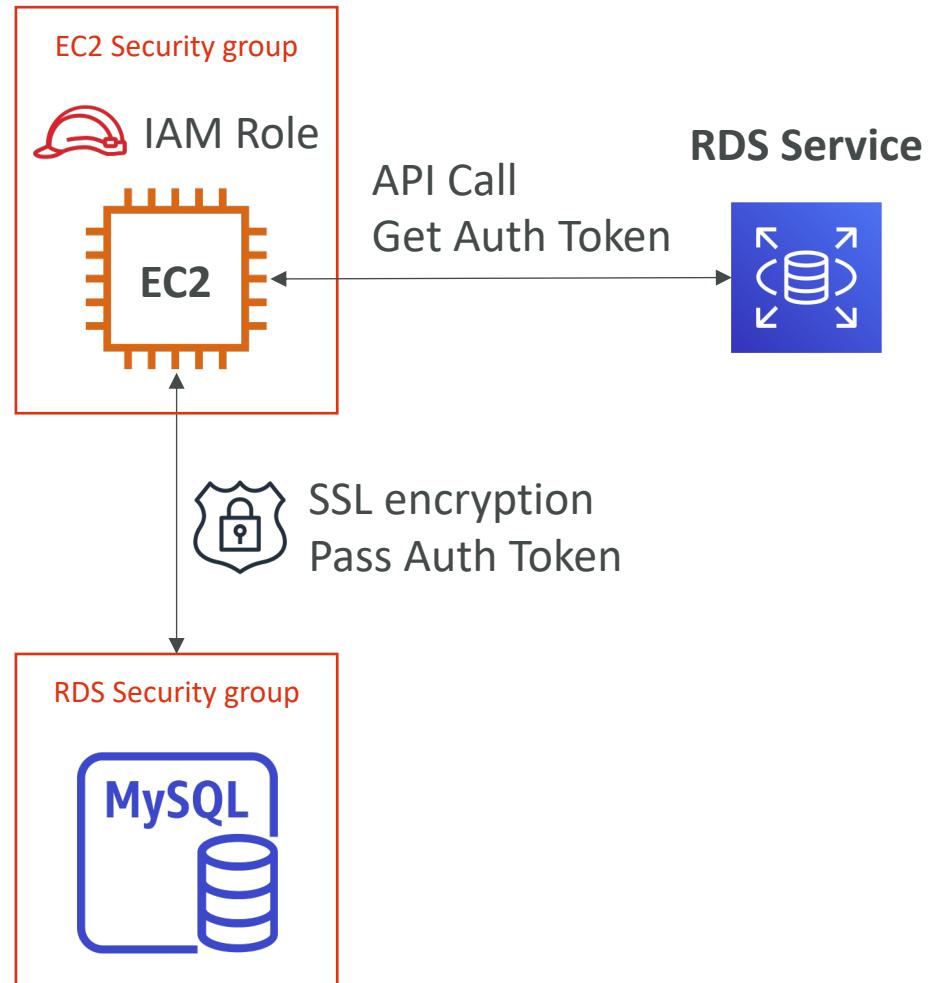
RDS Security – IAM Policy best practices

- IAM policies are used to control who can create, access, or delete DB resources in your account
- Grant **least privilege** to groups / users / roles (i.e. grant only the permissions required for the task)
- Use MFA for sensitive operations
- Use **policy conditions** – to restrict access to selected IP addresses, or within a specified date, or to require use of SSL / MFA



RDS - IAM Authentication (IAM DB Auth)

- IAM database authentication works with MySQL and PostgreSQL
- You don't need a password, just an authentication token obtained through IAM & RDS API calls
- Auth token has a lifetime of 15 minutes
- Benefits:
 - Network in/out must be encrypted using SSL
 - IAM to centrally manage users instead of DB
 - Can leverage IAM Roles and EC2 Instance profiles for easy integration



Using IAM DB Auth

- Enable IAM DB authentication on the DB cluster
- Create a DB user (without a password)
- Attach an IAM policy to map the DB user to the IAM role
- Attach the IAM role to an IAM user (or to EC2 instance)
- Now you can connect to the DB using IAM token over SSL

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "rds-db:connect"  
      ],  
      "Resource": [  
        "arn:aws:rds-db:us-east-2:1234567890:dbuser:db-ABCDEFGHIJKLM01234/db_user"  
      ]  
    }  
  ]  
}
```

Using IAM DB Auth (MySQL)

- Enable IAM DB authentication on the DB cluster
- Create a DB user (without a password)
- Attach an IAM policy to map the DB user to the IAM role
- Attach the IAM role to an IAM user (or to EC2 instance)
- Now you can connect to the DB using IAM token over SSL
- Note: use native GRANT / REVOKE for DB specific privileges

```
CREATE USER {db_username} IDENTIFIED WITH
AWSAuthenticationPlugin as 'RDS';

GRANT USAGE ON *.* TO '{db_username}'@'%' REQUIRE
SSL;

wget https://s3.amazonaws.com/rds-downloads/rds-
ca-2019-root.pem

TOKEN=$(aws rds generate-db-auth-token
--hostname {db_or_cluster_endpoint}
--port 3306 --username {db_username})"

mysql --host={db_or_cluster_endpoint} --port=3306
--ssl-ca=/home/ec2-user/rds-combined-ca-
bundle.pem --enable-cleartext-plugin
--user={db_username} --password=$TOKEN
```

Using IAM DB Auth (PostgreSQL)

- Enable IAM DB authentication on the DB cluster
- Create a DB user (without a password)
- Attach an IAM policy to map the DB user to the IAM role
- Attach the IAM role to an IAM user (or to EC2 instance)
- Now you can connect to the DB using IAM token over SSL
- Note: use native GRANT / REVOKE for DB specific privileges



You don't need to modify pg_hba.conf file for DB access. With RDS, you use DB grants.

```
CREATE USER {db_username};  
  
GRANT rds_iam to {db_username};  
  
wget https://s3.amazonaws.com/rds-downloads/rds-ca-2019-root.pem  
  
export PGPASSWORD=$(aws rds generate-db-auth-token --hostname={db_endpoint} --port=5432 --username={db_username} --region us-west-2)"  
  
psql -h {db_endpoint} -p 5432 "dbname={db_name} user={db_username} password=$PGPASSWORD sslrootcert=/home/ec2-user/rds-combined-ca-bundle.pem sslmode=verify-ca"
```

Rotating RDS DB credentials

- Use AWS Secrets Manager
- Stores credentials centrally and securely, supports audits
- Supports automatic rotation of secrets
- Secrets Manager provides a Lambda rotation function and populates it automatically with the ARN in the secret
- Integrates with RDS for MySQL, PostgreSQL, and Aurora



A screenshot of the AWS Secrets Manager 'Store a new secret' wizard, showing the 'Select secret type' step. The 'Credentials for RDS database' option is selected. The interface includes fields for 'User name' and 'Password', and a 'Show password' checkbox. Navigation steps on the left include 'Step 1 Secret type', 'Step 2 Name and description', 'Step 3 Configure rotation', and 'Step 4 Review'. The top navigation bar shows 'AWS Secrets Manager > Secrets > Store a new secret'.

Windows authentication in RDS for SQL Server

- Create AWS Managed Microsoft AD directory and setup trust relationship between your corporate AD and AWS Managed AD (called forest trust)
- AWS Managed AD is best choice if you have > 5000 users and need a trust relationship with your on-premise directory
- Alternatively, you can also use an AD connector (for an existing on-premise directory) or a Simple AD (if you have under 5000 users)
- Set up users and groups in the AD
- Enable SQL Server Windows Authentication in the RDS instance to map the directory to the DB instance (appropriate IAM role is created automatically)
- Login to DB with master user credentials and create SQL Server Windows logins for AD users

Microsoft SQL Server Windows Authentication Refresh

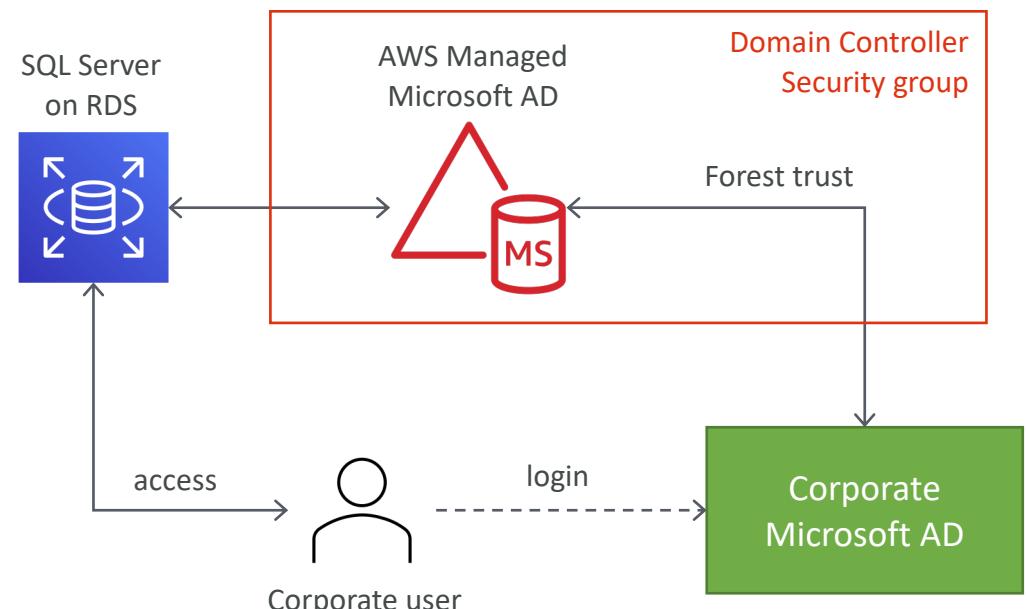
Choose a directory in which you want to allow authorized domain users to authenticate with this SQL Server instance using Windows Authentication.

Directory

corp.rizmax.com (d-9a67243a5e) ▼

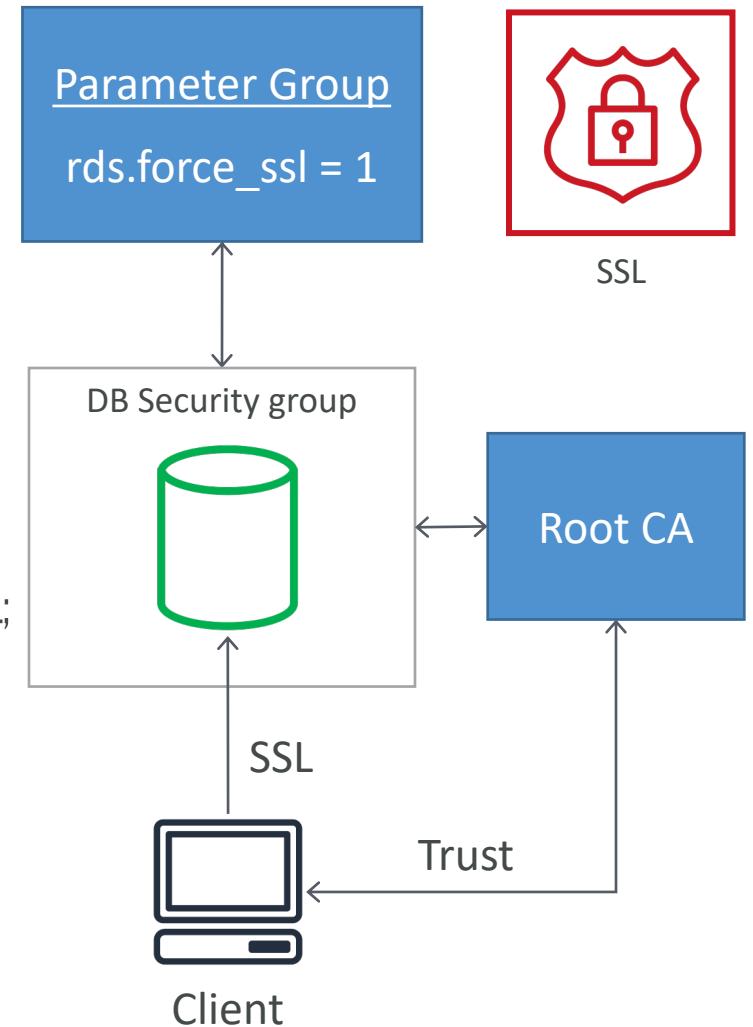
[Create a new directory](#)

By choosing a directory and continuing with database instance creation you authorize Amazon RDS to create the IAM role necessary for using Windows Authentication



RDS Encryption In Transit

- To encrypt data in transit use SSL/TLS connections (use SSL certificates to encrypt data in transit)
- Clients must trust the AWS Root CA (Certified Authority)
- To enforce SSL:
 - SQL Server or PostgreSQL: set parameter `rds.force_ssl=1` in the parameter group (static parameter, requires manual reboot)
 - MySQL or MariaDB: `ALTER USER 'mysqluser'@'%' REQUIRE SSL;`
 - Oracle: add **SSL** option to the DB instance option group
- To connect using SSL:
 - Provide the SSL Trust certificate (can be downloaded from AWS)
 - Provide SSL options when connecting to database
 - Not using SSL on a DB that enforces SSL would result in error



Examples of connecting over SSL

- PostgreSQL

```
sslrootcert=rds-cert.pem sslmode=[verify-ca | verify-full]
```

- MySQL

```
--ssl-ca=rds-cert.pem --ssl-mode=VERIFY_IDENTITY (MySQL 5.7+)
```

- MariaDB

```
--ssl-ca=rds-cert.pem --ssl-mode=REQUIRED (MariaDB 10.2+)
```

- MySQL / MariaDB (older versions)

```
--ssl-ca=rds-cert.pem --ssl-verify-server-cert
```

RDS Encryption At Rest

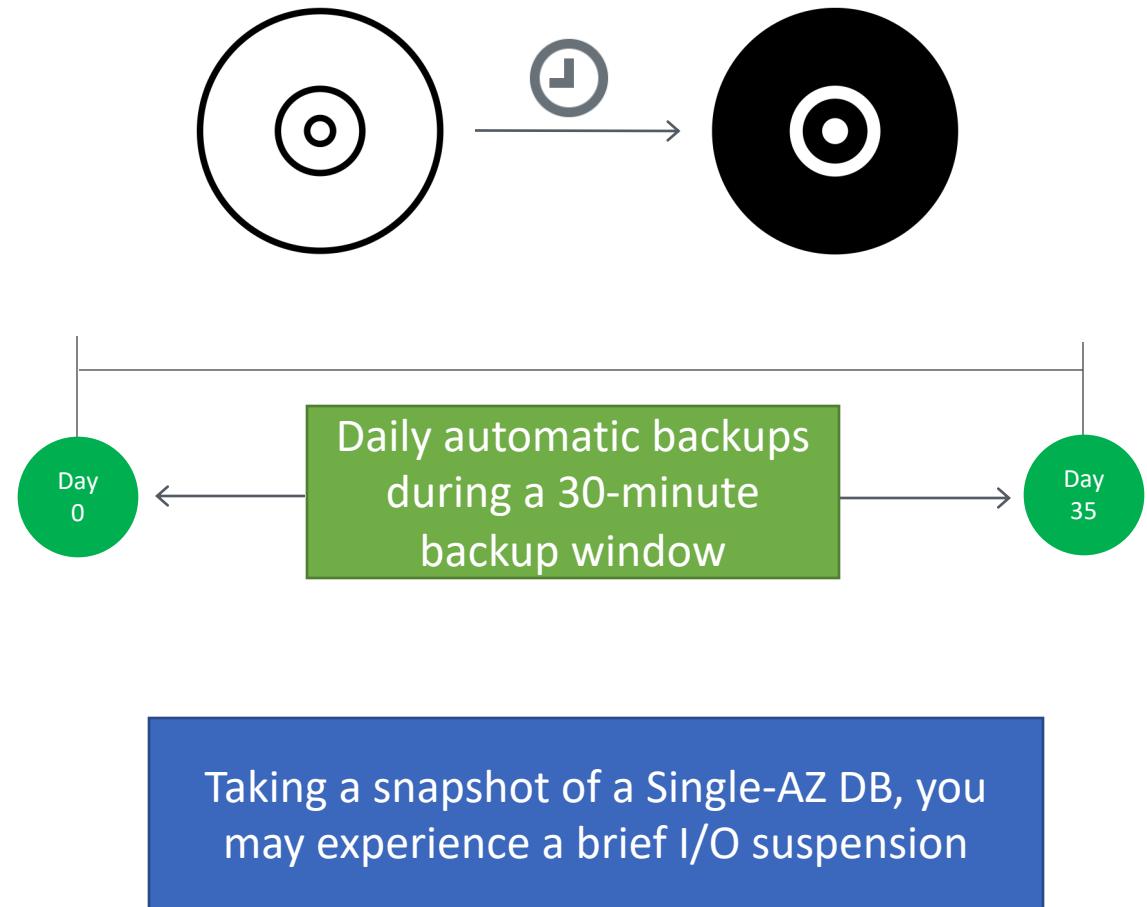
- RDS supports AES-256 encryption algorithm
- Keys managed through KMS
- Can encrypt both master and read replicas
- Encryption must be defined at launch time
- RDS also supports TDE (Transparent Data Encryption)
 - For SQL Server (Enterprise Edition only) and Oracle DB instances
 - Can be enabled using the option group
 - Use TDE option for Oracle
 - Use TRANSPARENT_DATA_ENCRYPTION option for SQL Server
- TDE be used together with encryption at rest
- Might slightly affect DB performance if you use TDE and encryption at rest simultaneously



KMS

RDS Backups

- RDS supports automatic backups
- Capture transaction logs in real time
- Enabled by default with a 7-days retention period (0-35 days retention, 0=disable automatic backups)
- You can provide backup window (daily time range) and backup retention period (no. of days)
- The first backup is a full backup. Subsequent backups are incremental
- Data is stored in a S3 bucket (owned and managed by RDS service, you won't see them in your S3 console)
- Recommended to use Multi-AZ option to avoid performance issues when backups are running
- Integrates with AWS Backup service for centralized management



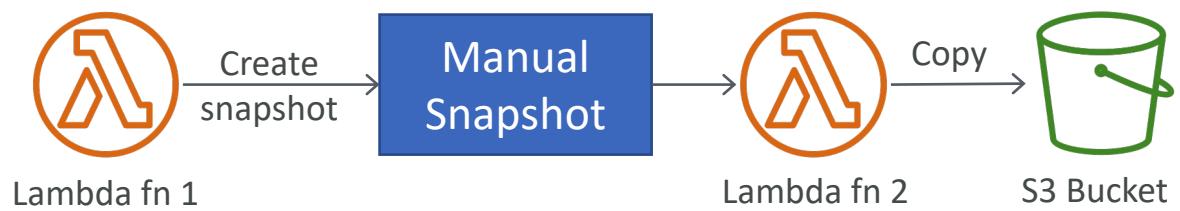
Backups vs Snapshots

Backups

- Are automated
- Are incremental
- Retention period up to 35 days
- **Support PITR** within retention period
- Great for unexpected failures
- A non-zero backup retention period in RDS also enables a snapshot before and after DB engine upgrades

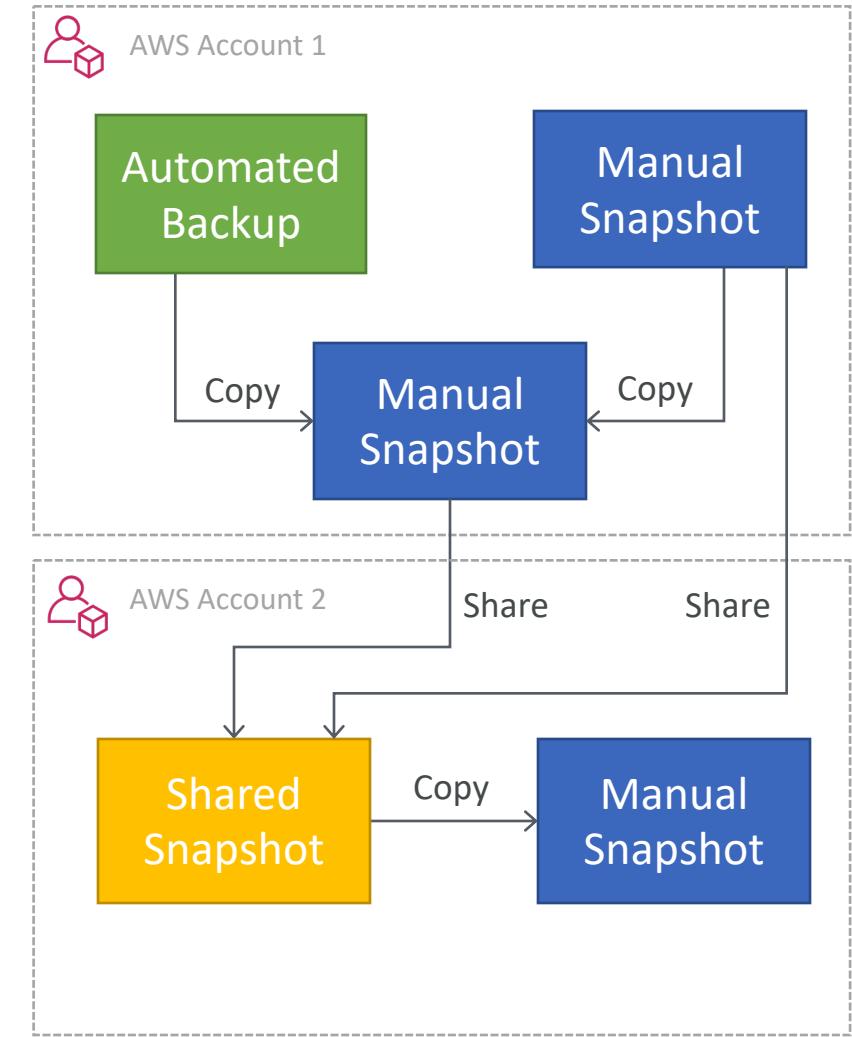
Snapshots

- Are manually triggered
- Are full backups
- Retained as long as you want
- Does not support PITR
- Great for known events like DB upgrades etc.
- Can use Lambda functions to take periodic backups and move them to S3 (say for compliance purposes)



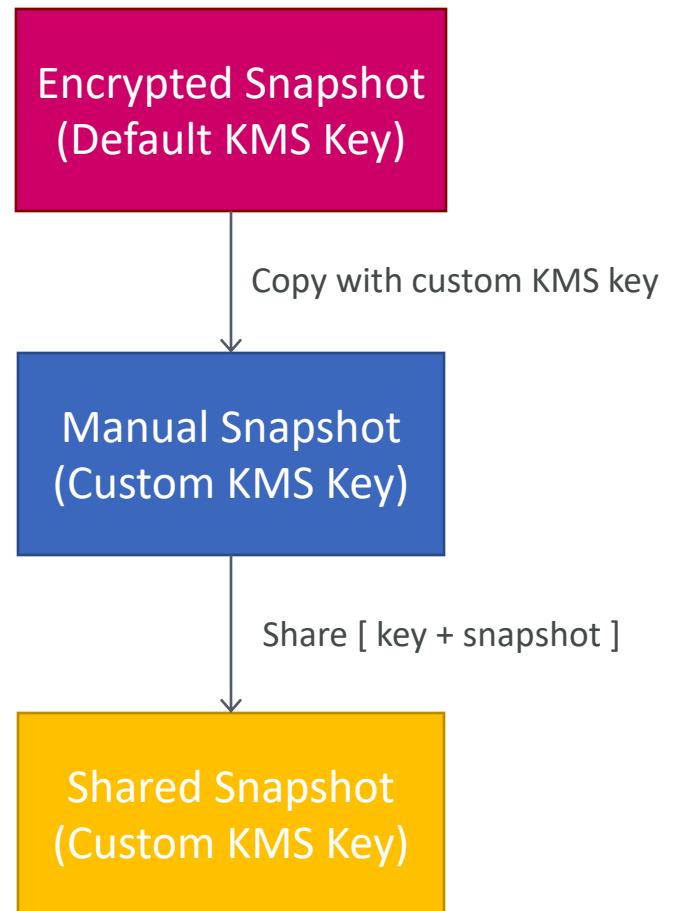
Copying and sharing RDS snapshots

- You can copy automated backups or manual snapshots
- The copy becomes a manual snapshot
- You can copy snapshots within region, across regions or across accounts
- For copying snapshot across account, you must share the snapshot first, and then copy it in the target account
- Automated backups cannot be shared directly. Must snapshot first.
- Copying across regions/accounts = data transfer costs



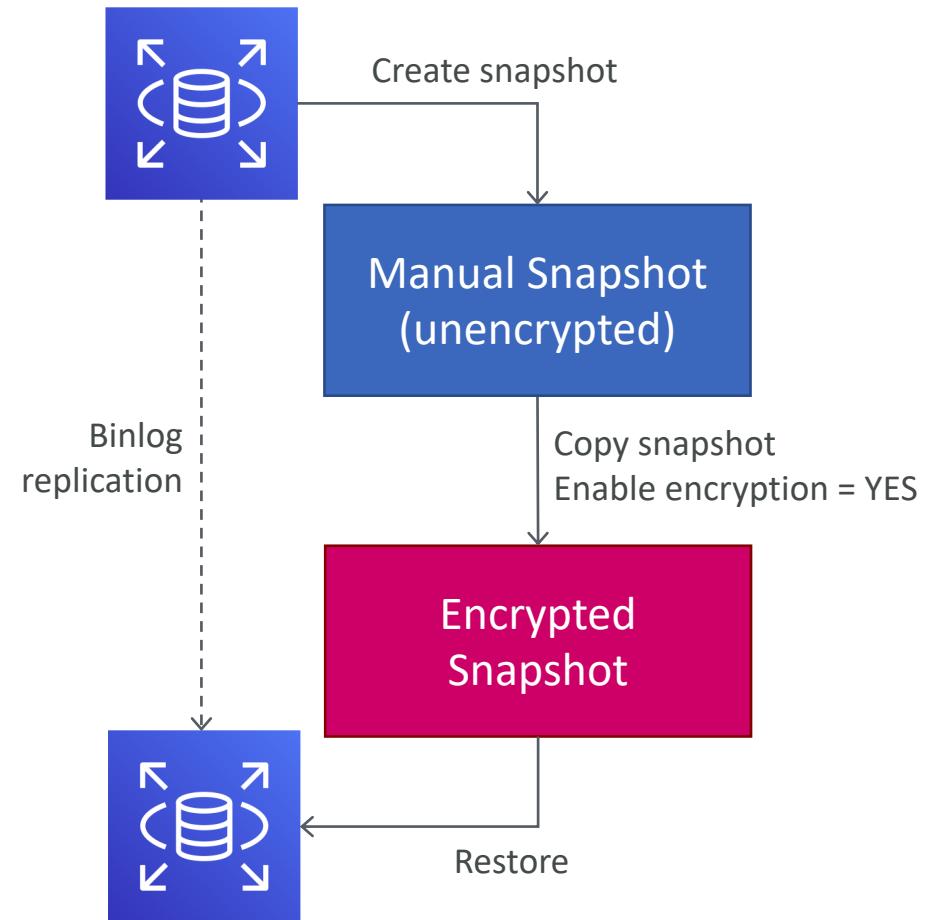
Copying and sharing RDS encrypted snapshots

- Snapshots encrypted w/ default RDS encryption key cannot be shared directly
 - Copy the snapshot using a custom encryption key and then share [key + snapshot]
- Snapshots with certain custom option groups cannot be shared (e.g.TDE)



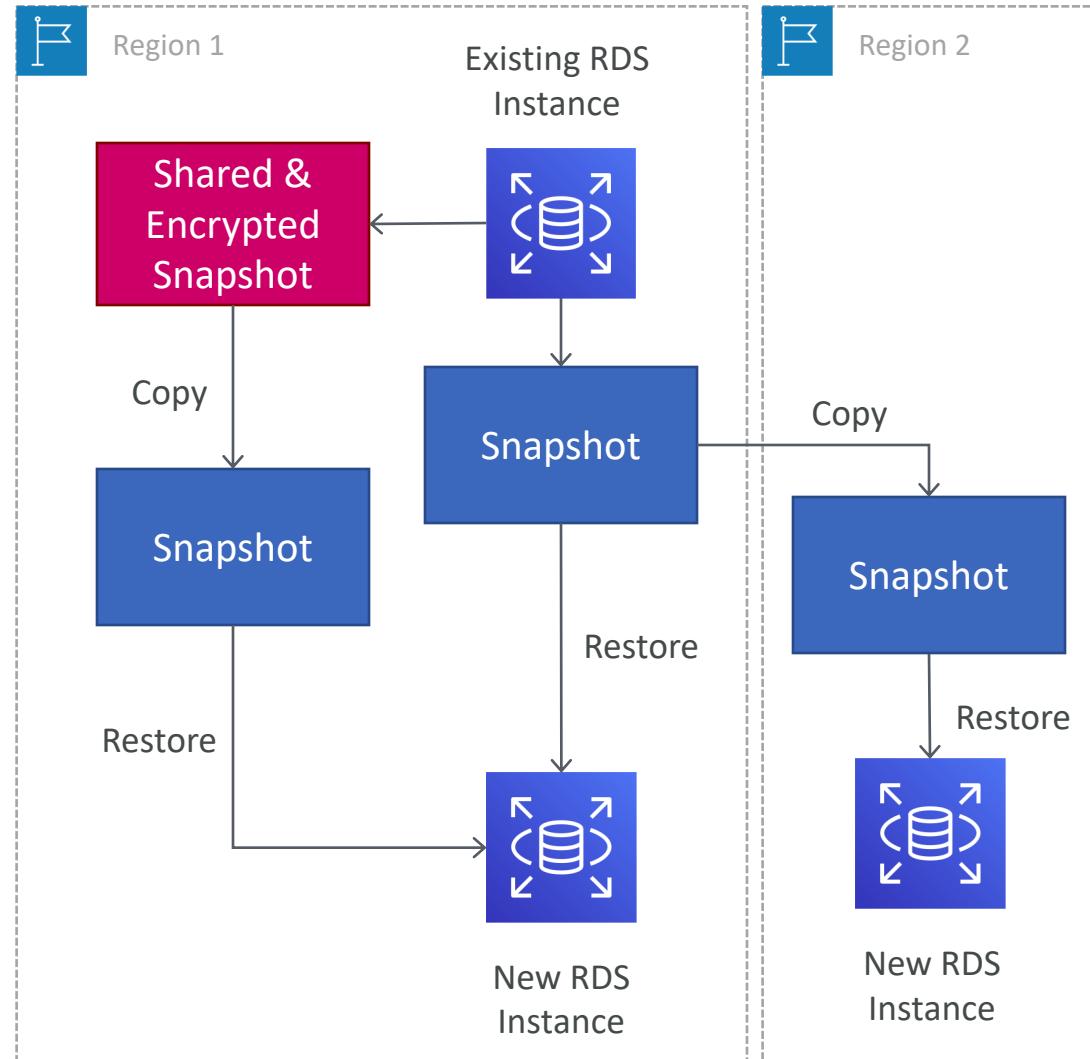
How to encrypt an unencrypted RDS DB

- Can't encrypt an existing unencrypted RDS DB instance
- Can't create an encrypted read replica from an unencrypted instance
- Copy an unencrypted snapshot with encryption enabled
- Restore the encrypted snapshot to a new RDS DB instance
- Can use MySQL replication to synchronize changes (binlog replication)
- Sidenote – if it's an Aurora unencrypted snapshot, then you can directly restore it to an encrypted aurora DB by specifying the KMS key. No need to copy the snapshot.



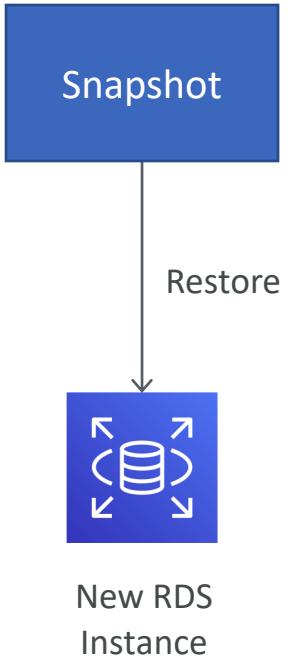
Restoring from a snapshot

- Can only restore to a **new instance**
- An instance can have one or more DBs and all these DBs will be restored
- To retain the same name, delete or rename the existing instance first
- Can't restore from a shared and encrypted snapshot directly (Copy first and then restore from copy)
- Can't restore from another region directly (Copy first and then restore from copy)
- Can restore from a snapshot of DB instance outside VPC to inside VPC (but not other way round)



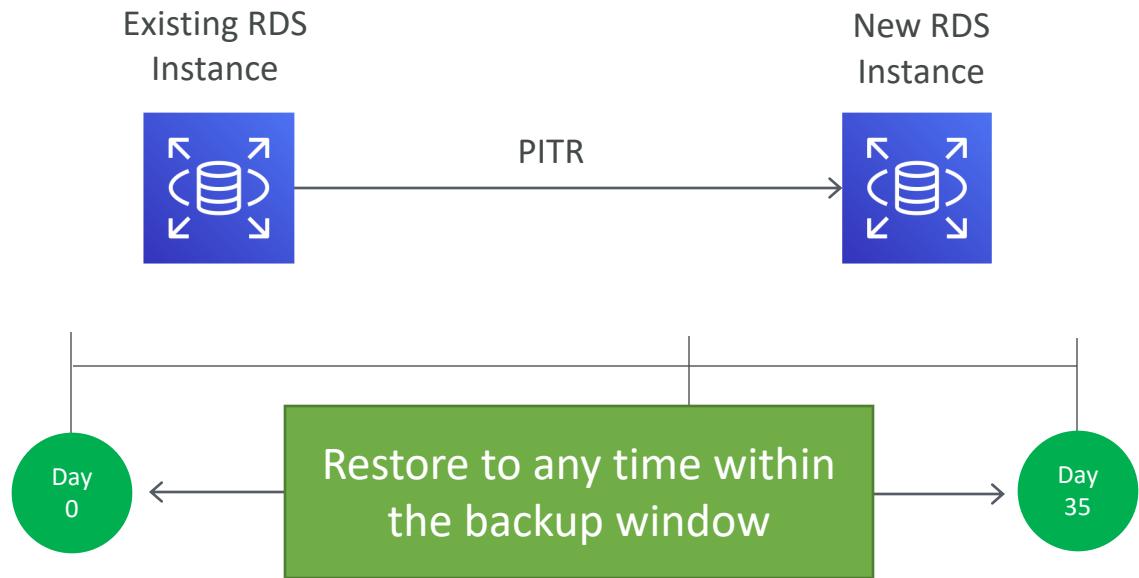
Restoring from a snapshot

- By default, restored cluster gets applied with
 - New security group
 - Default parameter group
 - Option group that was associated with the snapshot
- While restoring from a snapshot, be sure to
 - Choose the correct security group to ensure connectivity for the restored DB
 - Choose correct parameter group for the restored DB
 - Recommended to retain parameter group of the snapshot to help restore with the correct parameter group



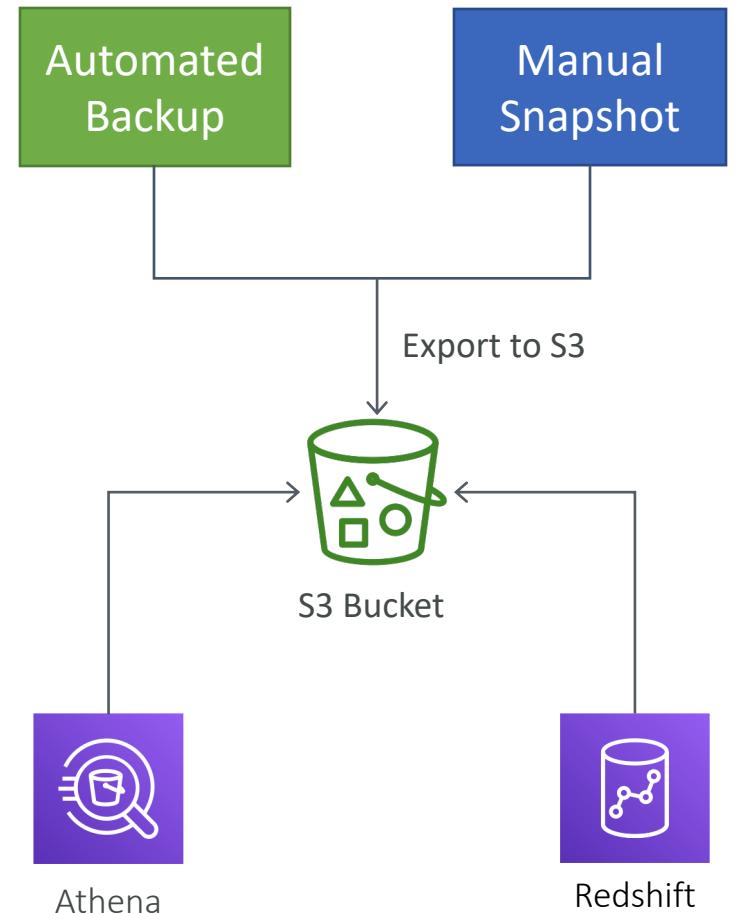
PITR with RDS

- Point-In-Time Recovery
- Can only restore to a new instance
- The backup retention period controls the PITR window
- Can restore to any point in time during your backup retention period
- RDS uploads DB transaction logs to S3 every 5 minutes (affects latest restorable time)
- You can move/restore a DB instance from outside VPC to inside VPC with PITR (but not other way round)



Exporting DB Snapshot Data to S3

- All types of backups can be exported (automatic/manual or those created with AWS Backup service)
- How to export?
 - Setup an S3 bucket with appropriate IAM permissions and create a KMS key for SSE
 - Export the snapshot using console (Actions → Export to Amazon S3) or using start-export-task CLI command
- Export runs in the background
- Doesn't affect the DB performance
- Data exported in Apache Parquet format (=compressed and consistent)
- Allows you to analyze the DB data using Athena or Redshift Spectrum



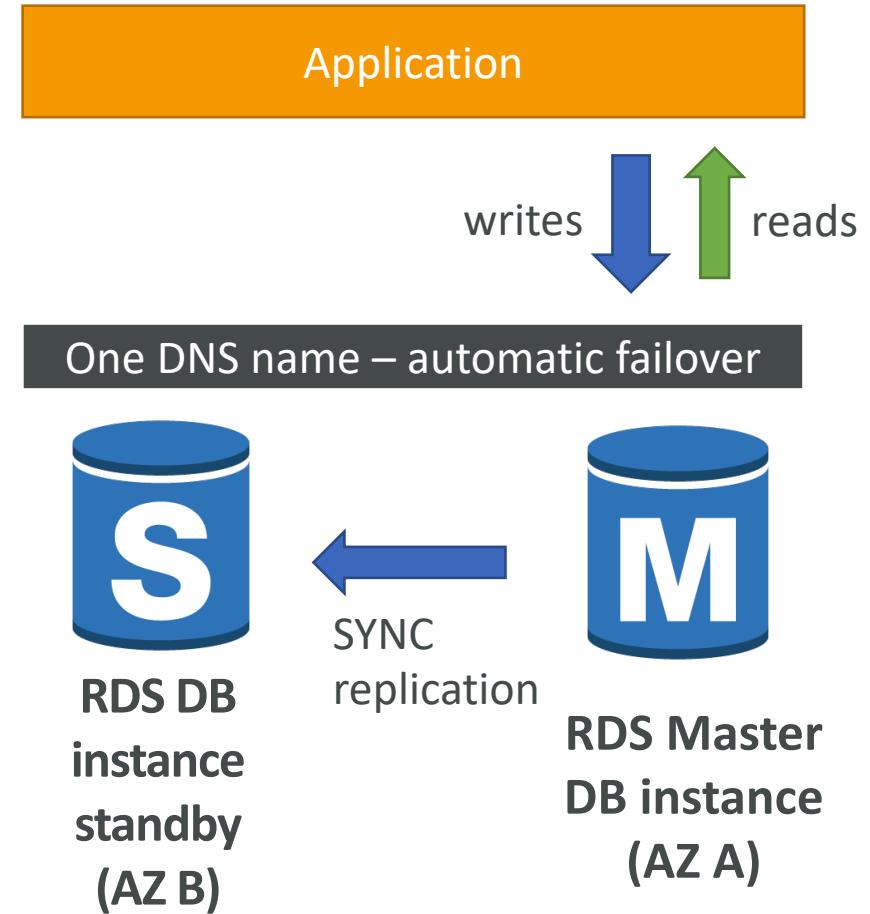
Backup and Restore in RDS (MySQL)



Demo

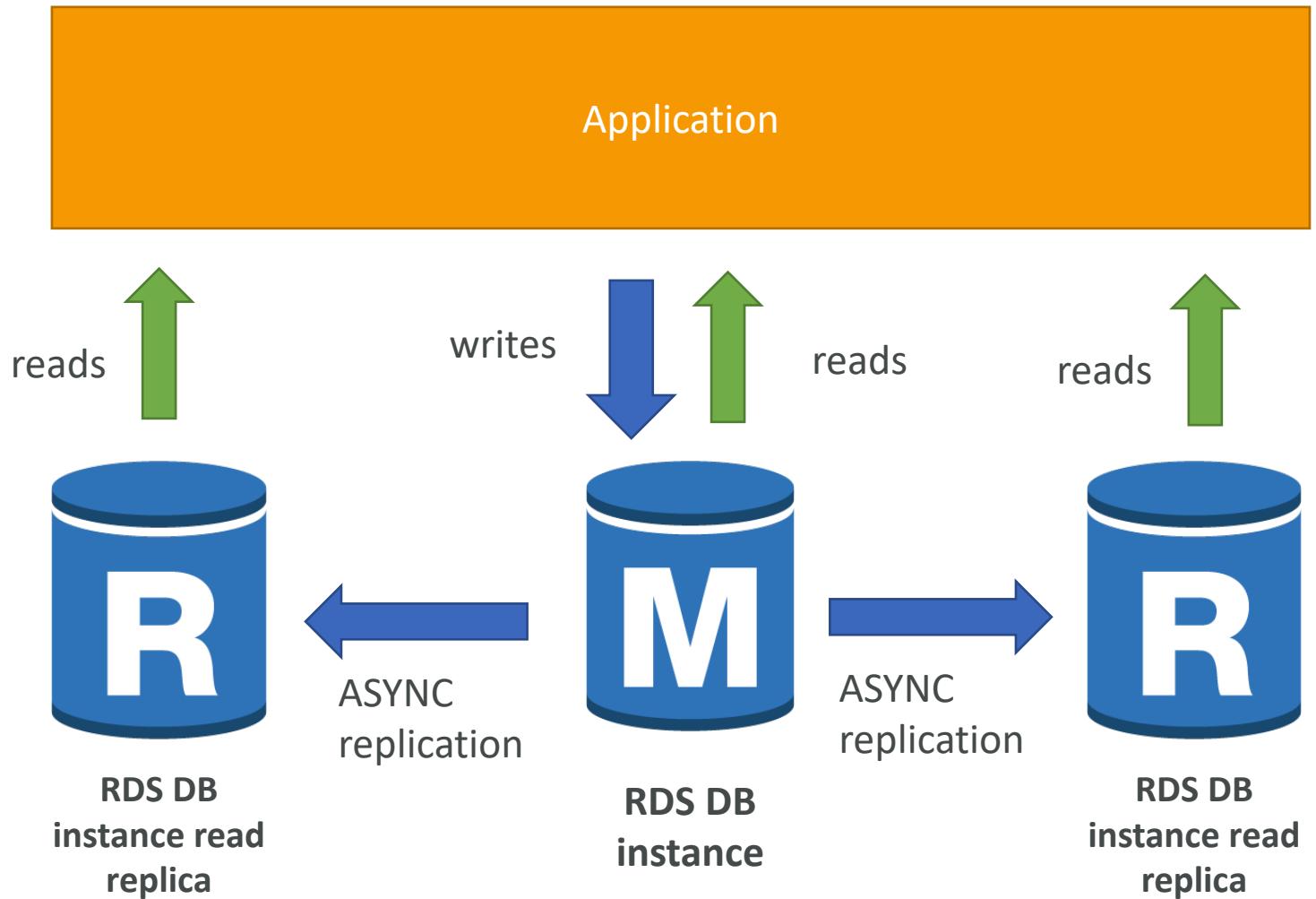
Multi-AZ Deployments in RDS

- For high availability, data durability and fault-tolerance (not used for scaling)
- Offers **SYNC** replication to standby instance in another AZ over low latency links
- Performs **automatic failover** to standby instance in another AZ in case of planned or unplanned outage
- Uses DNS routing to point to the new master (no need to update connection strings)
- Failover times (RTO) are typically 60-120 seconds (minimal downtime)
- Backups are taken from standby instead of primary to ensure performance level during backup activity
- Recommended for production use cases
- To force a failover or simulate **AZ-failure**, reboot the master instance and choose **Reboot with failover**



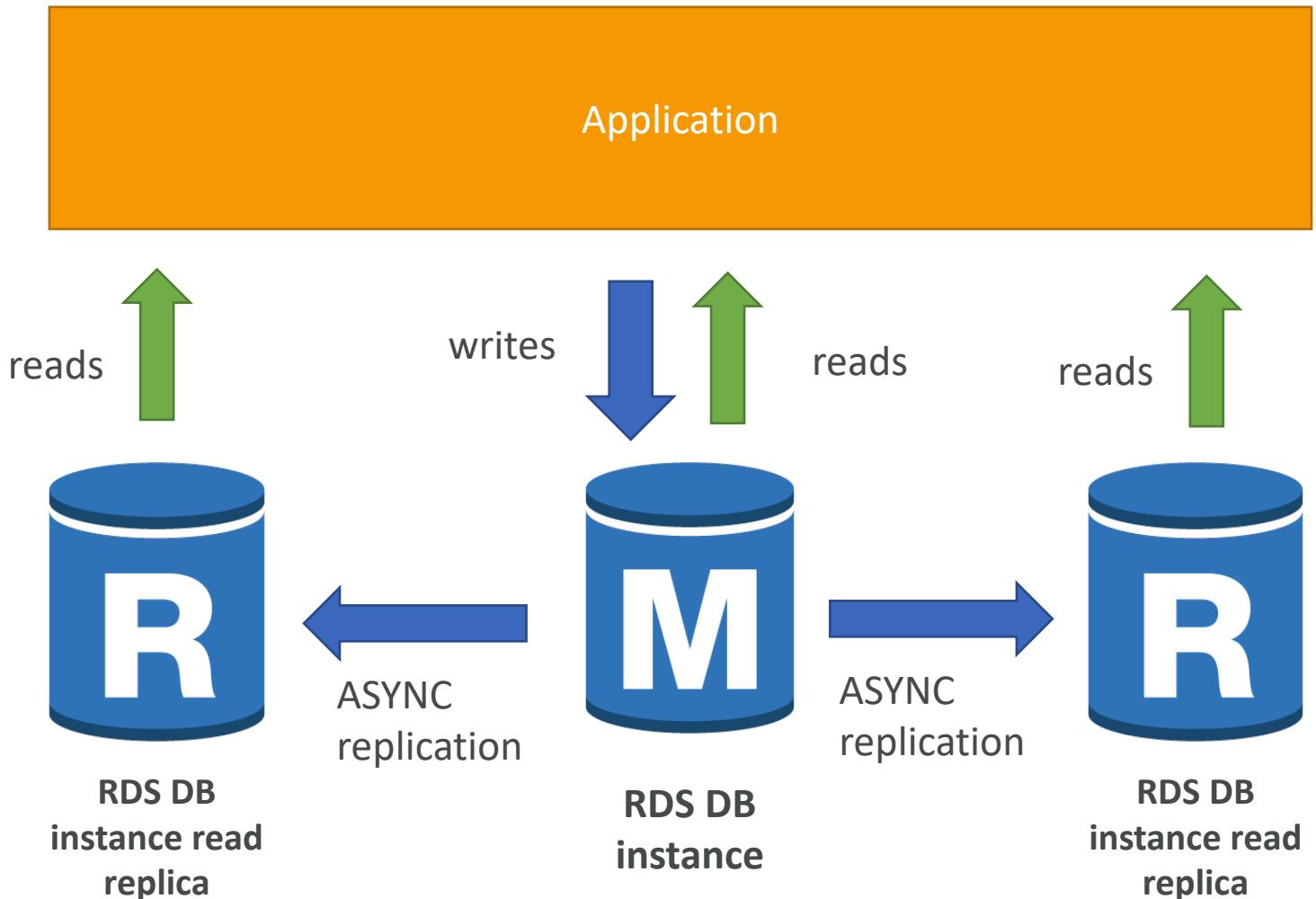
RDS Read Replicas

- Read-only copies of master (primary) DB instance
- Up to 5 Read Replicas
- Within AZ, Cross AZ or Cross Region
- Replication is **ASYNC**, so reads are eventually consistent
- Applications must update the connection string to leverage read replicas

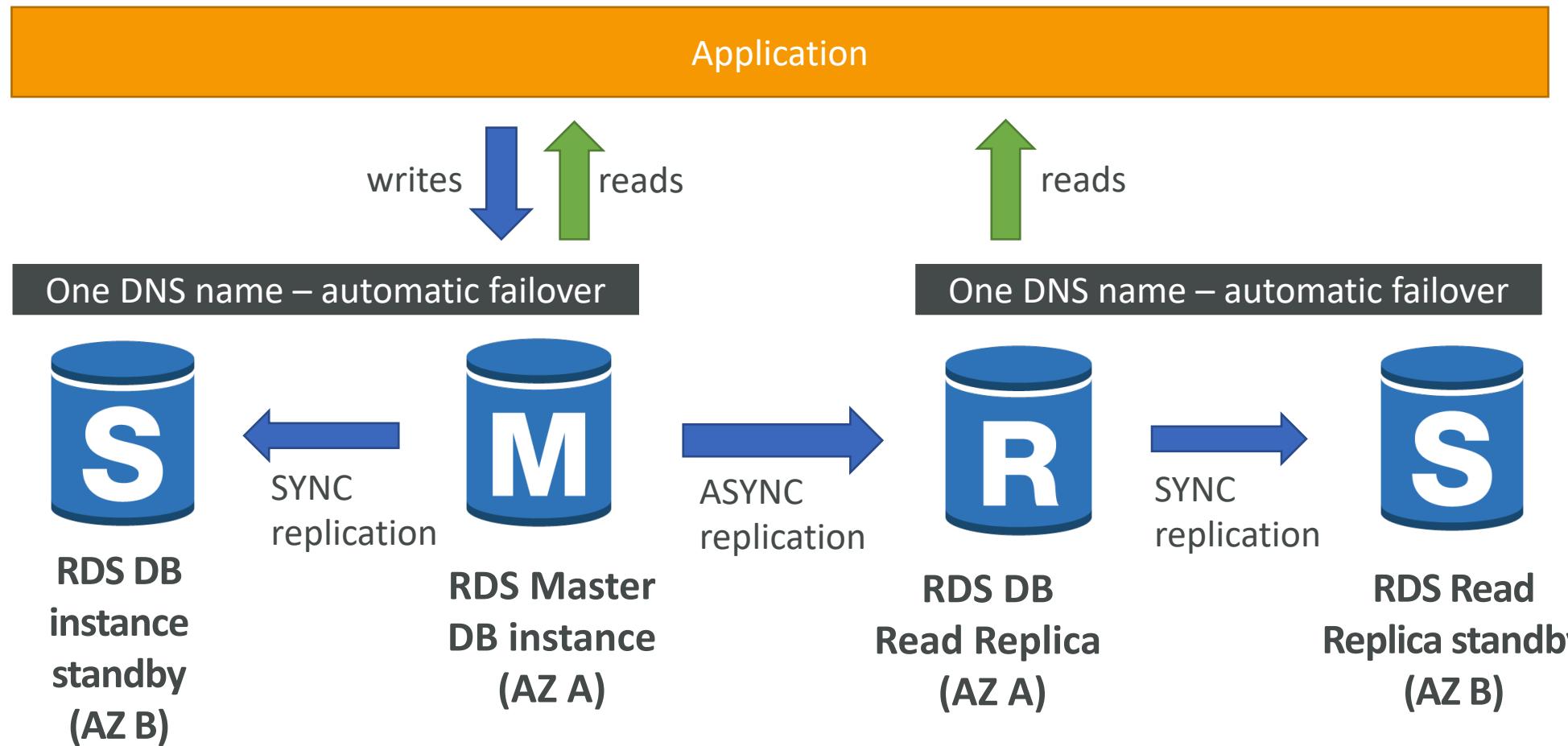


RDS Read Replicas

- Boost DB performance and durability
- Useful for scaling of read-heavy workloads
- Can be promoted to primary (complements Multi-AZ)
- To create a replica, you must enable automatic backups with at least one day retention period
- Replica can be Multi-AZ (= a replica with its own standby instance)

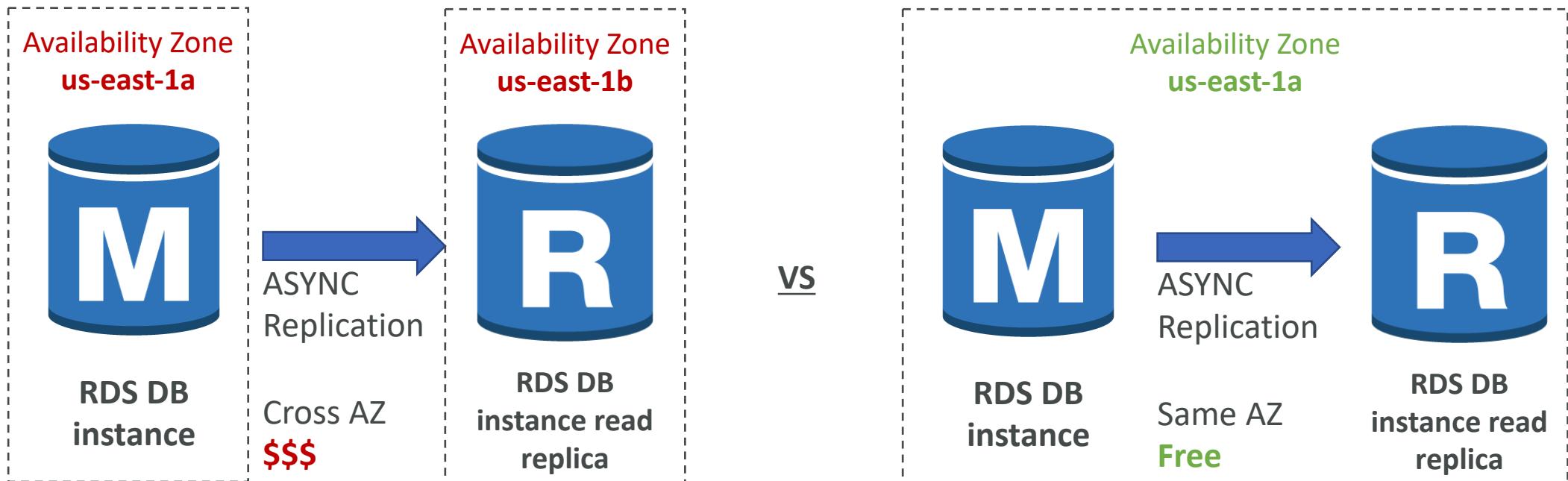


Multi-AZ Replicas in RDS



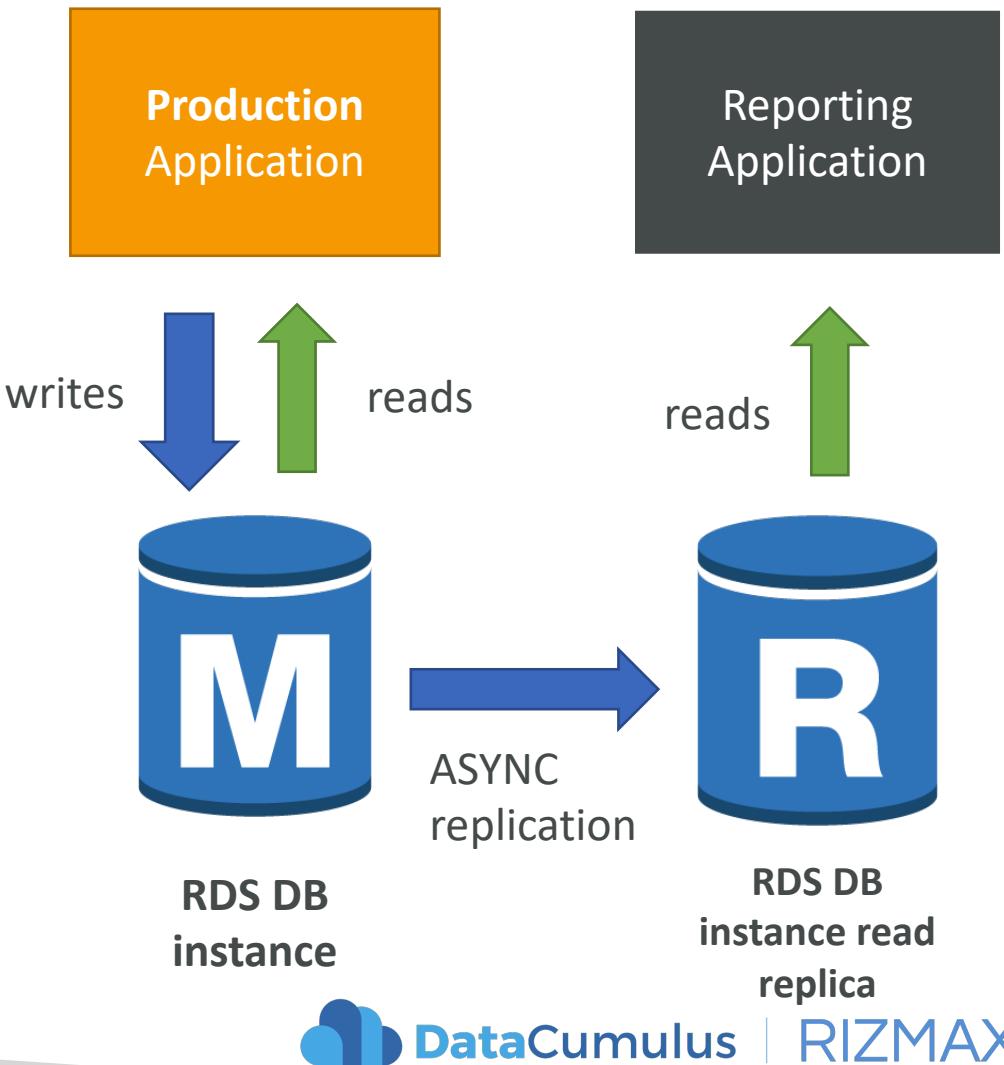
RDS Read Replicas as Multi-AZ

- Supported for MySQL / MariaDB / PostgreSQL / Oracle
- Works as a DR target. When promoted to primary, it works as Multi-AZ
- There's added network cost when data goes from one AZ to another



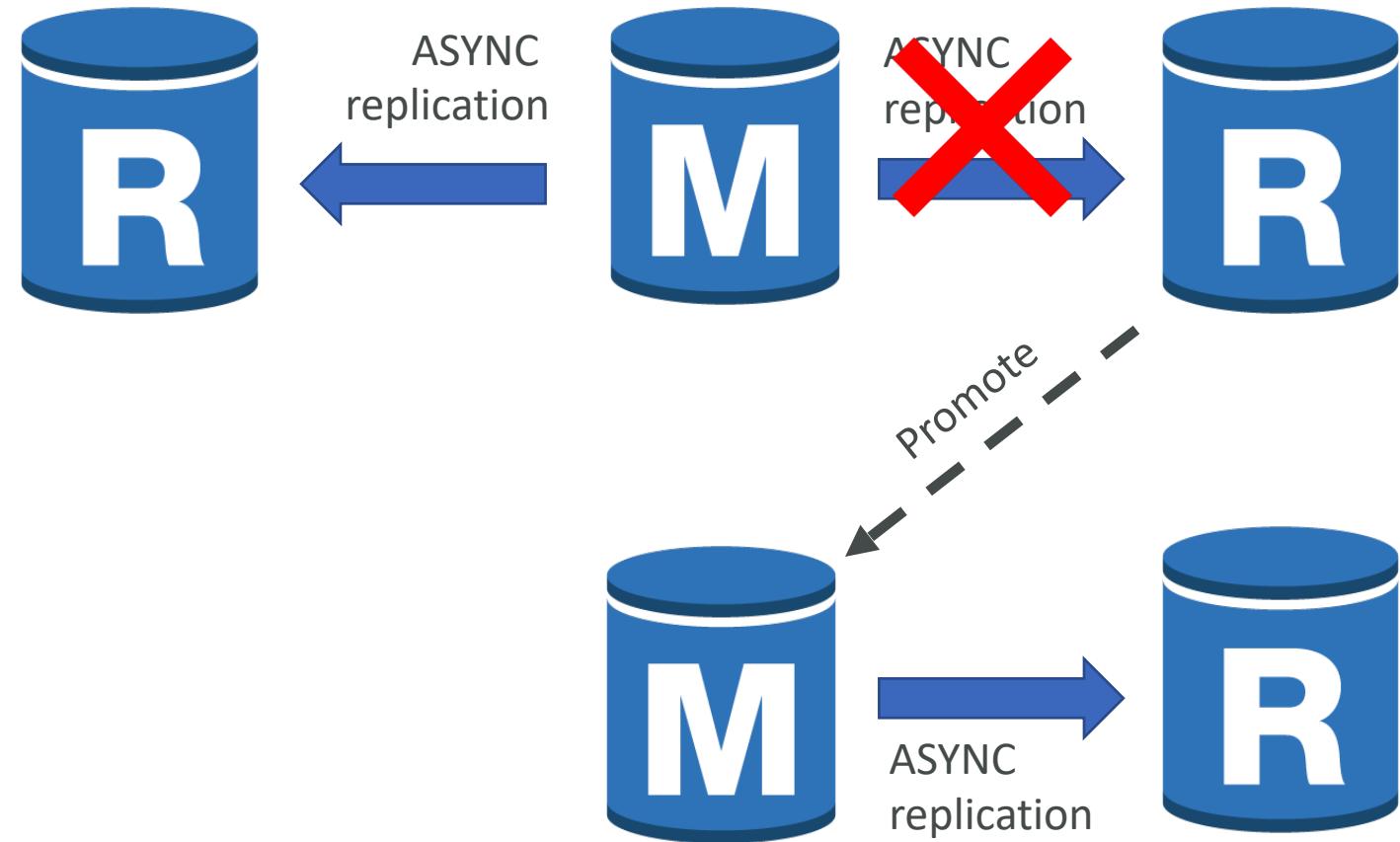
RDS Read Replicas – Use Case

- You have a production database that is taking on normal load
- You want to run a reporting application to run some analytics
- You create a Read Replica to run the new workload there
- The production application is unaffected
- Read replicas are used for SELECT (=read) only kind of statements (not INSERT, UPDATE, DELETE)



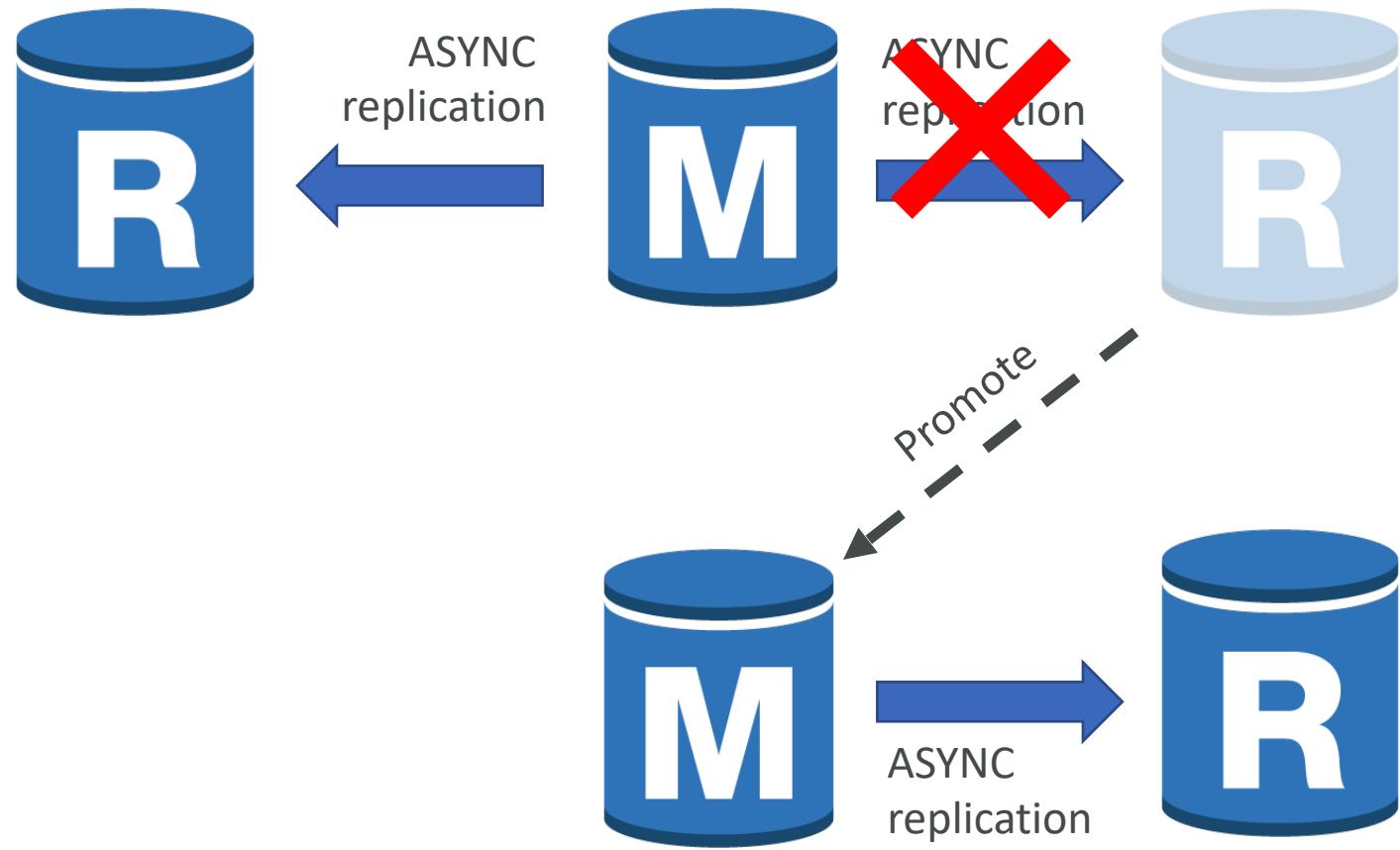
Promoting a Read Replica to a Standalone DB Instance

- Promoted instance is rebooted and becomes an independent DB instance (separate from its source)
- Will no longer work as a replica. Does not affect other replicas of the original DB instance
- You cannot promote a replica to a standalone instance while a backup is running



Promoting a Read Replica to a Standalone DB Instance – Use cases

- Use as a DR strategy
- Avoid performance penalty of DDL operations (like rebuilding indexes)
 - Perform DDL ops on a read replica and promote it to a standalone instance. Then point your app to this new instance.
 - Sharding (splitting a large DB into multiple smaller DBs)



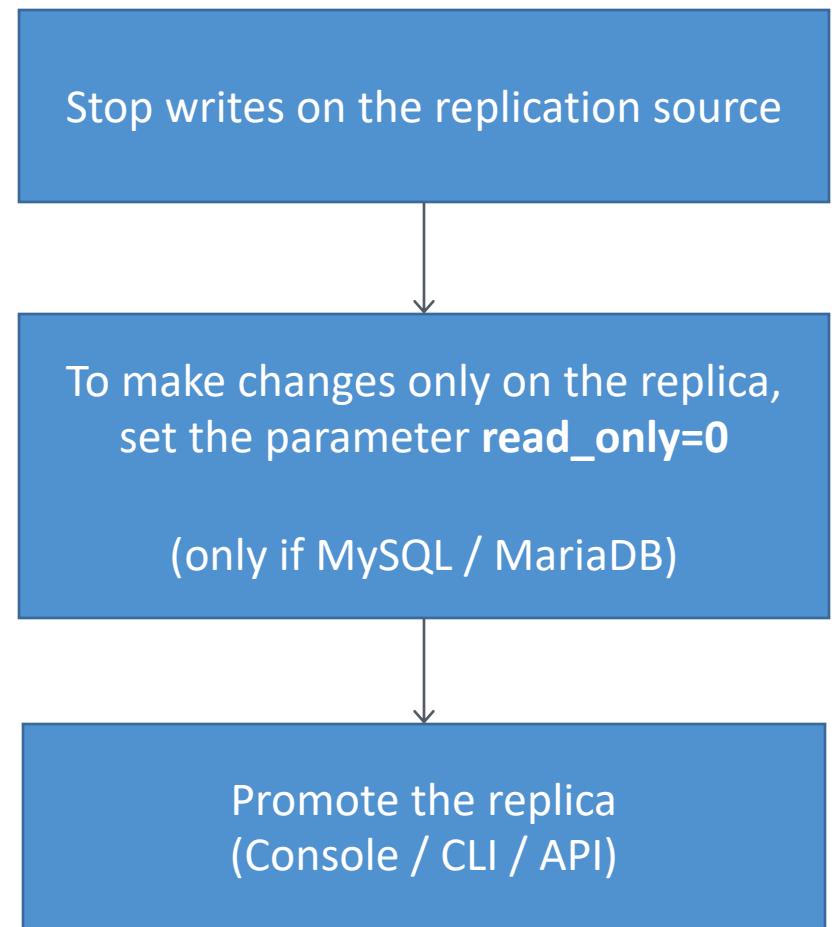
RDS Multi-AZ failover and replica promotion



Demo

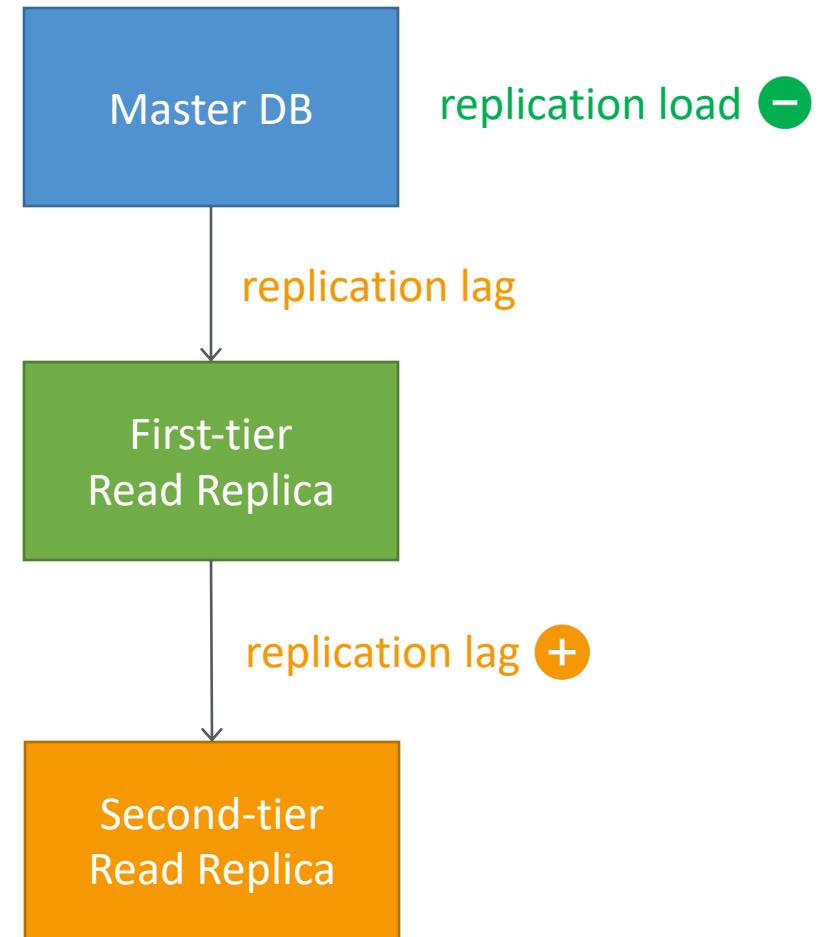
Enabling writes on a read replica

- For MySQL / MariaDB read replica, set the parameter `read_only = 0` for the read replica to make it writable
- You can then perform DDL operations on the read replica as needed without affecting the source DB
- Actions taken on the read replica don't affect the performance of the source DB instance
- You can then promote the replica to a standalone DB



RDS Read Replica capabilities

- Can create multiple read replicas in quick succession
- Can use DB snapshot to perform PITR of a Read Replica
- Can create a replica from an existing replica
 - reduces replication load from the master DB instance
 - second-tier replica can have higher replication lag



RDS Second-tier replicas and replica promotion



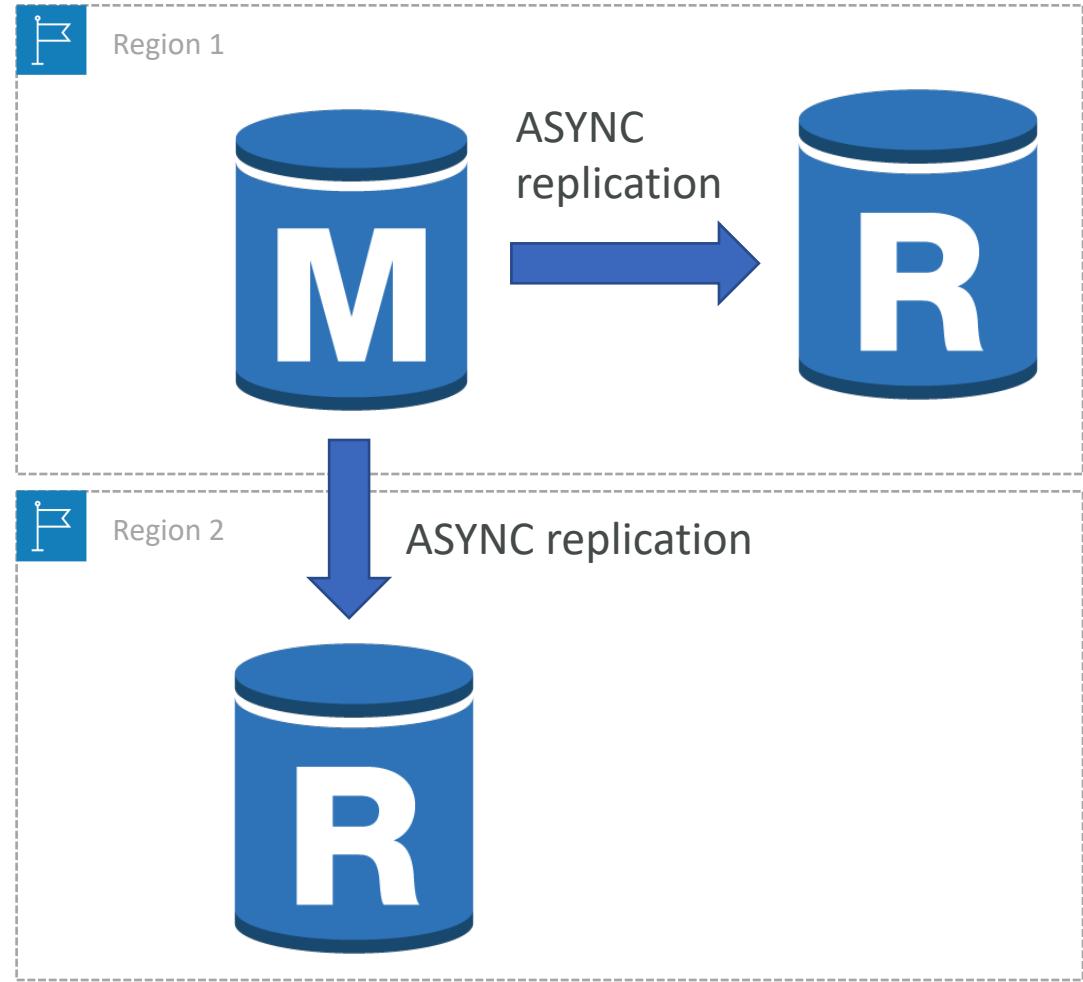
Demo



DataCumulus | RIZMAXed

Cross-Region Read Replicas in RDS

- Supported for MariaDB, MySQL, Oracle, and PostgreSQL
- Not supported for SQL Server
- Advantages
 - Enhanced DR capability
 - Scale read operations closer to the end-users
- Limitations
 - Higher replica lag times
 - AWS does not guarantee more than five cross-region read replica instances



RDS replicas with an external database

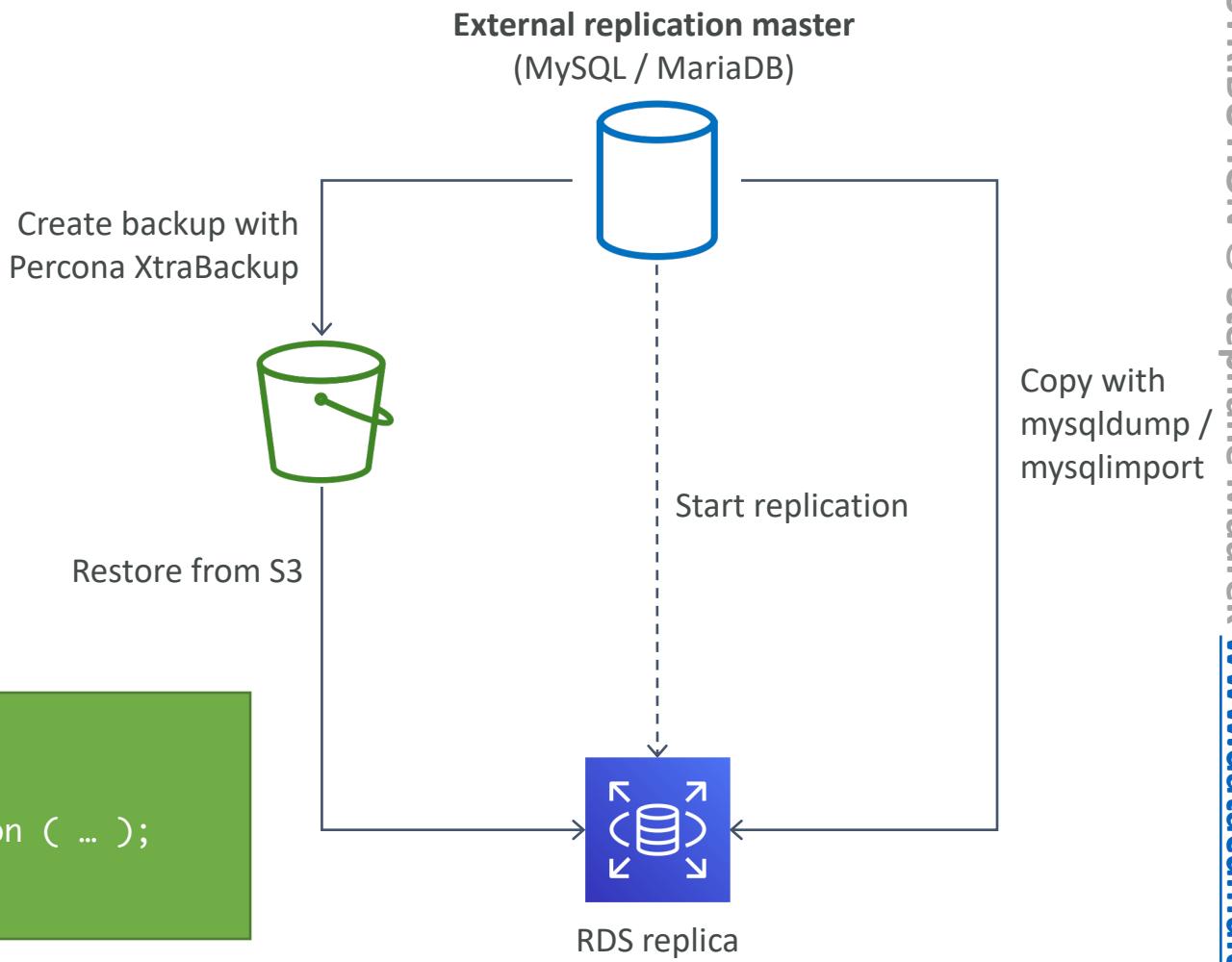
- Replication b/w an external DB and an RDS replica
- Supported for MySQL / MariaDB engines
- Two ways
 - Binlog replication
 - GTID based Replication

Binary Log File Position Replication

```
CALL mysql.rds_set_external_master (... );
CALL mysql.rds_start_replication;
```

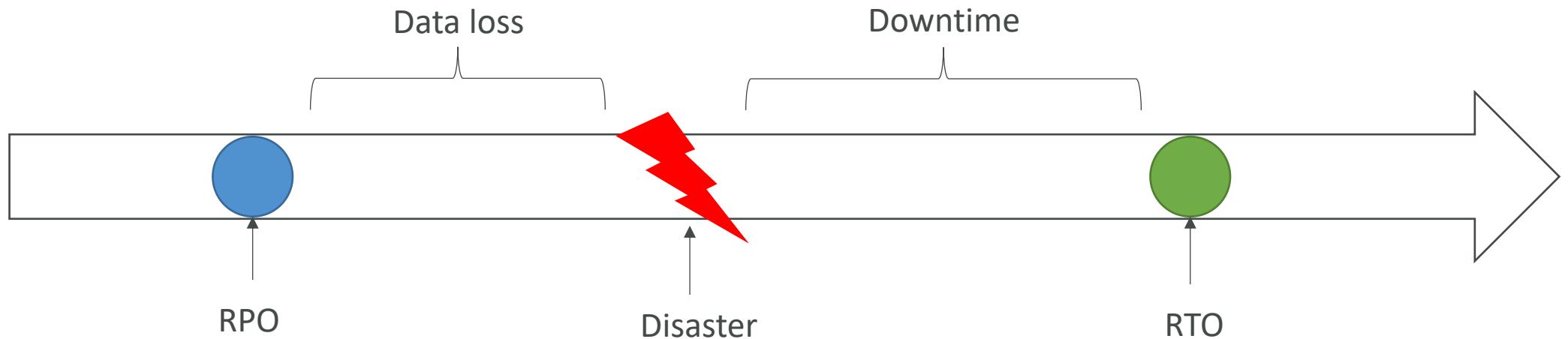
GTID-based Replication

```
CALL mysql.rds_set_external_master_with_auto_position (... );
CALL mysql.rds_start_replication;
```



RDS Disaster Recovery Strategies

- To ensure business continuity despite unexpected failures/events
- Multi-AZ is not enough (it can't protect from logical DB corruption, malicious attacks etc.)
- Key metrics for DR plan – RTO and RPO
- RDS PITR offers RPO of 5 minutes (typically)
- RTO (Recovery time objective)
 - How long it takes you to recover after a disaster
 - Expressed in hours
- RPO (Recovery point objective)
 - How much data you could lose due to a disaster
 - Expressed in hours (e.g. RPO of 1 hour means you could lose an hour worth of data)



Comparing RDS DR Strategies

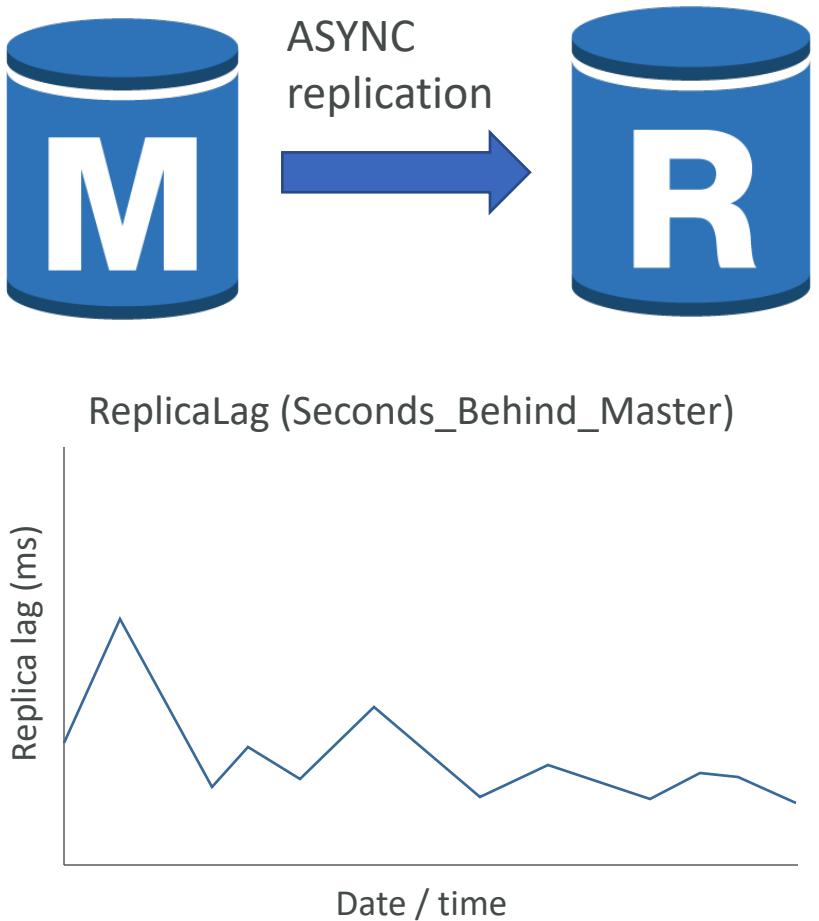
	RTO	RPO	Cost	Scope
Automated backups	Good	Better	Low	Single Region
Manual snapshots	Better	Good	Medium	Cross-Region
Read replicas	Best	Best	High	Cross-Region

- Replica lag – the amount of time that the replica is behind the source DB
- Replica lag can impact your recovery
- Failover to an RDS read replica is a manual process (not automated)
- A good DR plan should include a combination of backups, replicas and Multi-AZ/Multi-region deployment

Source: <https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/>

Troubleshooting high replica lag

- Asynchronous logical replication typically results in replica lag
- You can monitor ReplicaLag metrics in CloudWatch
- **ReplicaLag** metric reports `Seconds_Behind_Master` values
- Replication delays can happen due to:
 - Long-running queries on the primary instance (slow query log can help)
 - Insufficient instance class size or storage
 - Parallel queries executed on the primary instance



Troubleshooting replication errors

Recommendations:

- Size the replica to match the source DB (storage size and DB instance class)
- Use compatible DB parameter group settings for source DB and replica
- Ex. max_allowed_packet for read replica must same as that of the source DB instance
- Monitor the Replication State field of the replica instance
- If Replication State = Error, then see error details in the Replication Error field
- Use RDS event notifications to get alerts on such replica issues

Troubleshooting replication errors (contd.)

- Writing to tables on a read replica
 - Set `read_only=0` to make read replica writable
 - Use `only` for maintenance tasks (like creating indexes only on replica)
 - If you write to tables on read replica, it might make it incompatible with source DB and break the replication
 - So set `read_only=1` immediately after completing maintenance tasks
- Replication is only supported with transactional storage engines like InnoDB. Using engines like MyISAM will cause replication errors
- Using unsafe nondeterministic queries such as `SYSDATE()` can break replication
- You can either skip replication errors (if its not a major one) or delete and recreate the replica

Troubleshooting MySQL read replica issues

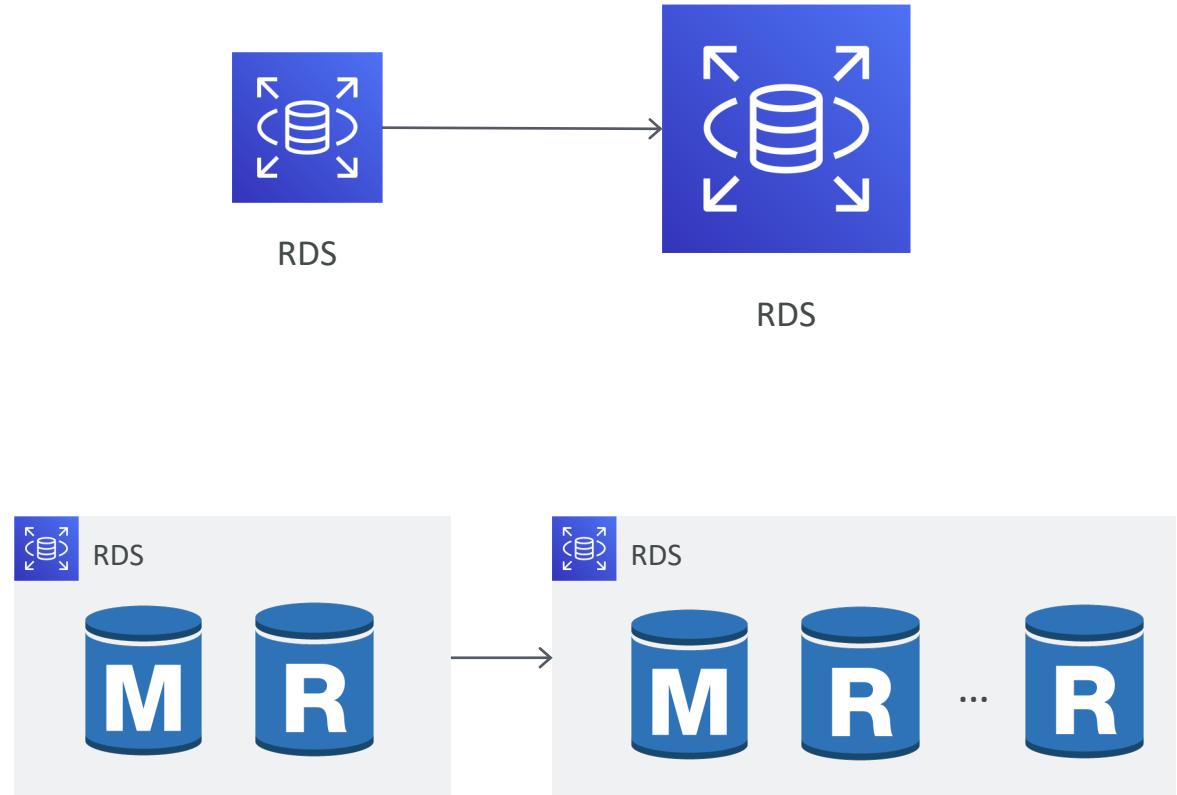
- Errors or data inconsistencies b/w source instance and replica
 - Can happen due to binlog events or InnoDB redo logs aren't flushed during a replica or source instance failure
 - Must manually delete and recreate the replica
- Preventive recommendations:
 - sync_binlog=1
 - innodb_flush_log_at_trx_commit=1
 - innodb_support_xa=1
- These settings might reduce performance (so test before moving to production)

Performance hit on new read replicas

- RDS snapshots are EBS snapshots stored in S3
- When you spin up a new replica, its EBS volume loads lazily in the background
- This results in first-touch penalty (when you query any data, it takes longer to retrieve it for the first time)
- Suggestions:
 - If DB is small, run “SELECT * FROM <table>” query on each table on the replica
 - Initiate a full table scan with VACUUM ANALYZE (in PostgreSQL)
- Another reason could be an empty buffer pool (cache for table and index data)

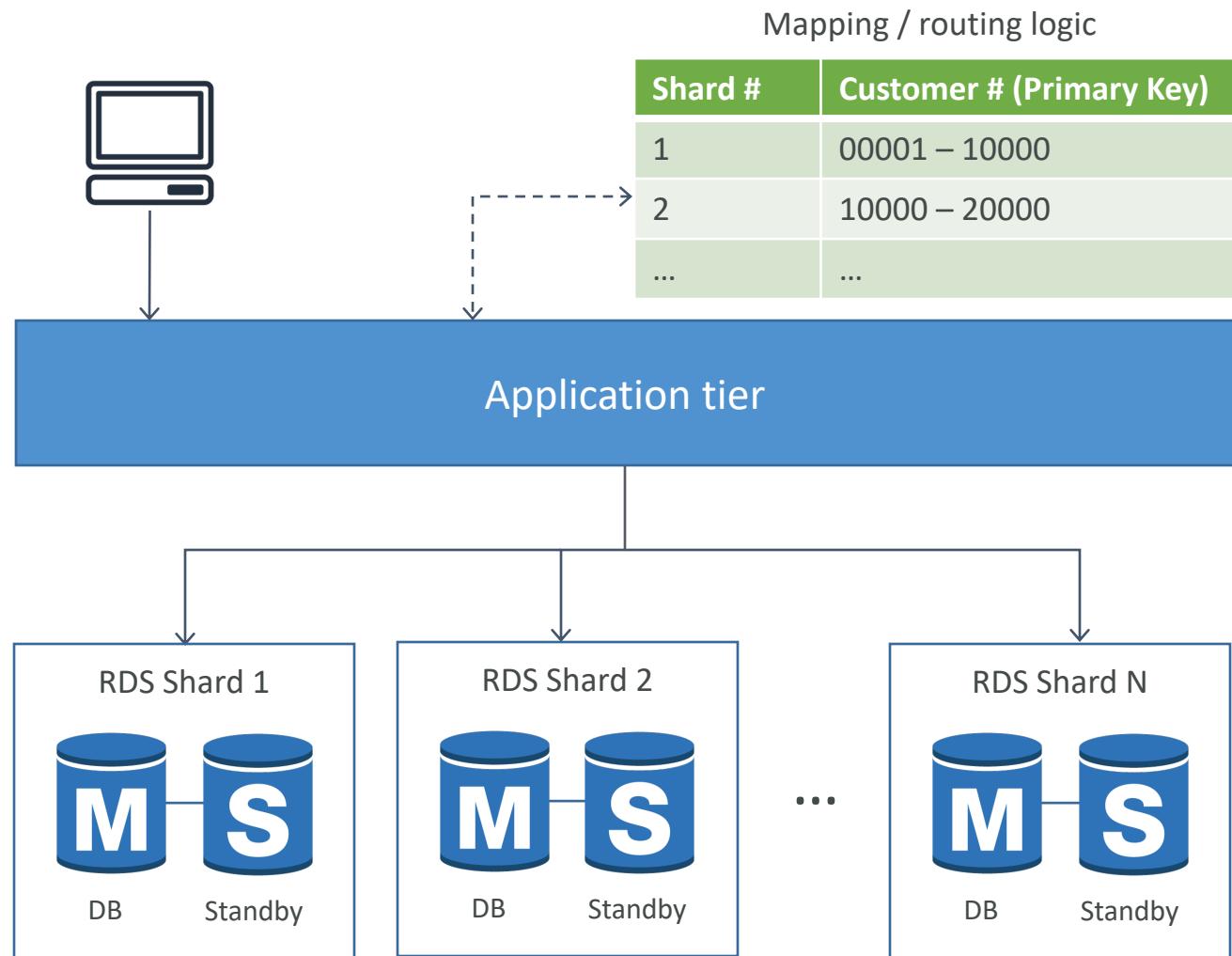
Scaling in RDS

- Vertical Scaling (Scaling up)
 - Single-AZ instance will be unavailable during scaling op
 - Multi-AZ setup offers minimal downtime during scaling op – standby DB gets upgraded first and then primary will failover to the upgraded instance
- Horizontal Scaling (Scaling out)
 - Useful for read-heavy workloads
 - Use read-replicas
 - Replicas also act as a DR target



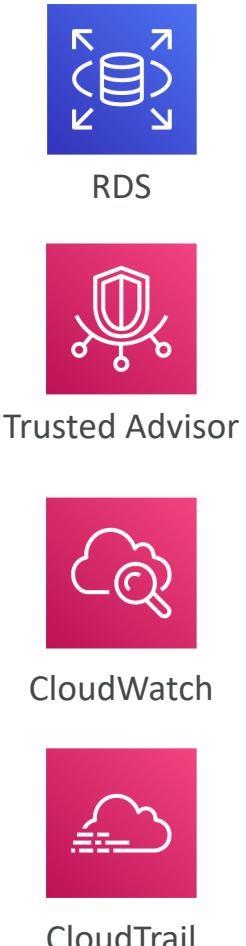
Sharding in RDS

- Sharding = horizontal partitioning
- Split and distribute data across multiple DBs (called shards)
- Mapping / routing logic maintained at application tier
- Offers additional fault tolerance (since no single point of failure)
- If any shard goes through failover, other shards are not impacted



RDS Monitoring

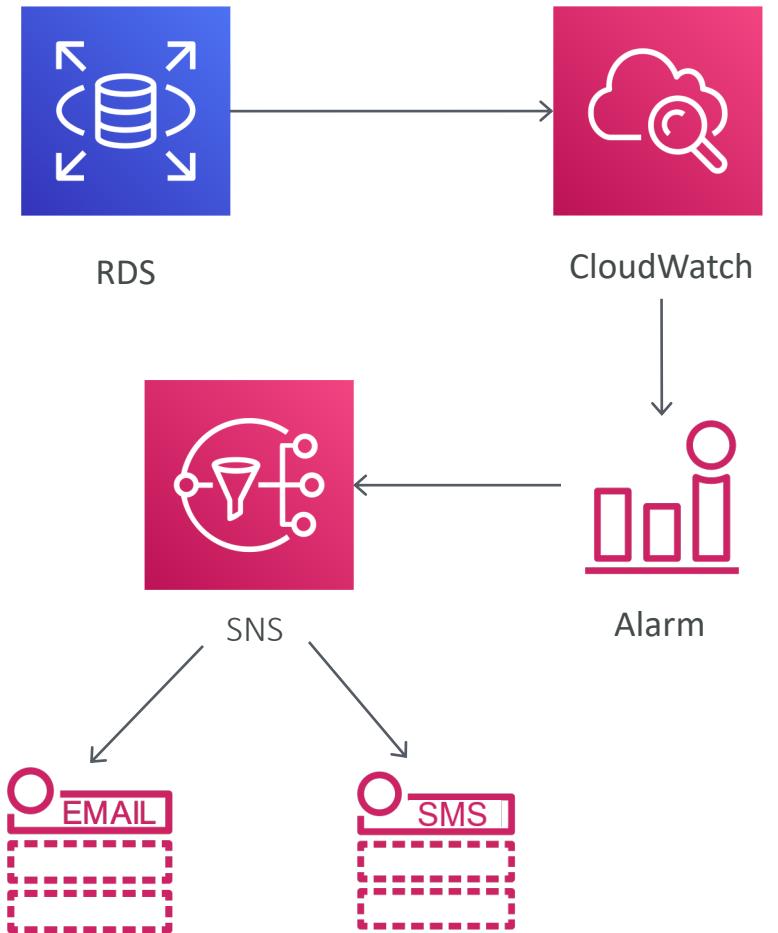
- Common metrics
 - CPU, RAM, disk space consumption / Network traffic / DB connections / IOPS metrics
- Native logs / extensions
 - e.g. pgaudit extension in PostgreSQL for auditing (DML / DCL / DDL etc)
- Manual Monitoring Tools
 - RDS console (DB connections, R/W ops, storage consumption, memory utilization, N/W traffic)
 - AWS Trusted Advisor (cost optimization, security, fault tolerance, performance improvement checks)
 - CloudWatch (service health status etc.)



- Automated Monitoring Tools
 - RDS event notifications
 - Database logs (can be exported to CloudWatch Logs)
 - CloudWatch (Metrics / Alarms / Logs)
 - Enhanced Monitoring (real-time)
 - Performance Insights
 - RDS Recommendations
 - CloudTrail (captures all RDS API calls, can be viewed in CloudTrail console or delivered to an S3 bucket)
 - Up to 90 days of your account activity can be viewed in CloudTrail console (can create a trail to deliver the audit logs to S3)

RDS Notifications / Event subscriptions

- Available within the RDS console
- Allows you to create CloudWatch alarms to notify you whenever certain metric data crosses a threshold
- You can send alarm notifications to an SNS topic (email / SMS)
- You can also subscribe to RDS events
- Event sources can be snapshots, instances, security groups, parameter groups, clusters, cluster snapshots, etc.
- Events like DB instance creation, deletion, availability (shutdown / restart), backup, recovery, failover, failure, backtrack, config change etc.



RDS Recommendations

- Periodic automated suggestions for DB instances, read replicas, and DB parameter groups

The screenshot shows the AWS RDS Recommendations page. At the top, there are four tabs: Active (5), Dismissed (0), Scheduled (0), and Applied (0). The Active tab is selected.

Enhanced Monitoring is not enabled (3)
DB instances that don't have Enhanced Monitoring enabled. Enhanced Monitoring provides real-time operating system metrics for monitoring and troubleshooting. [Info](#)

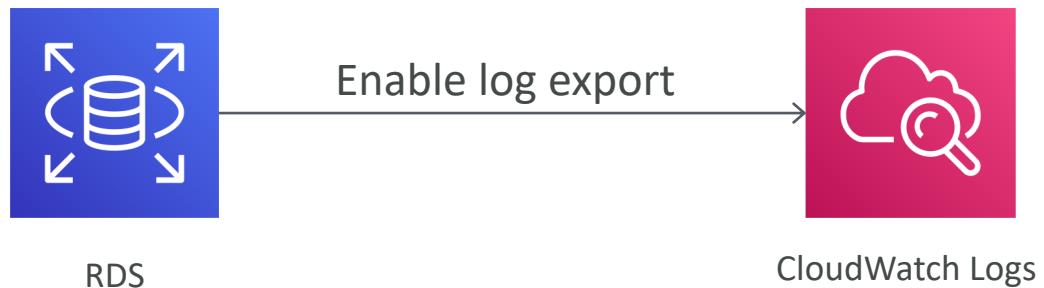
DB clusters with only one DB instance (2)
DB clusters only have one DB instance. Use more than one DB instance for improved performance and availability. [Info](#)

DB clusters

<input type="checkbox"/>	Resource	Recommendation
<input type="checkbox"/>	database-3	Add another DB instance database-3-instance-1-rds with DB instance class db.t3.medium
<input type="checkbox"/>	database-4	Add another DB instance database-4-instance-1-rds with DB instance class db.t2.small to

RDS Logs

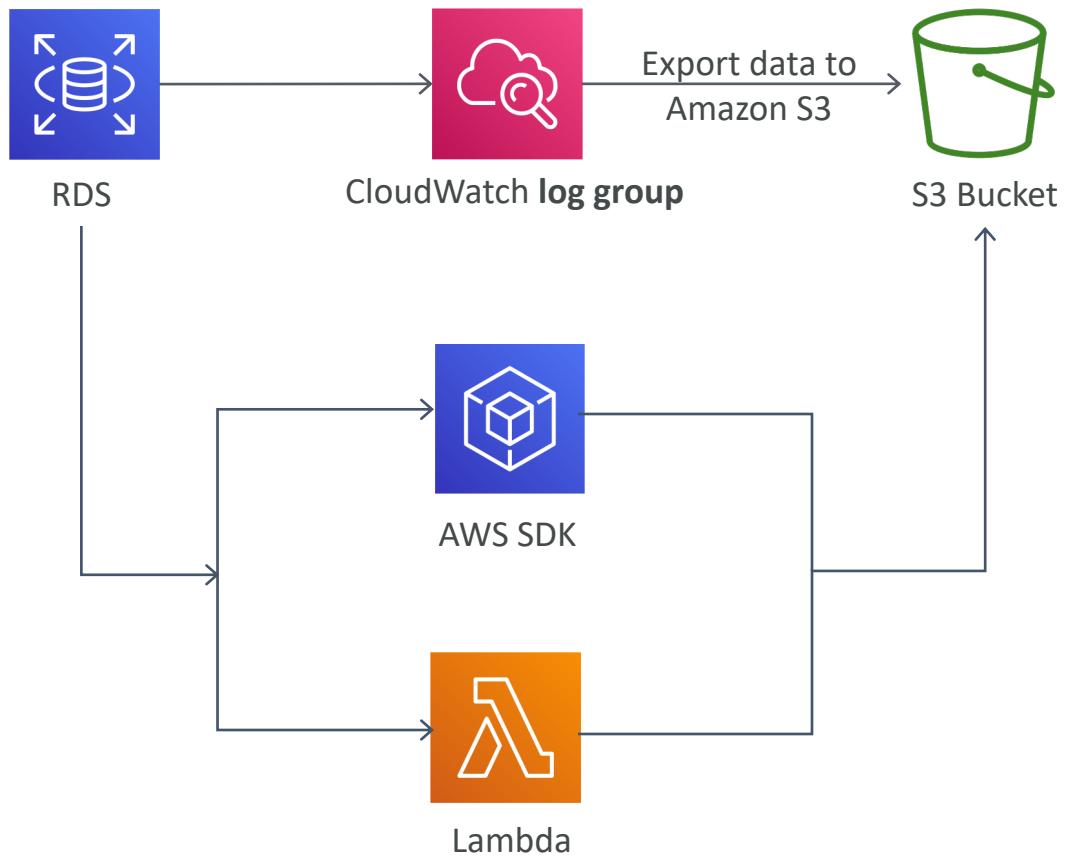
- View / watch / download DB logs from the RDS console
- Can export logs to CloudWatch Logs (log types vary by DB engine)
- **CloudWatch Logs never expire.** To expire them, set log group retention policy (1 day - 10 yrs)
- Logs are accessible from RDS console even if you disable log export to CloudWatch Logs



- Log types that can be exported to CloudWatch Logs
 - Alert log – Oracle
 - Audit log – Oracle, MariaDB, MySQL (must use option group with **MARIADB_AUDIT_PLUGIN** option for MariaDB and MySQL to audit database activity)
 - Listener log – Oracle
 - Trace log – Oracle
 - Error log – SQL Server, MariaDB, MySQL
 - Postgresql log – PostgreSQL (contains audit logs)
 - Upgrade log – PostgreSQL
 - General log – MariaDB, MySQL
 - Slow query log – MariaDB, MySQL

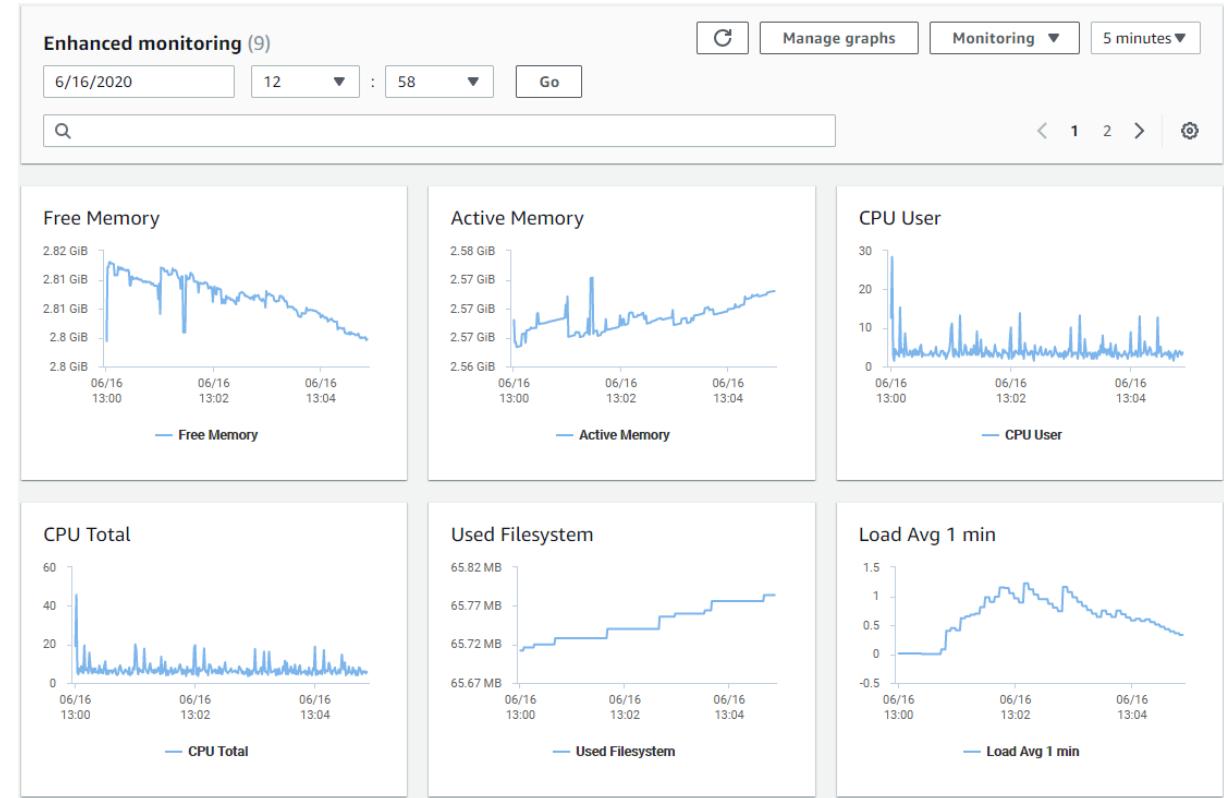
Exporting AWS RDS logs to S3

- RDS database log files can be accessed via RDS console, CLI or API
- Transaction logs cannot be accessed
- You can export log data from CloudWatch Logs to S3 by creating an export task in CloudWatch (create-export-task CLI command)
- Log files can also be downloaded using the RDS API and uploaded to S3 (using Lambda or AWS SDK)

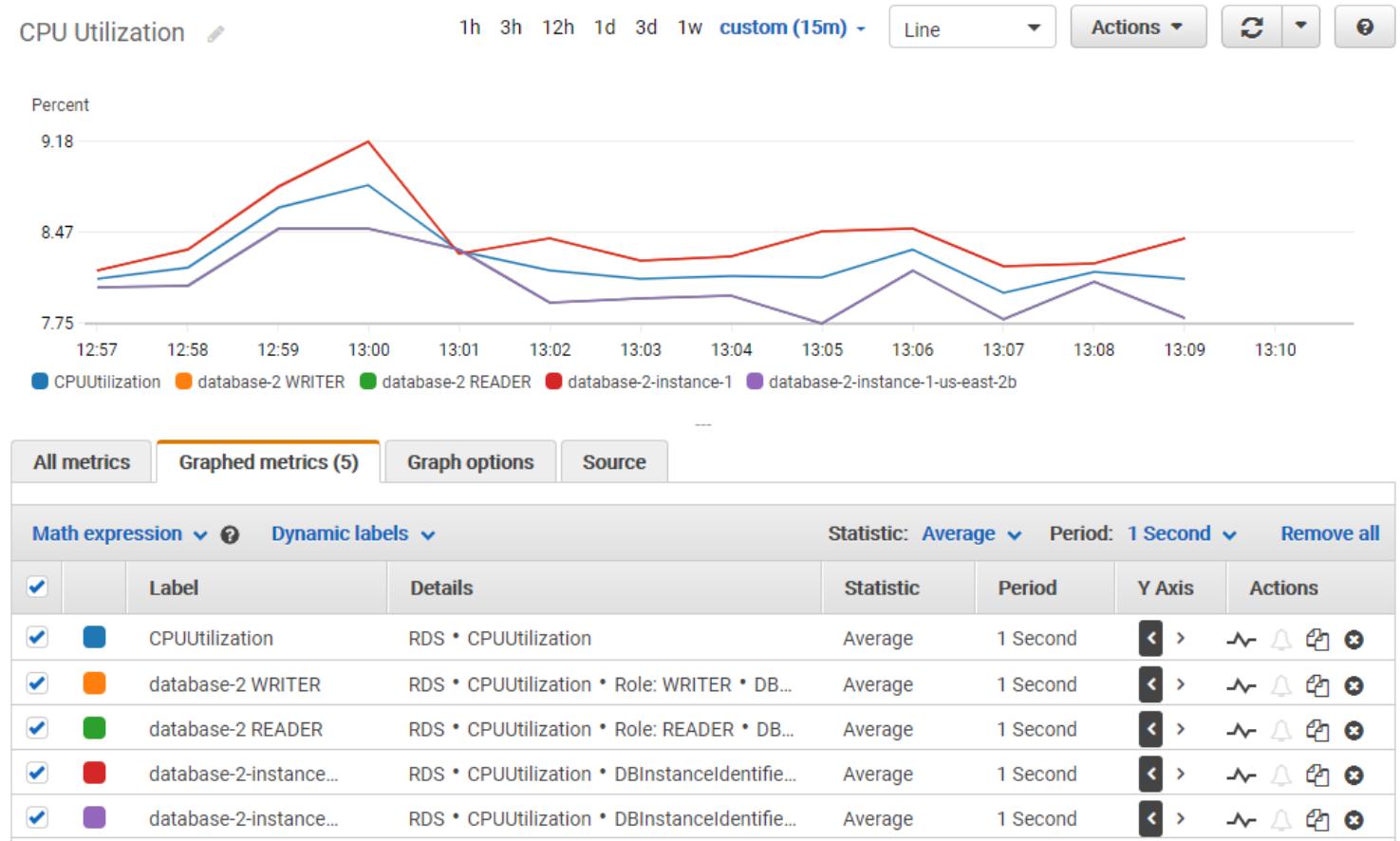


RDS Enhanced Monitoring

- To analyze real-time OS level metrics (CPU / memory usage etc.)
- To monitor different processes or threads that are using the CPU
- Helps identify performance issues
- Increased granularity of 1 to 60 seconds
 - 1, 5, 10, 15, 30, or 60 seconds
- Requires an agent to be installed on the DB server to collect metrics

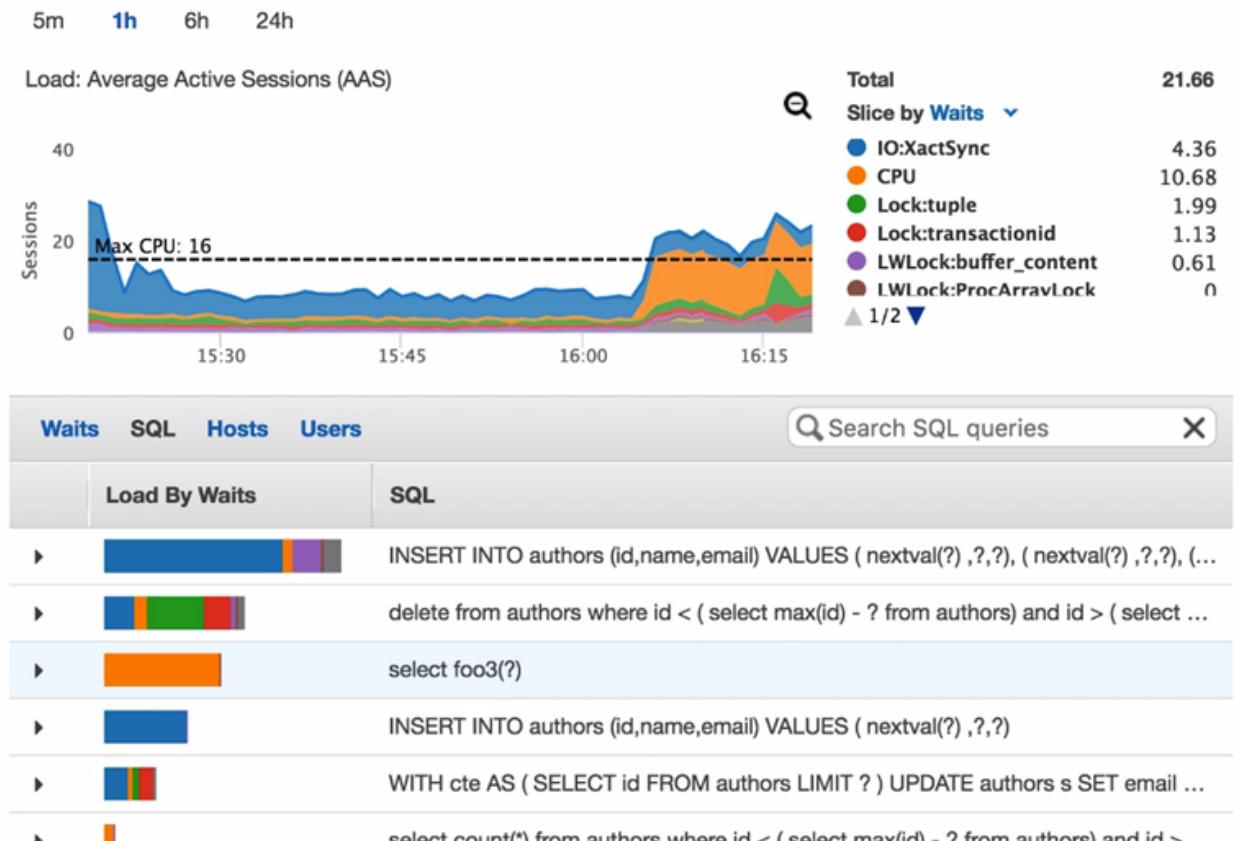


RDS Monitoring in CloudWatch



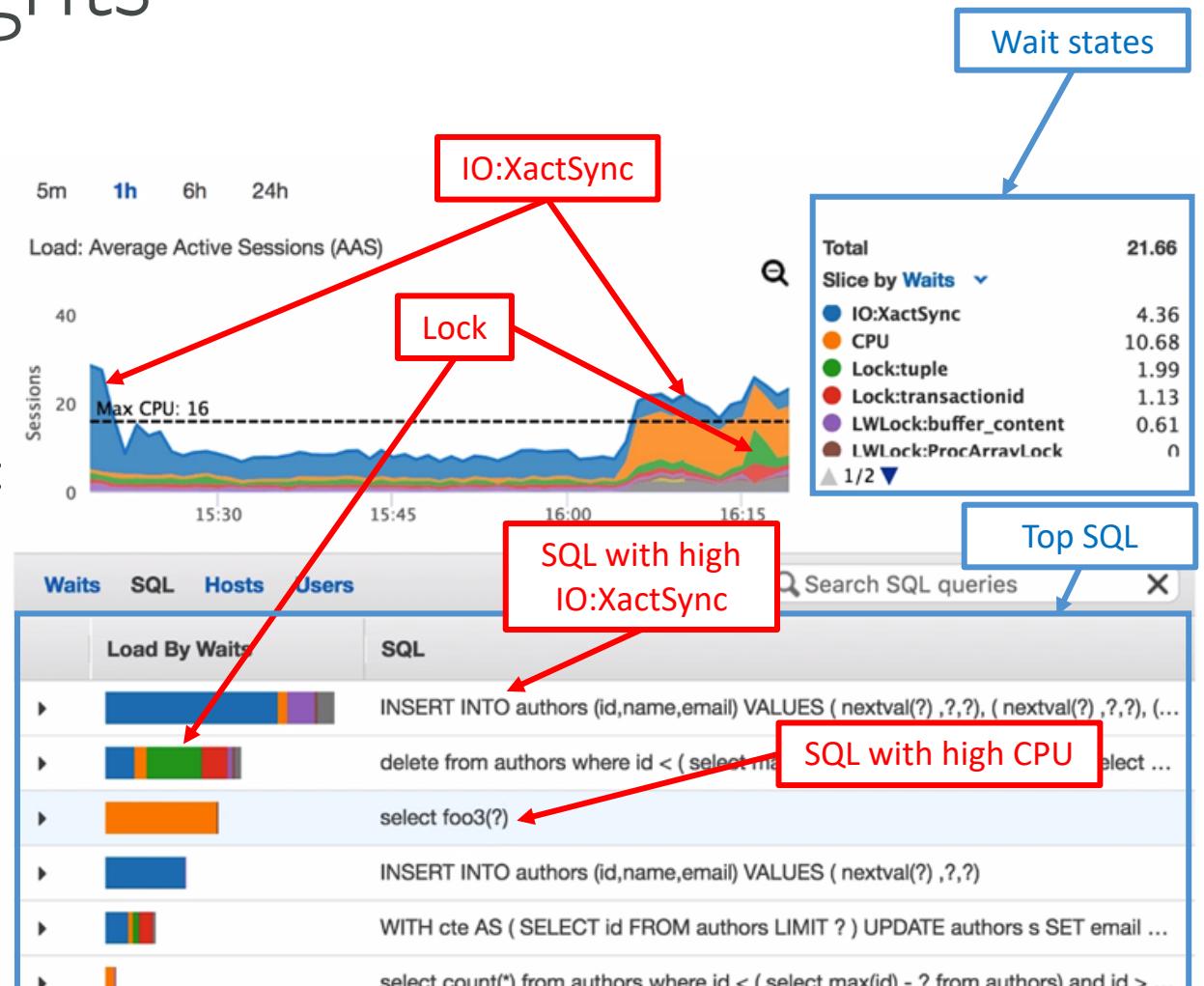
RDS Performance Insights

- Offers **visual dashboard** for performance tuning, analysis and monitoring
- Monitors **DB load** for the instance (if the instance has multiple DBs, you'll see aggregated metrics)
- DB load – average number of active sessions (AAS – average active sessions)
- Performance problems will appear as spikes in the DB load graph
- Helps identify performance bottlenecks, expensive SQL statements, etc.



RDS Performance Insights

- You can visualize the DB load, filter it by waits / SQL / hosts / users
 - Waits - wait state for CPU, IO, Lock etc.
 - SQL – SQL statements
 - Hosts
 - Users
- Identify slow queries (top SQL), locks



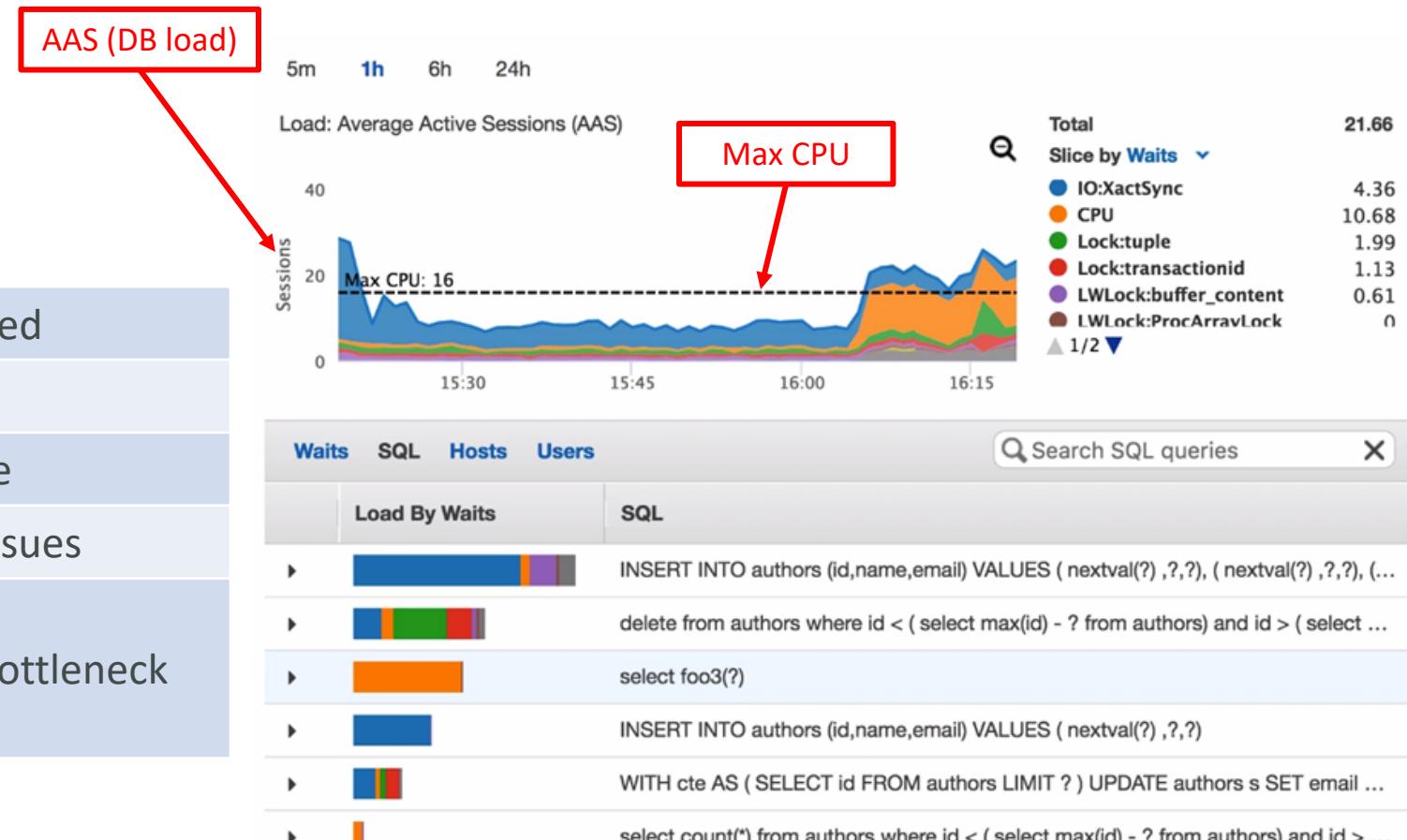
<https://aws.amazon.com/rds/performance-insights/>

RDS Performance Insights

- Use AAS and Max CPU together for analysis

AAS < 1	DB is not blocked
AAS ≈ 0	DB is idle
AAS < Max CPU	CPU is available
AAS > Max CPU	Performance issues
AAS >> Max CPUs (spikes or frequent occurrence)	Performance bottleneck

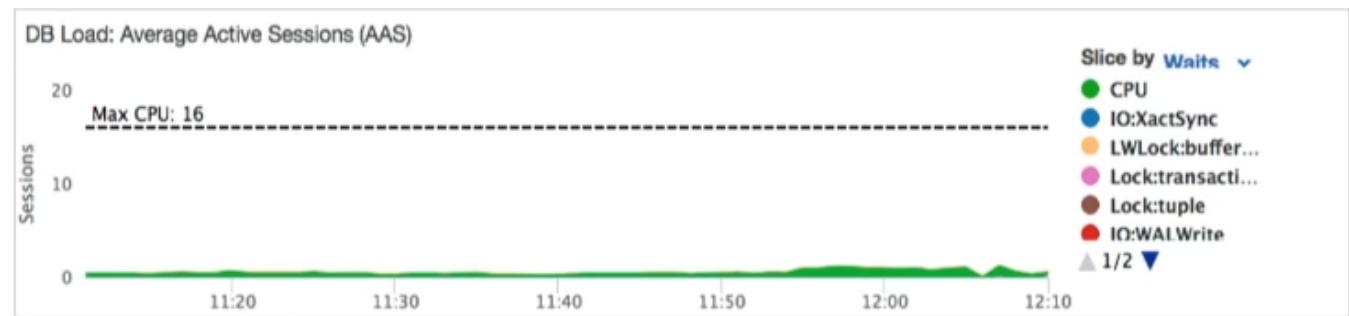
DAT402: <https://youtu.be/RyX9tPxffmw>



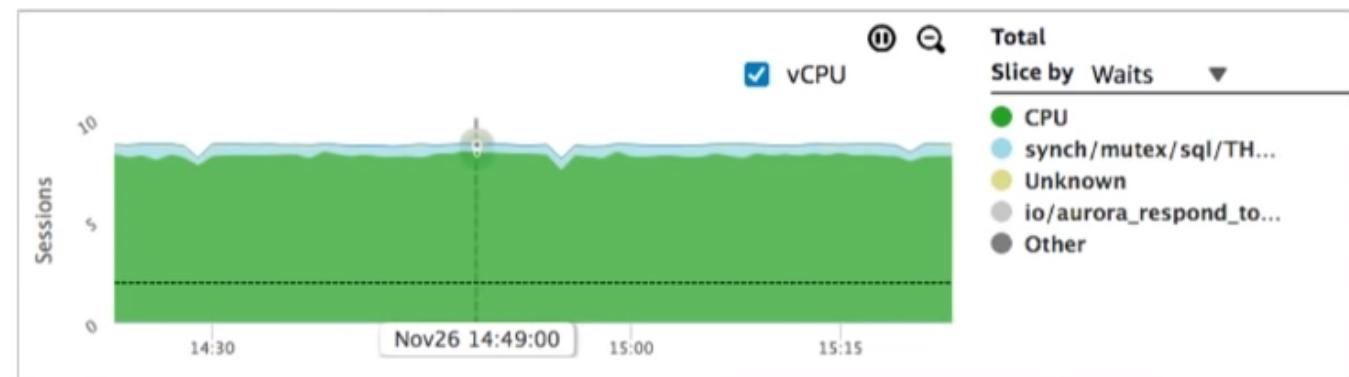
RDS Performance Insights

Can also be used for sizing

- If CPU load is significantly less than Max CPU => Oversized



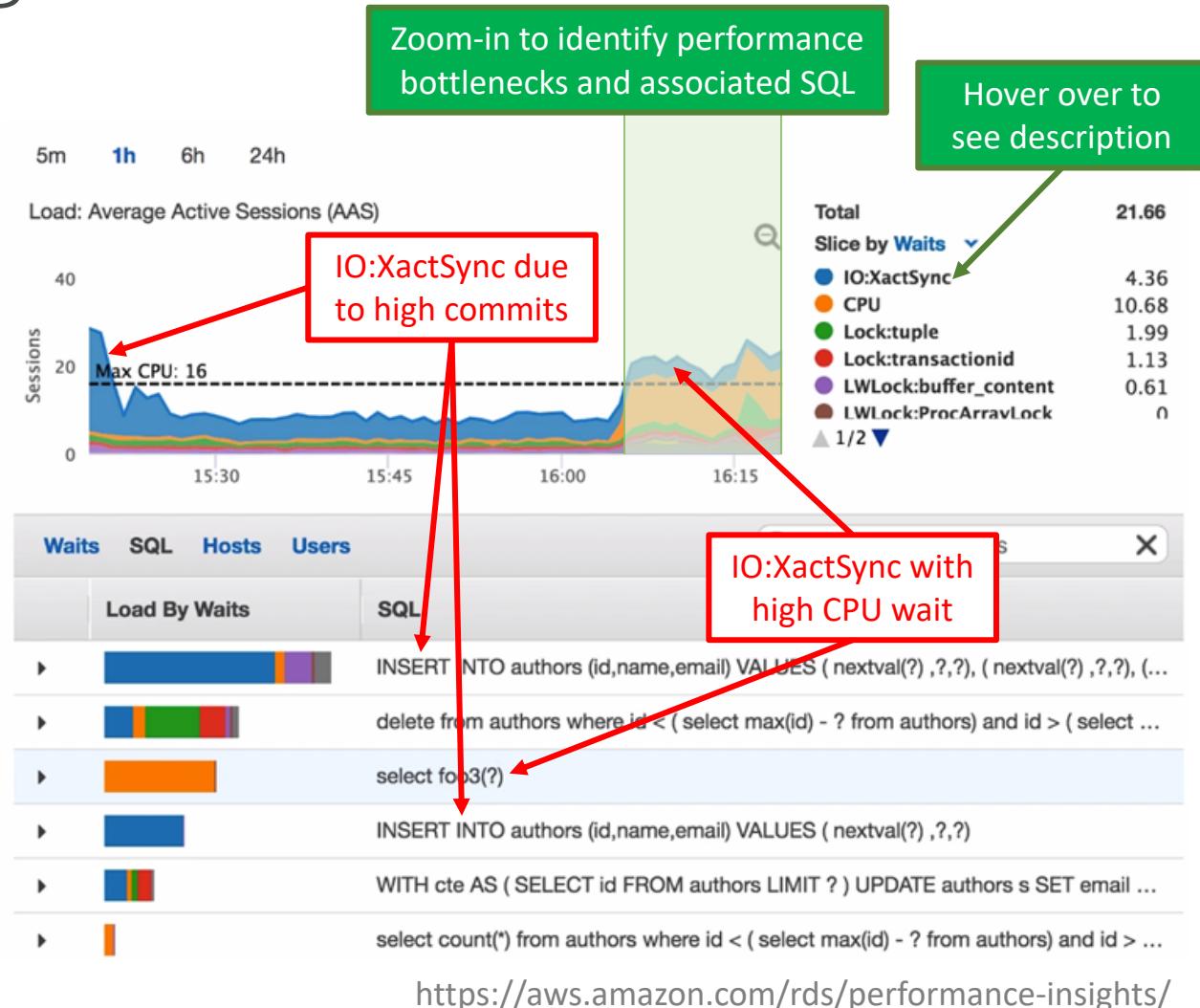
- If CPU load > Max CPU => Undersized



DAT402: <https://youtu.be/RyX9tPxffmw>

RDS Performance Insights

- Hover over wait state legend to see description of any particular state
- **IO:XactSync** – is a wait state in PostgreSQL where a session is issuing COMMIT / ROLLBACK and RDS Aurora is waiting for storage to acknowledge persistence
- Can arise when there is a very high rate of commits on the system
 - modify your application to commit transactions in batches
- If seen along with high CPU waits, it often means DB load exceeds allocated vCPUs
 - Reduce those workloads or scale up to higher CPUs
- Common wait events (can vary by DB engine):
 - <https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Reference.html>



RDS Performance Insights

- Automatically publishes metrics to CloudWatch
- Easily integrates with on-premise or third-party monitoring tools
- Two options for access control
 - Use AmazonRDSFullAccess policy, or
 - Use a custom IAM policy and attach it to the IAM user or role

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": "pi:*",  
      "Resource": "arn:aws:pi:*:metrics/rds/*"  
    }  
  ]  
}
```

CloudWatch Application Insights

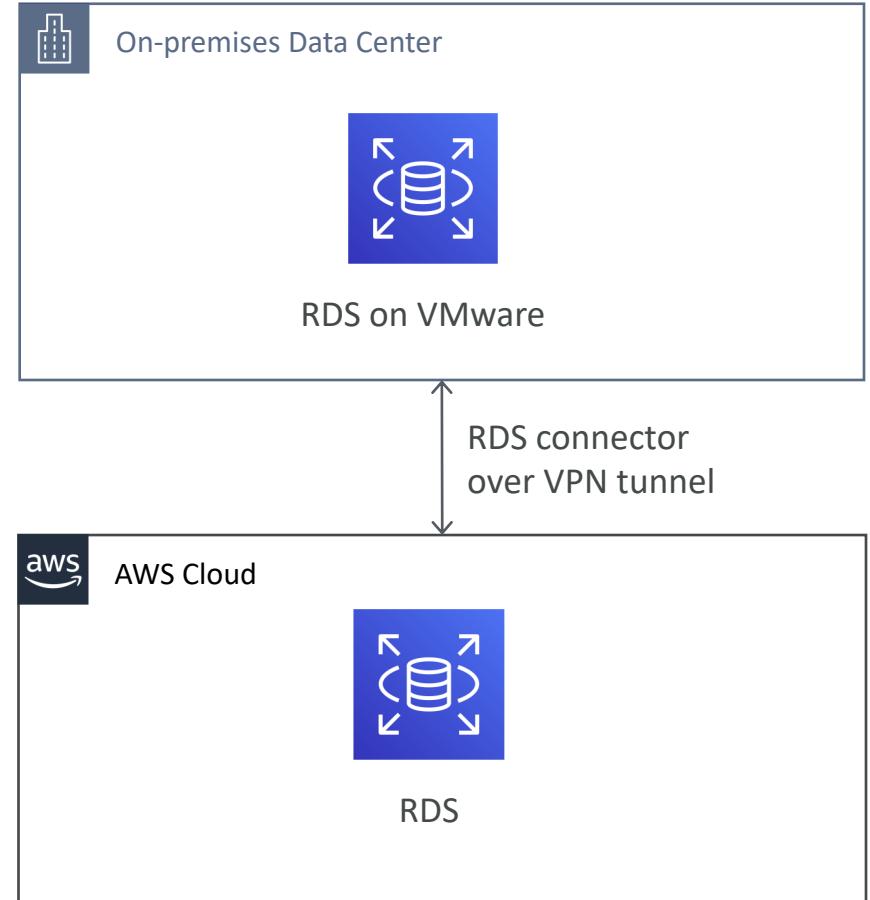
- For .NET and SQL Server
- Also supports DynamoDB tables
- Identifies and sets up key metrics, logs, and alarms for SQL Server workloads
- Uses CloudWatch events and alarms
- Useful for problem detection, notification and troubleshooting



CloudWatch

RDS on VMware

- Lets you deploy RDS DBs in on-premises VMware environments (VMware vSphere)
- Same user interface as in AWS
- Supports MySQL, PostgreSQL, and SQL Server
- Fully managed DBs
- Uses health monitoring to detect unhealthy database instances and automatically recovers them
- Support manual and automatic backups with PITR
- Can use CloudWatch for monitoring



RDS Multi-AZ, Replicas and Failovers (MySQL)



Demo

RDS – Good things to know

- Read replica can be made writable (for MySQL/MariaDB)
- For other engines
 - read replica cannot be made writable, but you can promote it to make it writable
- For Oracle and SQL Server
 - Automatic backups or manual snapshots are not supported on the replica
- For Oracle
 - Does not yet support Oracle RAC (a cluster DB with a shared cache architecture)
- For SQL Server
 - Supports both Multi-AZ options – Database Mirroring and Always On
- For PostgreSQL
 - Only manual snapshots are supported on the replica (no automatic backups)
 - Set log retention with parameter rds.log_retention_period
- For MySQL/MariaDB
 - Set log retention with stored procedures
 - e.g. call mysql.rds_set_configuration('binlog retention hours', 24);



RDS

Amazon Aurora



Amazon Aurora

- MySQL and PostgreSQL-compatible relational database in the cloud (that means your drivers will work as if Aurora was a Postgres or MySQL DB)
- 5x faster than standard MySQL databases
- 3x faster than standard PostgreSQL databases
- 1/10th the cost of commercial-grade RDBMS
- Up to 15 read replicas (Multi AZ, Auto Scaling Read Replicas)
- “Aurora Serverless”(Automatic start/stop, Autoscaling, Self-healing storage)
- Aurora Global DB – Supports multi-region read replication (= local reads w/ under one second latency)
- Auto scaling of storage from 10 GB to 64 TB (soft limit)
- Same security / monitoring / maintenance features as RDS

Amazon Aurora

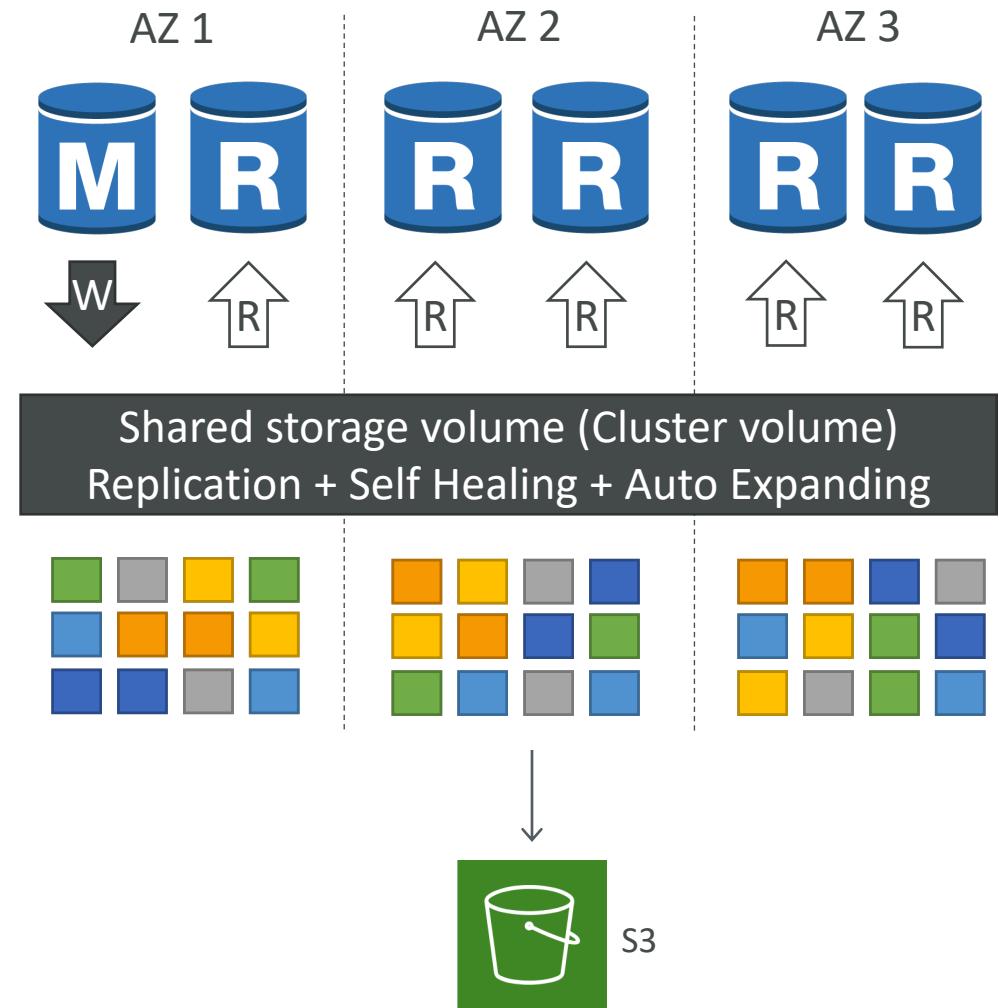


Amazon Aurora

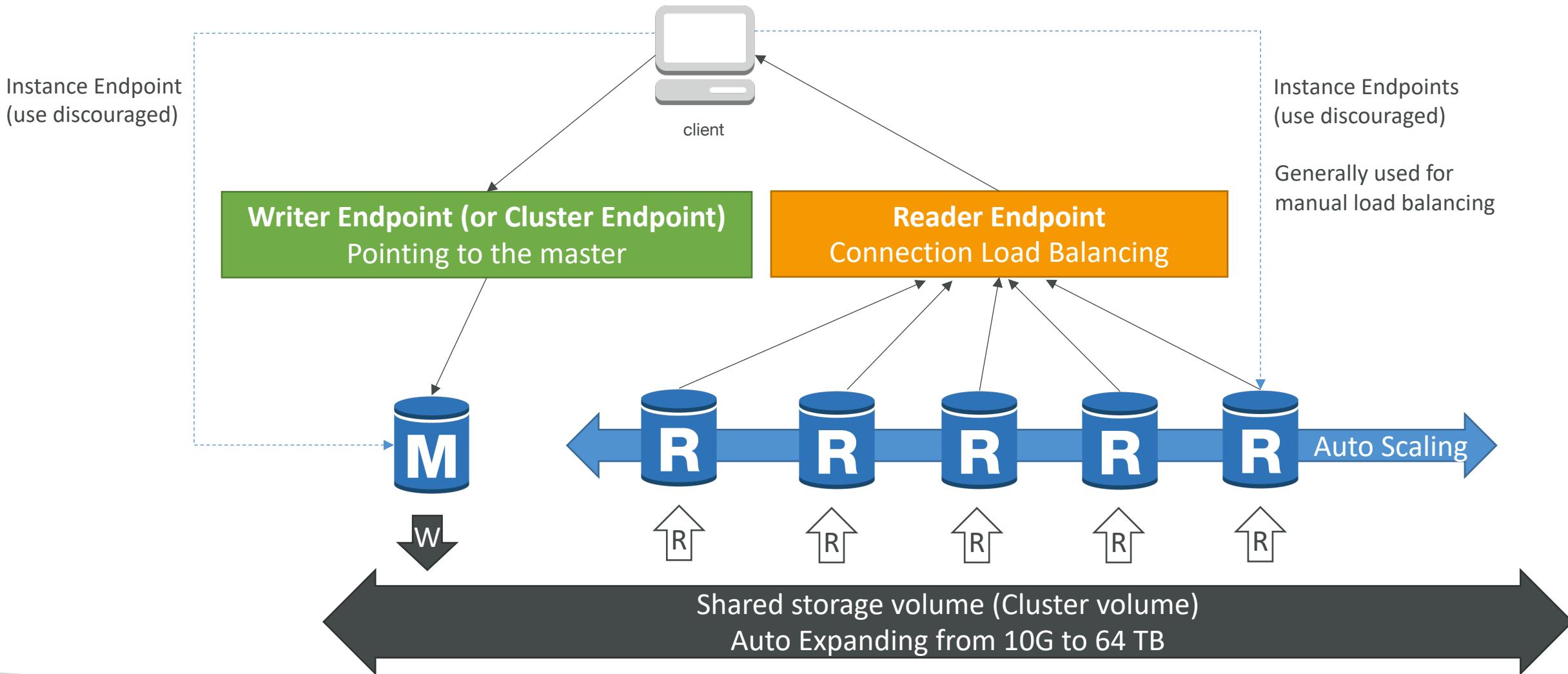
- Only available on RDS (can't be hosted on EC2 or elsewhere)
- Maintains 6 copies across 3 AZs
- Backups are stored on S3
- Fast backtracking option for PITR
- Automatic and fast failovers
- If desired, table indexes can exist on only the replicas
- Plug-and-play like migration for MySQL and PostgreSQL databases (no changes needed to your application)
- Aurora costs more than RDS (20% more) – but is more efficient
- **Use case:** same as RDS, but with less maintenance / more flexibility / higher performance

Aurora Architecture (High Performance)

- One Aurora Instance takes writes (master)
- Compute nodes on replicas do not need to write/replicate (=improved read performance)
- 6 copies of your data across 3 AZ (distributed design)
 - Lock-free optimistic algorithm (quorum model)
 - 4 copies out of 6 needed for writes (4/6 write quorum - data considered durable when at least 4/6 copies acknowledge the write)
 - 3 copies out of 6 needed for reads (3/6 read quorum)
 - Self healing with peer-to-peer replication, Storage is striped across 100s of volumes
- Log-structured distributed storage layer: passes incremental log records from compute to storage layer (=faster)
- Master + up to 15 Aurora Read Replicas serve reads
- Data is continuously backed up to S3 in real time, using storage nodes (compute node performance is unaffected)

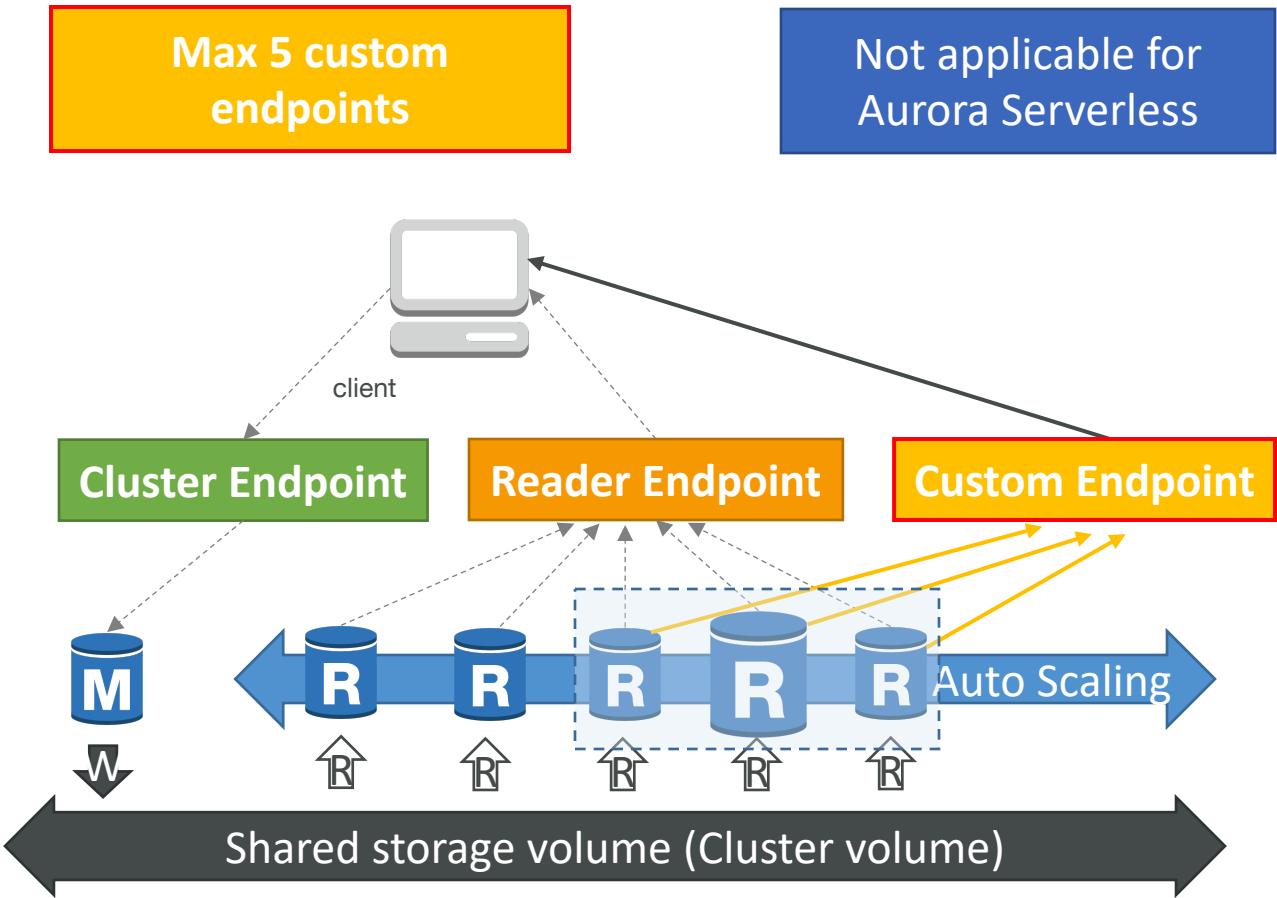


Aurora DB Cluster



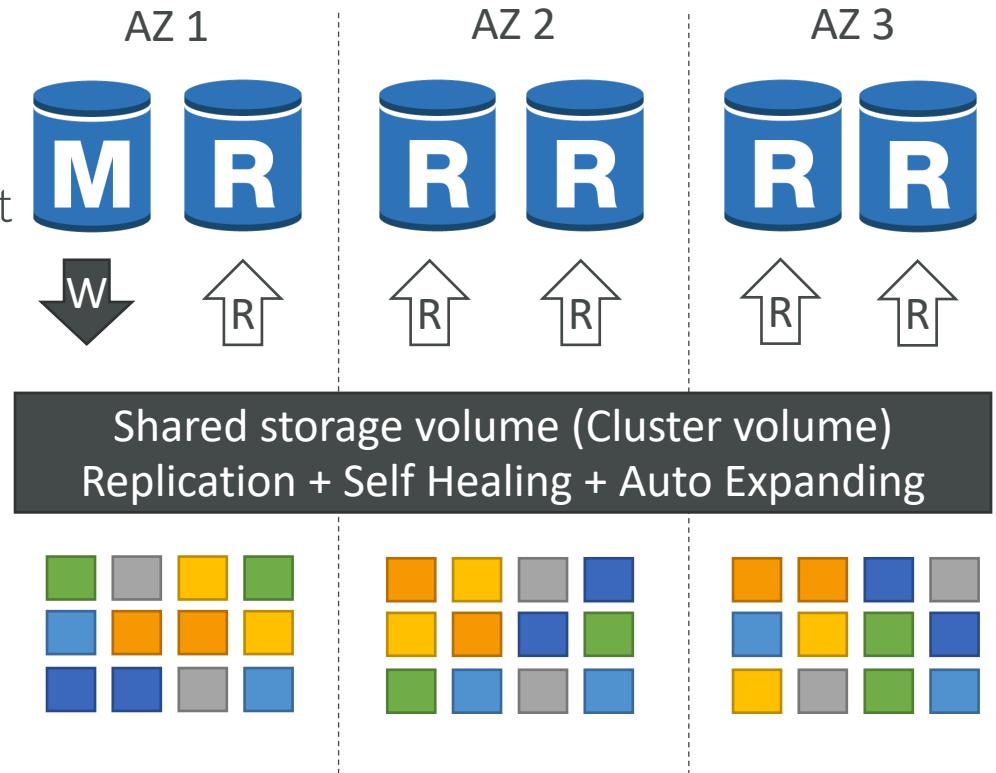
Custom Endpoints

- Use custom endpoints to simplify connection management
- When your cluster contains DB instances with different sizes and configs
- Use case – custom load balancing with HA
 - Direct internal users to low-capacity instances (for reporting / analytics)
 - Direct production traffic to high-capacity instances.
 - No need to maintain CNAME aliases for custom load balancing when you use custom endpoints



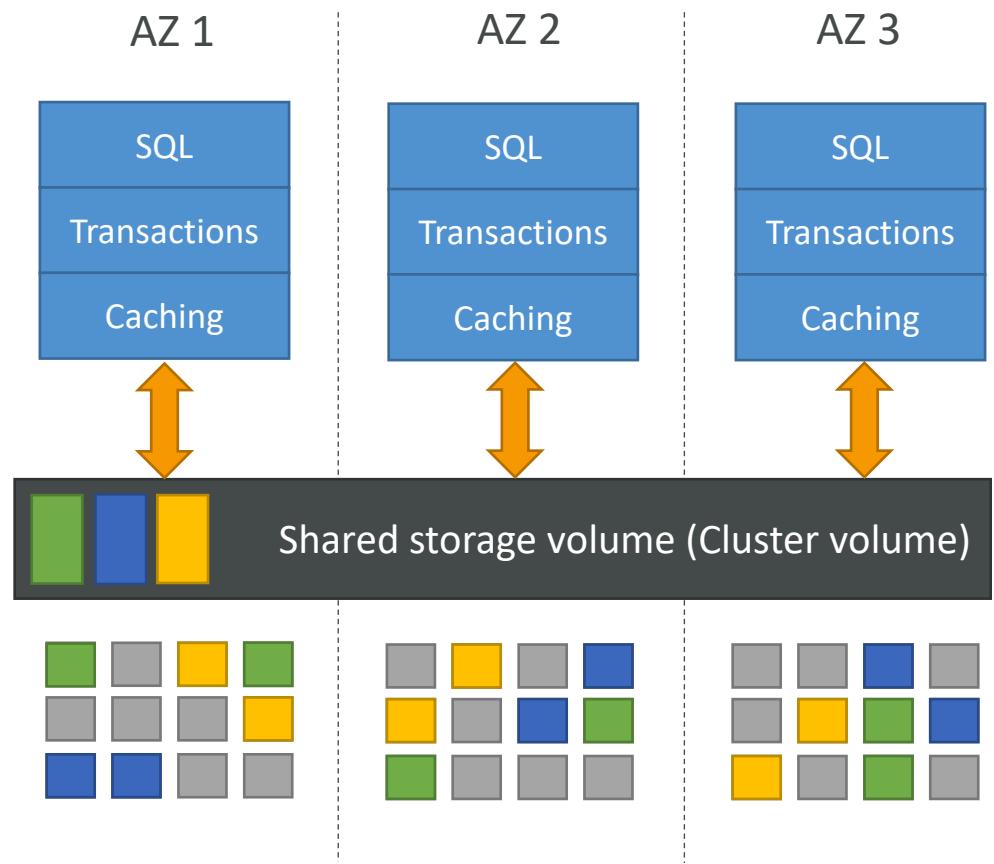
Aurora Architecture (High Availability)

- Replicas double up as failover targets (standby instance is not needed)
- Automatic failovers from the master instance
- Failover to a running instance (replica) typically takes about 30 seconds
- Failover to a new instance happens on a best-effort basis and can take longer
- Supports ZDP (Zero-Downtime Patching) – preserves client connections through an engine patch, on a best-effort basis
- Aurora offers integrated caching layer with built in write-through capabilities (can use external caches like ElastiCache on top of this)



Aurora Parallel Query

- Allows for faster analytical queries (data-intensive queries)
- Can run **analytical queries in parallel** across thousands of **storage nodes**
- No need to copy data into a separate system
- Parallelized query processing in the Aurora storage layer
- Only for Aurora MySQL engine
- PostgreSQL engine has an unrelated feature, also called "parallel query"
- For Aurora clusters enabled for Parallel query
 - Performance Insights is not supported
 - Backtrack (PITR) is not supported
 - IAM Authentication is not supported



Aurora Serverless

- Fully-managed on-demand, auto-scaling Aurora configuration
- Supports both MySQL and PostgreSQL
- Automatically starts up, shuts down, scales up/down based on application needs
- Automatically shuts down when not in use
 - Supports automatic pause
 - no compute charge when its not running
 - to “wake up”, it could take ~30 seconds
- Aurora serverless typically results in 40% lesser overall costs as compared to RDS
- Great for **infrequent, intermittent, or unpredictable workloads**
 - No capacity planning needed
 - e.g. DEV/TEST envs which are typically used only during business hours (9 to 5)



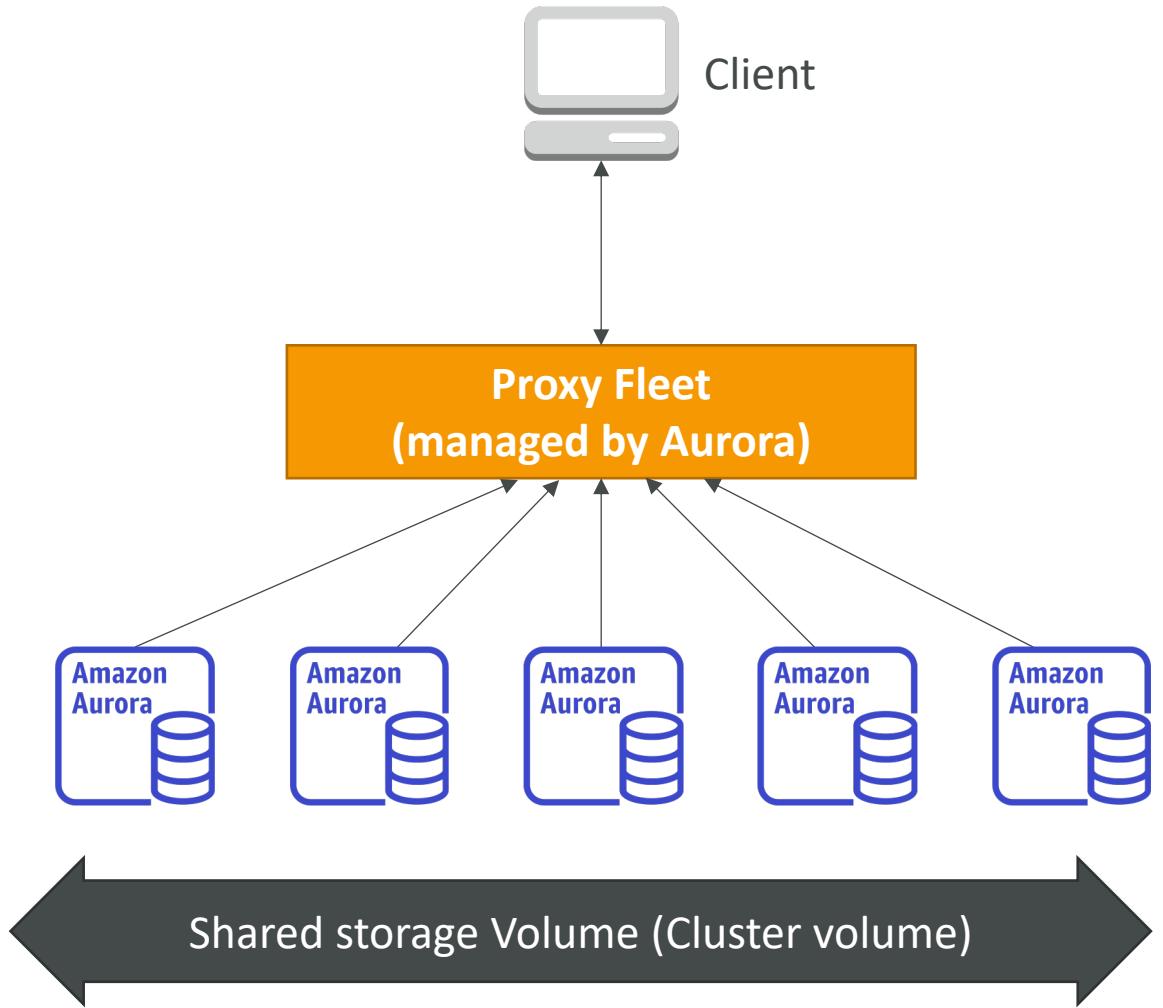
Auto Shutdown

Auto Scaling

DEV / TEST Env

Aurora Serverless

- Compute layer is placed in a single AZ (not multi-AZ)
- Storage volume for the cluster is spread across multiple AZs
- DB endpoint connects to a proxy fleet that routes the workload to a warm pool of DB resources that can be quickly scaled
- Eliminates the need to implement read replicas and HA (multi-AZ)
- In case of cluster or AZ-failure, Aurora creates the DB instance in another AZ (automatic multi-AZ failover)
- Failover time is longer than a provisioned cluster
- Data remains available during DB instance or AZ outage



Data API for Aurora Serverless

- Run SQL queries over API (versus a DB connection)
- Run queries using
 - Query Editor within the RDS console
 - Command line
 - AWS SDK
- No need of connection management
- Uses DB credentials stored in AWS Secrets Manager
- Perfect for using Aurora with Lambda functions
- No need to configure your Lambda function to access VPC resources

The screenshot shows the AWS RDS Query Editor interface. At the top, it displays the URL "RDS > Editor: aurora-sls-db". Below that is a navigation bar with tabs: "Editor" (which is selected), "Recent", and "Saved queries".
The main area contains a code editor with the following SQL query:

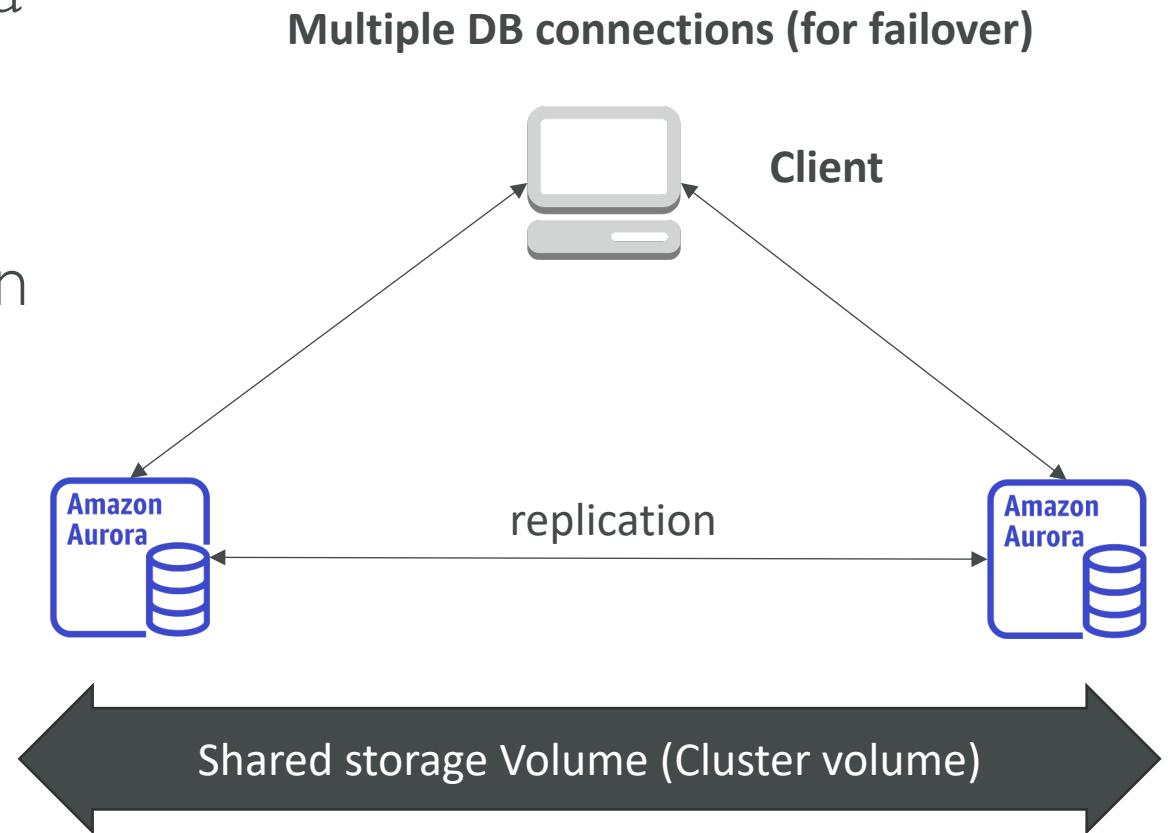
```
1 SELECT * FROM `students`
```

Below the code editor are three buttons: "Run" (orange), "Save", and "Clear". To the right of these buttons is a "Change database" link.
The bottom section is titled "Output" and shows the results of the query. It says "Result set 1 (3)" and "Rows returned (3)". There is a "Search rows" input field and a navigation bar with arrows and a page number "1".
The results table has columns: "student_id", "student_name", and "contact_number". The data is as follows:

student_id	student_name	contact_number
1	Amanda Smith	8432399899
2	Justin Kingsley	9895895154
3	Dennis Green	8882568751

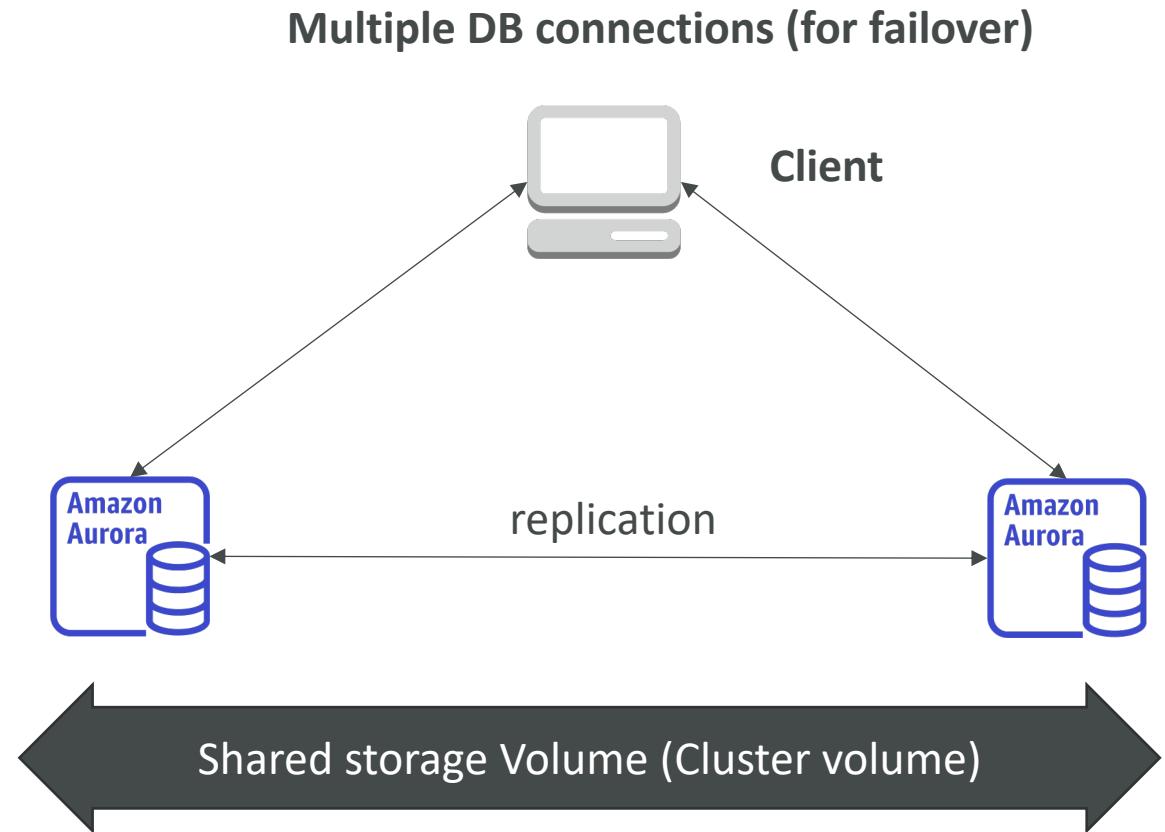
Aurora Multi-Master (Multiple Writers)

- Every node can R/W (vs promoting a read replica as the new master)
- There is no failover – another writer takes over in case primary goes down (termed as continuous availability as against high availability)
- Typically results in zero downtime
- Max two DB instances in a multi-master cluster (as of now)



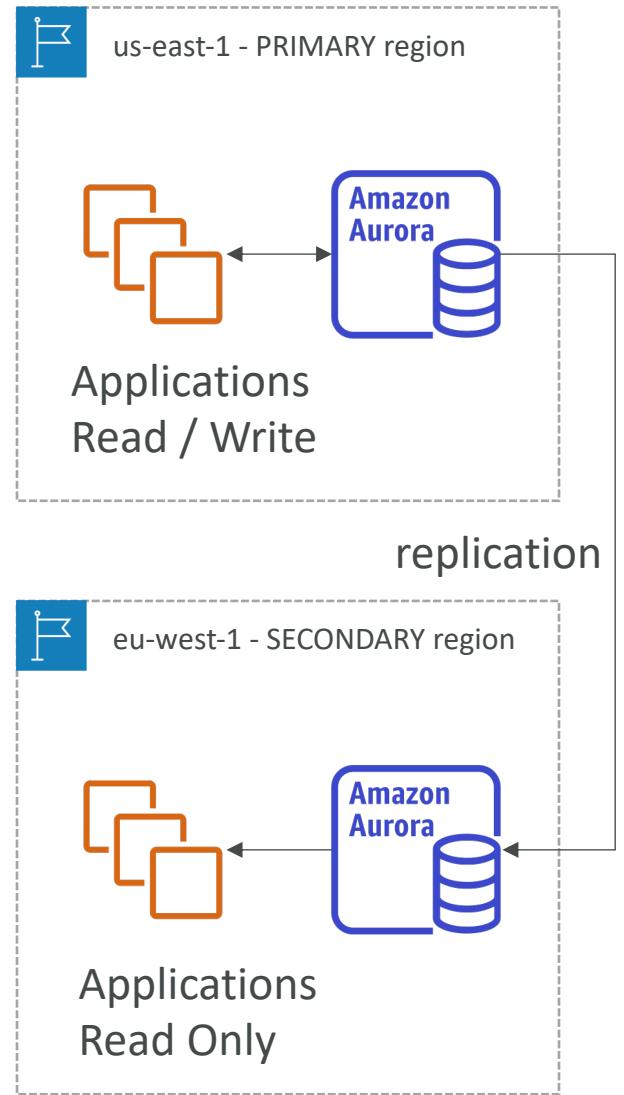
DDL Operations on a Multi-Master Cluster

- DDL = Data Definition Language
 - Define DB schema
 - CREATE / DROP / ALTER etc.
- DDL operations on a table prevent concurrent writes to that table
- Try to avoid issuing large numbers of short DDL statements in your application



Global Aurora

- Aurora Cross Region Read Replicas:
 - Useful for disaster recovery
 - Simple to put in place
 - Replica promotion can take a few minutes depending on workload
- Aurora Global Databases (recommended):
 - 1 Primary Region (read / write)
 - Up to 5 secondary (read-only) regions, replication lag is less than 1 second (i.e. local reads with < 1 second latency)
 - Up to 16 Read Replicas per secondary region
 - Helps for decreasing latency
 - Promoting another region (for disaster recovery) has an RTO of < 1 minute, RPO = 1 second



Reliability features in Aurora

Designed to be reliable, durable, and fault tolerant

Storage Auto-Repair

- automatically detects and repairs disk volume failures in the cluster volume
- quorum model ensures that there is no data loss due to disk failures

Survivable Cache Warming

- Aurora page cache is managed in a separate process from the DB
- Page cache stores pages for known common queries
- Every time Aurora starts / restarts, it **preloads the buffer pool cache** from the page cache
- Eliminates the need to warm up the buffer cache => faster failovers / restores



Crash Recovery

- Designed to recover from crash almost instantaneously
- **Does NOT need to replay the redo log from DB checkpoint**
- **Does NOT need binary logs** for replication within cluster or for PITR (only used for external replication)
- Binary logging on Aurora directly affects the recovery time after a crash
- Higher the binlog data size, longer it takes for crash recovery
- Disable binary logging (`binlog_format = OFF`) to reduce recovery time

Aurora Pricing Model

- 1/10th the cost of competing commercial-grade RDBMS solutions
- Costs more than RDS (20% more) – but is more efficient
- Pricing model like RDS (pay as you go)
- When creating an Aurora database, you choose:
 - Instance type (on-demand / reserved (for discounts) / serverless)
 - Engine type (PostgreSQL / MySQL)
 - DB instance class (Memory-optimized / Burstable performance)
 - Regional or Global DB (CRR latency < 1 second)
- Storage (GB/month) / Backups / Backtrack / Snapshot Export to S3
- I/O (per million requests)
- Data transfer



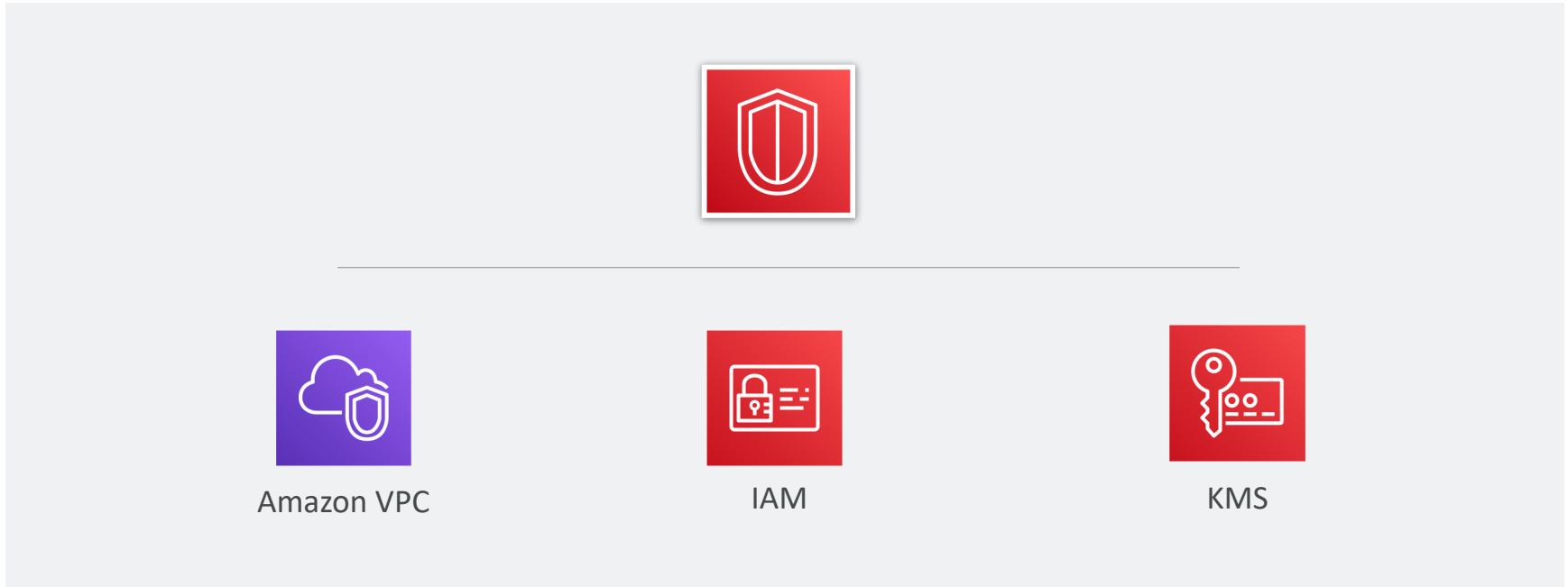
Aurora Serverless Pricing

- Pay per second pricing model
- No charge when database instance is not running
- You pay for database capacity + database storage + I/O
- Database capacity is measured in Aurora Capacity Units (ACUs)
- 1 ACU \approx 2 GB of memory with corresponding CPU and networking
- You choose a range (min and max) of ACUs for autoscaling
- You pay for actual ACUs consumed (on per second basis)



Aurora Security – Network, IAM & Encryption

- Aurora uses the native RDS infrastructure or provisions for network, IAM and encryption



SSL for Aurora Serverless

- Same procedure as connecting to RDS / Aurora provisioned
- With Aurora Serverless, you can use certificates from ACM
- No need to download RDS SSL/TLS certificates
- To enforce SSL:
 - PostgreSQL: set parameter `rds.force_ssl=1` in the DB cluster parameter group (is a dynamic parameter, unlike PostgreSQL on RDS)
 - `rds.force_ssl` is a dynamic parameter in both Aurora PostgreSQL provisioned and Aurora PostgreSQL Serverless
 - MySQL: `ALTER USER 'mysqluser'@'%' REQUIRE SSL;`
- Connection examples:
 - PostgreSQL: `sslrootcert=rds-cert.pem sslmode=[verify-ca | verify-full]`
 - MySQL: `--ssl-ca=rds-cert.pem --ssl-mode=VERIFY_IDENTITY` (MySQL 5.7+)
 - MySQL (older versions): `--ssl-ca=rds-cert.pem --ssl-verify-server-cert`



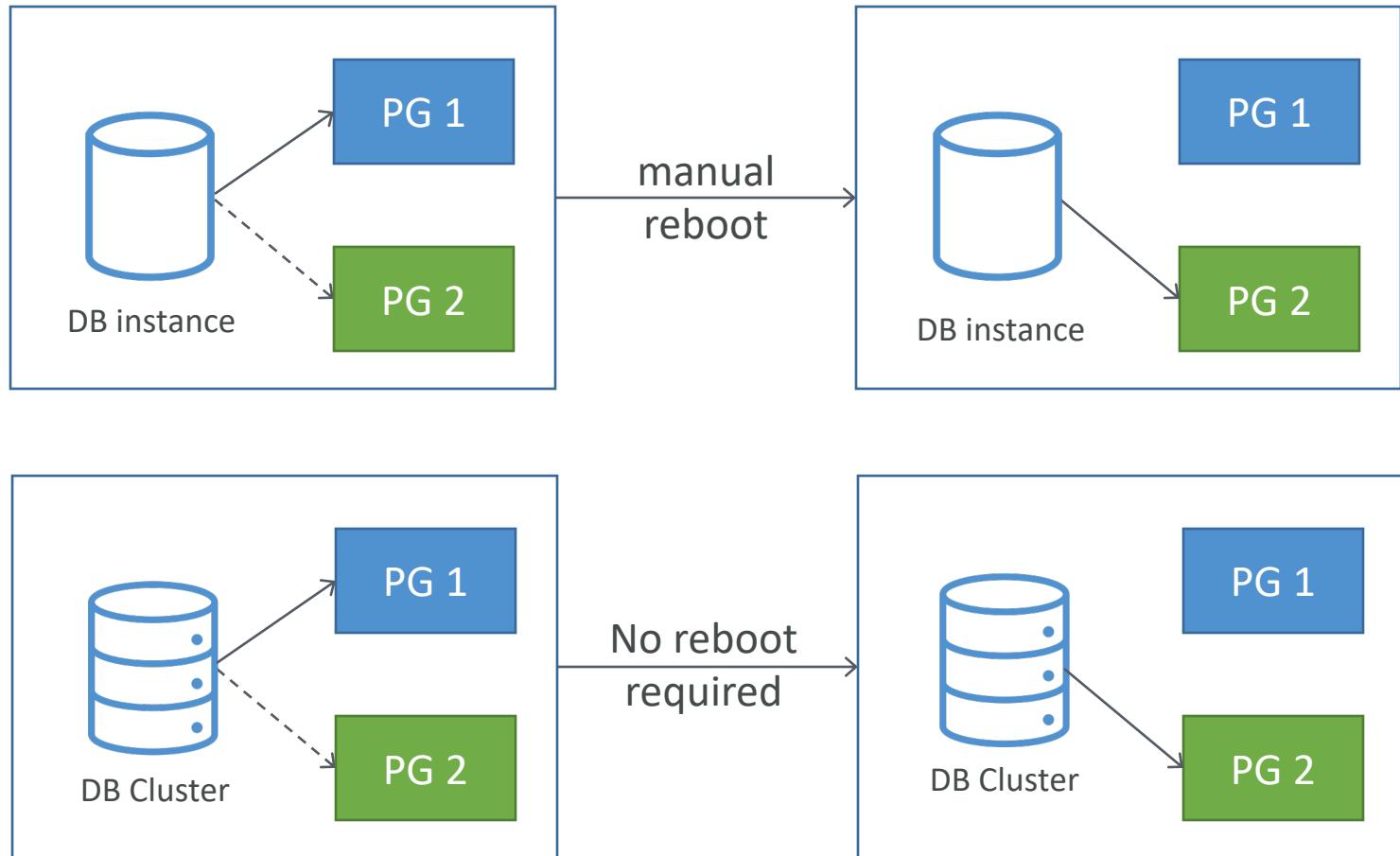
Parameter groups in Aurora

- Work in same manner as other RDS engines
- In addition, also has cluster parameter groups
- DB parameter group = engine config for the given DB instance
- Cluster parameter group = engine config for all DB instances within an Aurora DB cluster



Changing the parameter group

- Changing the parameter group associated with a DB instance requires a manual reboot
- Changing the parameter group associated with a DB cluster doesn't require a reboot



Parameter group precedence

- By default, cluster parameter group values take precedence over those of the DB instance parameter group
- Any DB parameter settings that you modify take precedence over the DB cluster parameter group values (even if you change them to their default values)
- To make the DB cluster parameter group values take precedence again, you must reset the DB instance parameter group
 - Use reset-db-parameter-group command / ResetDBParameterGroup API
- To identify the overridden parameters
 - Use describe-db-parameters command / DescribeDBParameters API



Parameter groups in Aurora Serverless

- Only DB cluster parameter groups (no DB parameter groups)
- This is because there are no permanent DB instances
- Aurora manages the capacity configuration options
- You can define your own DB cluster parameter group to define other options
- All parameter changes are applied immediately (irrespective of *Apply Immediately* setting)

Creating an Aurora DB



Demo

Creating Aurora Serverless DB



Demo



DataCumulus | RIZMAXed

Using Data API with Aurora Serverless DB



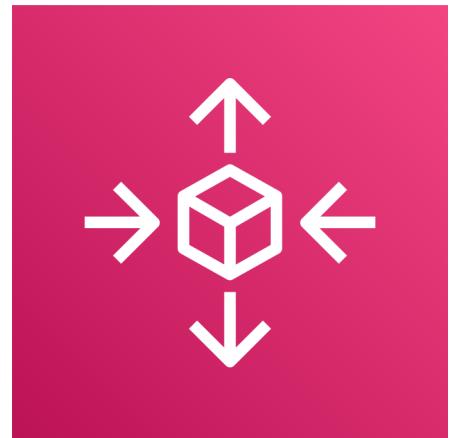
Demo



DataCumulus | RIZMAXed

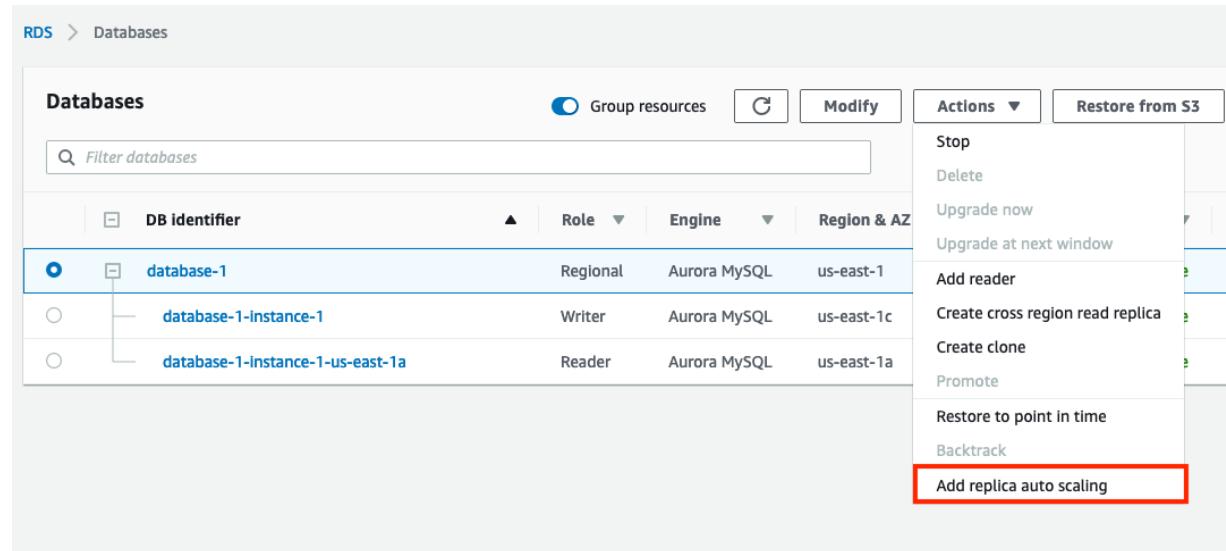
Scaling in Aurora

- Storage scaling
 - Built in and automatic
 - 10 GB increments up to 64 TB (soft limit)
- Compute scaling
 - Instance scaling
 - Vertical scaling
 - Minimal downtime possible using replica promotion (force failover)
 - Read scaling
 - Horizontal scaling
 - Up to 15 read replicas
 - Can also set higher value for max_connections parameter in the instance level PG
- Auto Scaling



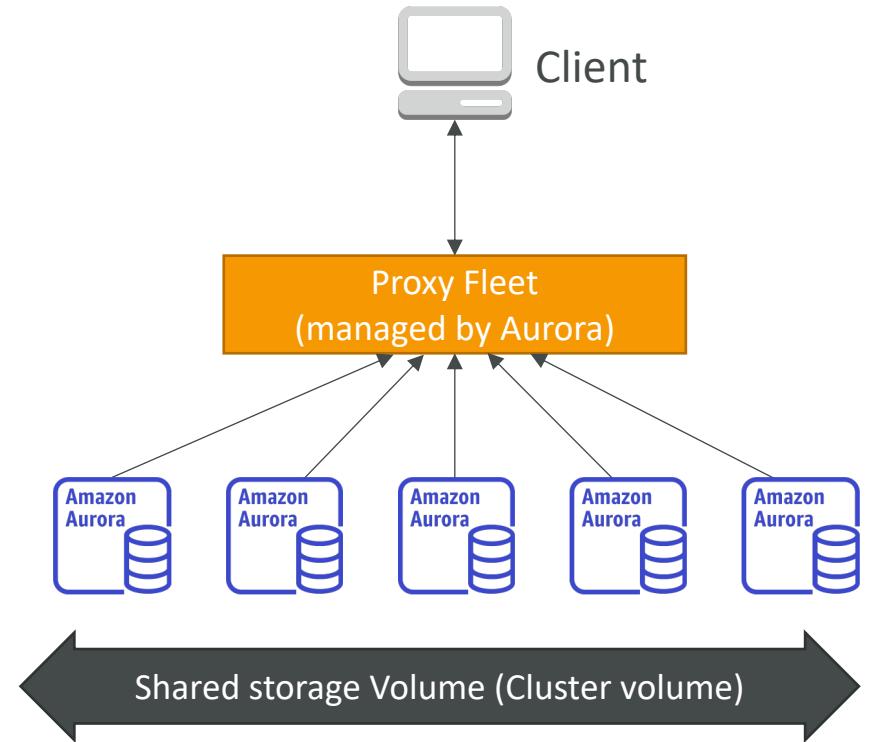
Replica auto scaling in Aurora

- You define scaling policies
- Horizontal scaling (**read scaling**) achieved by defining min and max replicas and scaling conditions
- Condition can be defined using a target metric – CPU utilization or number of connections
- Makes use of the CloudWatch metrics and alarms
- You also define a service-linked IAM role and cooldown period



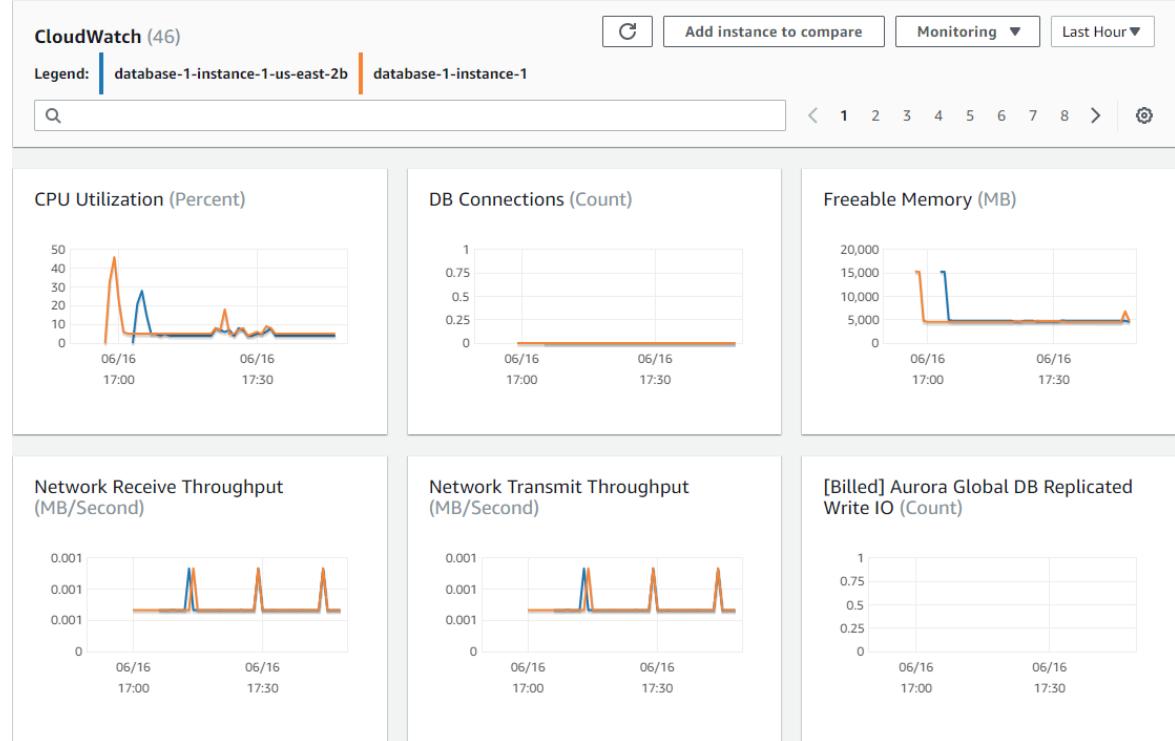
Autoscaling in Aurora Serverless

- Scales up and down based on the load
- Load = CPU utilization and number of connections
- After scale up
 - 15-minute cooldown period for subsequent scale down
- After scale-down
 - 310-seconds cooldown period for subsequent scale down
- No cooldown period for scaling up (scale up can happen anytime as and when necessary)
- Scaling cannot happen in case
 - Long running queries/transactions are in progress
 - Temporary tables or table locks are in use



Aurora Monitoring

- Same as in RDS (uses the RDS monitoring infrastructure)
 - RDS Notifications
 - Log Exports (can be exported to CloudWatch Logs, S3)
 - CloudWatch (Metrics / Alarms / Logs / service health status)
 - Enhanced Monitoring
 - Performance Insights
 - **RDS Recommendations** (periodic automated suggestions for DB instances, read replicas, and DB parameter groups)
 - CloudTrail for audits
 - AWS Trusted Advisor



Monitoring in RDS and Aurora



Demo



DataCumulus | RIZMAXed

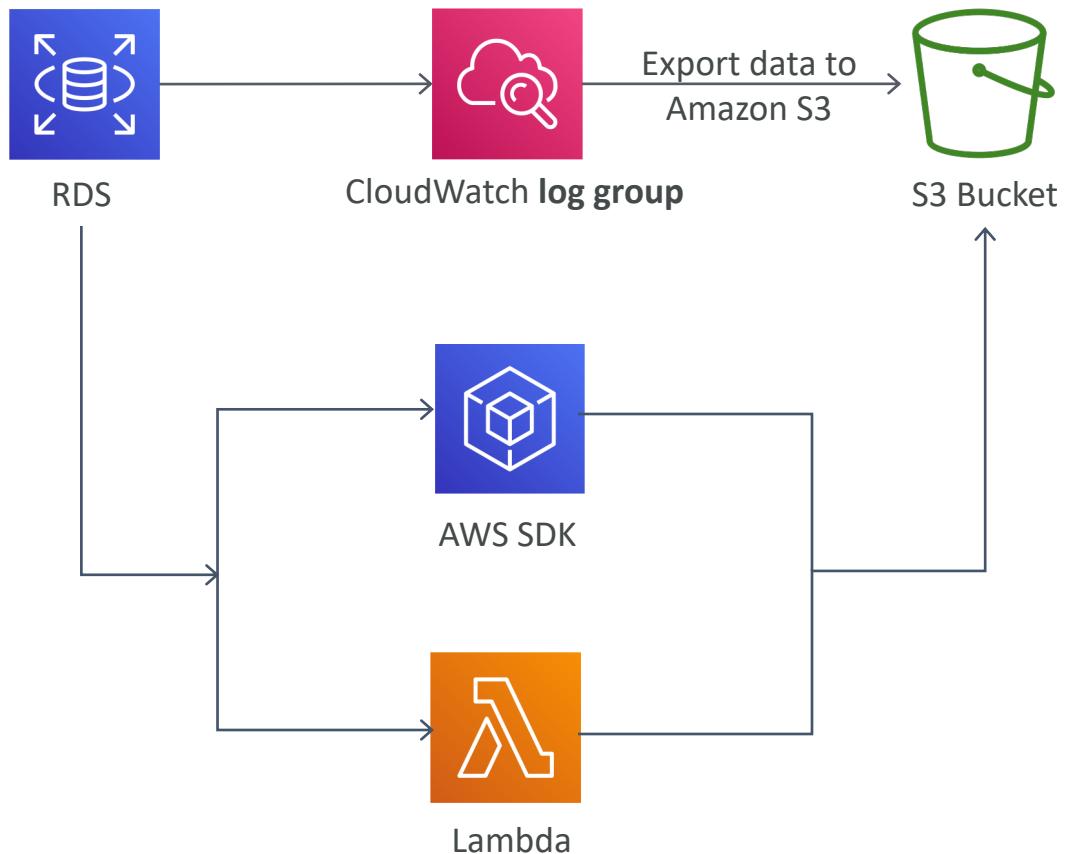
Advanced Auditing in Aurora MySQL

- To audit DB activity
- Can be viewed in Logs section of the database in the RDS console
- Enable with `server_audit_logging` parameter
- Use `server_audit_events` parameter to choose which events to log

Audit event	Description
CONNECT	Logs successful / failed connections and disconnections along with user info (log ins, log outs, failed login attempts)
QUERY	Logs all queries in plain text (including failed queries)
QUERY_DCL	Logs only DCL queries i.e. permission changes (GRANT, REVOKE etc.)
QUERY_DDL	Logs only DDL queries, i.e. schema changes (CREATE, ALTER etc.)
QUERY_DML	Logs only DML queries, i.e. data changes and reads (INSERT, UPDATE etc., and SELECT)
TABLE	Logs the tables that were affected by query execution

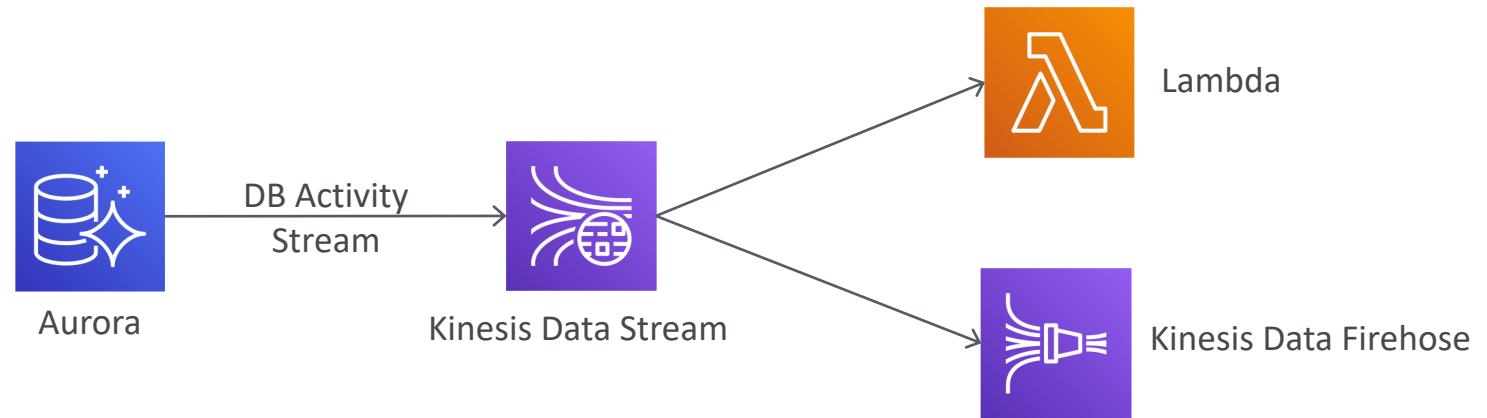
Exporting Aurora Logs

- Logs can be exported to CloudWatch Logs or S3 in the same way as you'd with any other RDS DB
- Alternate way to publish logs to CloudWatch Logs is to enable advanced auditing and set cluster level DB parameter
`server_audit_logs_upload = 1`



Database Activity Streams

- Near real-time data stream of the DB activity
- Can be used for monitoring, auditing and compliance purposes
- Aurora creates a Kinesis data stream and pushes the activity stream to it
- Can monitor your database activity in real time from Kinesis
- Can also be consumed by Firehose, Lambda etc.



Troubleshooting storage issues in Aurora

- Shared cluster storage is used for persistent data
- Local storage in the instance is used for temporary data and logs
- Local storage size depends on the instance class
- Monitor the with CloudWatch metrics `FreeLocalStorage / FreeableMemory` to see available and freeable local storage space
- Can also monitor **native counters** in Performance Insights
- Identify and fix the failing queries to prevent logs from using excessive storage
- Increase `max_heap_table_size / tmp_table_size` parameters (tables will consume more memory on the instance, reducing the local storage used)

ERROR: could not write block XXXXXXXX of temporary file: No space left on device.

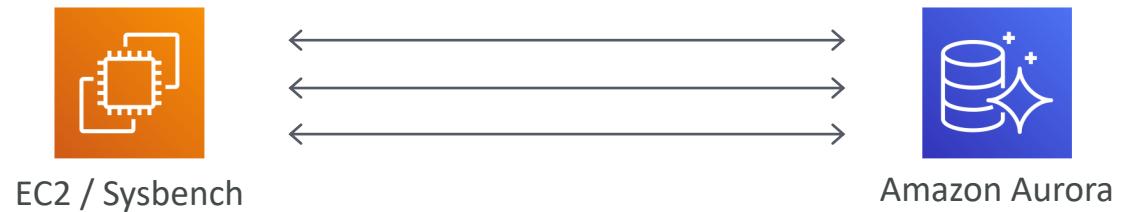
The free storage capacity for DB Instance: <instance> is low at x% of the provisioned storage ...

ERROR 1114 table full

- Scale up the DB instance class
- For PostgreSQL, you can also enable the `log_temp_files` parameter to monitor temporary tables and files
- Reduce `rds.log_retention` value to reclaim space

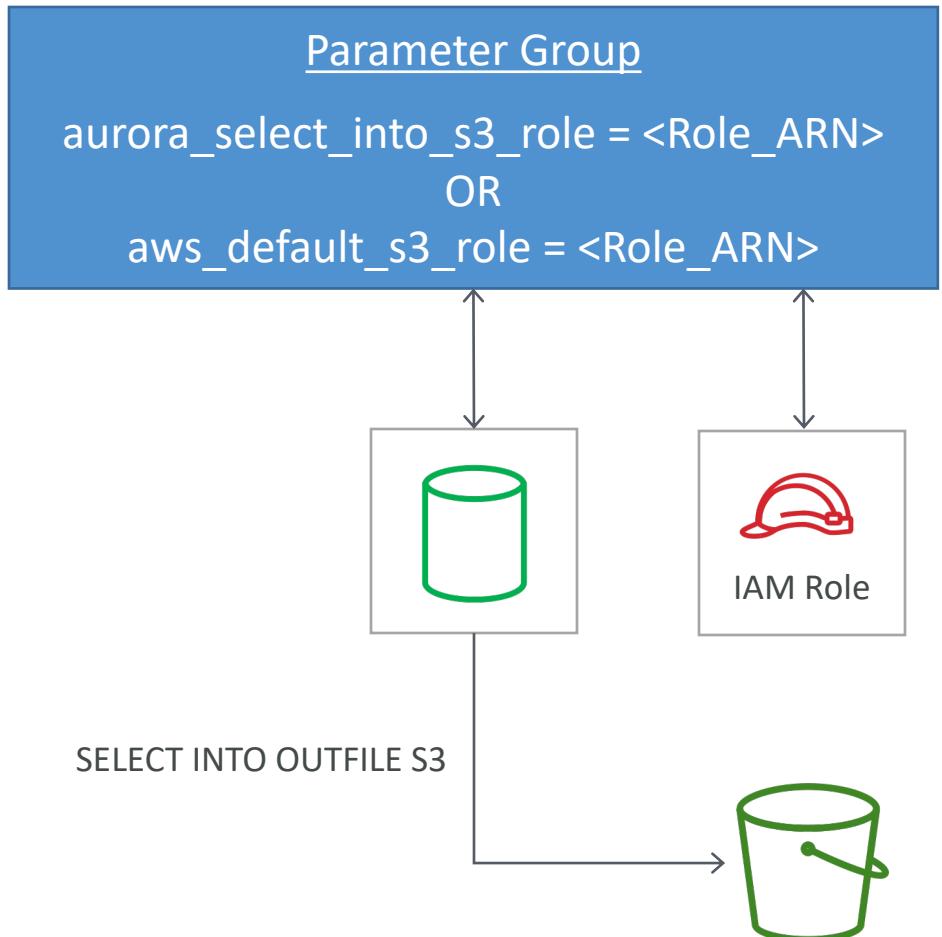
Aurora Benchmarking

- Aurora boasts 5x performance of MySQL and 3x performance of PostgreSQL
- AWS provides a benchmarking guide with a CloudFormation template to help customers replicate/validate this
- The Sysbench benchmarking tool is recommended for this purpose



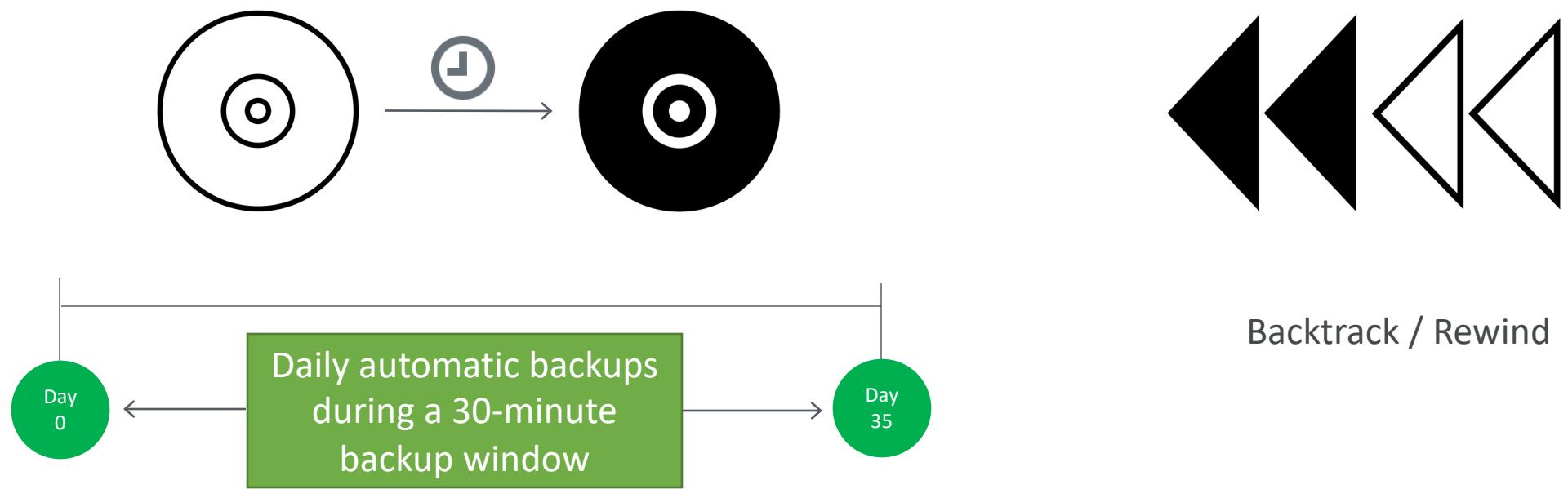
Exporting Data from Aurora into S3

- Create IAM role with a policy that allows Aurora to write to the given S3 bucket
- Attach the role to the Aurora cluster
 - add the role ARN to a cluster parameter group and attach the parameter group to the Aurora Cluster
- Now you can export data from Aurora DB using SQL
 - SELECT INTO OUTFILE S3
 - e.g. `SELECT * FROM db.table INTO OUTFILE S3 s3_file_path`
FIELDS TERMINATED BY '';
LINES TERMINATED BY '\n';



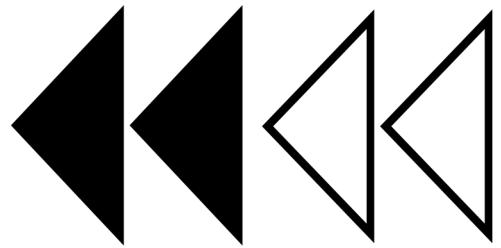
Aurora Backups and Backtracking

- Automatic backups and snapshots – same as in RDS
- Unlike RDS, you cannot disable automatic backups (min 1 day retention)
- Additional feature is Backtrack



Backtrack in Aurora

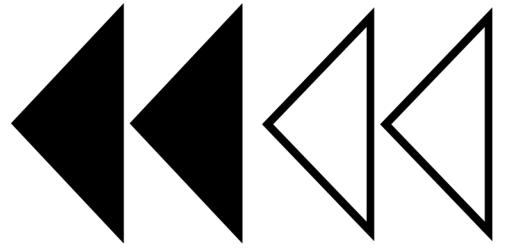
- Only for Aurora with MySQL compatibility
- Quickly rewind the DB cluster to a specific point in time
- Performs in-place restore (does not require restoring to a new instance)
- Lets you quickly recover from a user error
- Can restore when creating a new DB cluster or restoring a from a snapshot
- Up to 72 hours of PITR or “rewind” option (target backtrack window)
- Can repeatedly backtrack backward or forward in time
- You can backtrack the entire DB cluster but not individual instances or tables



Backtrack / Rewind

Backtrack in Aurora (contd.)

- DB cluster remains unavailable during a backtrack operation (few minutes downtime)
- Not a replacement for the good old backups feature
- Enable backtrack when creating/restoring the DB cluster
- Cross-region replication must be disabled to use backtrack
- Causes a brief DB instance disruption (must stop/pause applications before starting backtrack operation)
- Not supported with Aurora multi-master clusters



Backtrack / Rewind

Backups vs Snapshots vs Backtrack

Backups

- Automated
- Can only restore to a new instance (takes hours)
- Support PITR within backup retention period (up to 35 days)
- Great for unexpected failures

Snapshots

- Manually triggered
- Can only restore to a new instance (takes hours)
- Does not support PITR
- Great for known events like DB upgrades etc.

Backtrack

- Automated
- Supports in-place restore (takes minutes)
- Supports PITR up to 72 hours
- Great for undoing mistakes, for quick restores, for testing, and for exploring earlier data changes
- Can repeatedly backtrack backward or forward in time

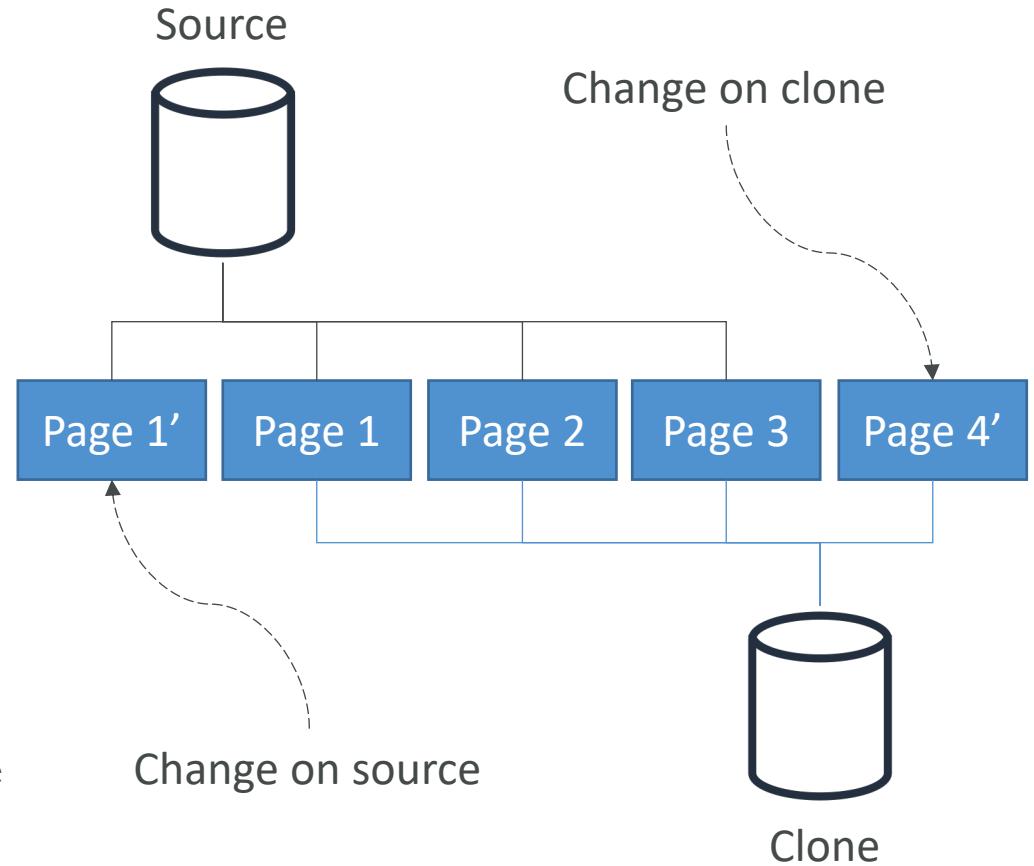
Aurora Backup, Restore, PITR and Backtrack



Demo

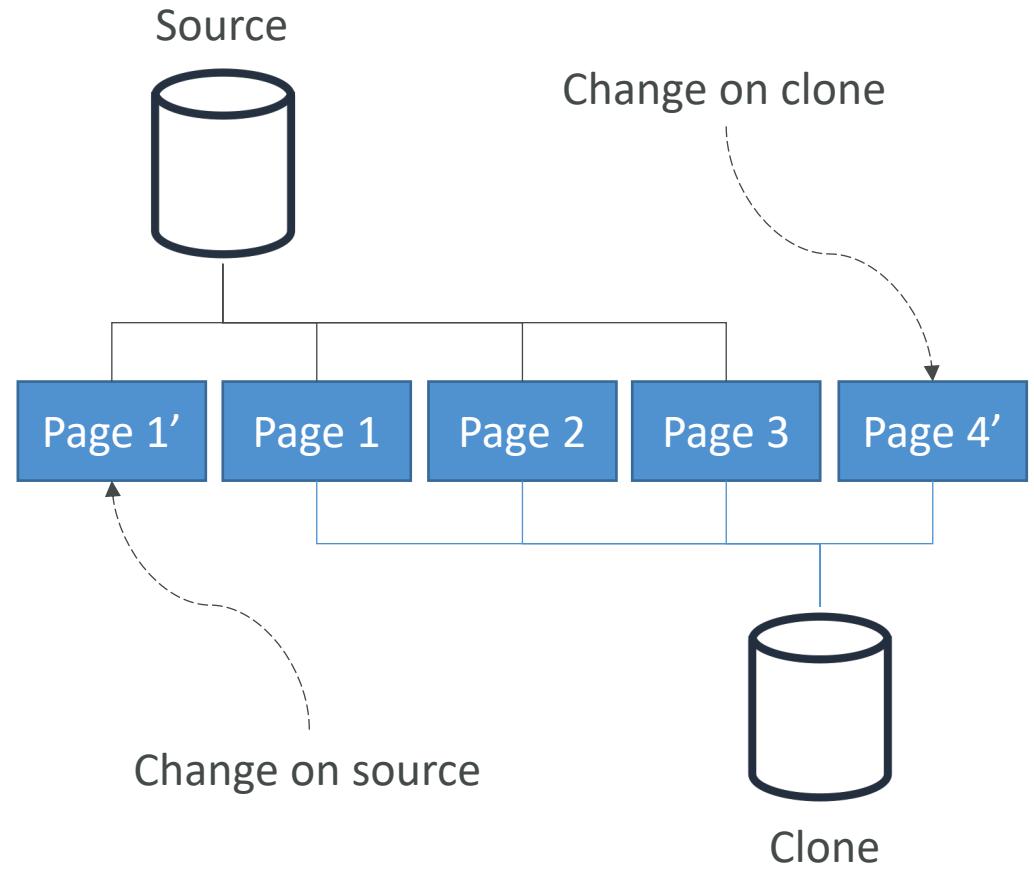
Cloning Databases in Aurora

- Different from creating read replicas – clones support R + W
- Different from replicating a cluster – clones use same storage layer as the source cluster
- Requires only minimal additional storage
- Can be created from existing clones
- Quick, cost-effective, no administrative effort
- Only within region (can be in different VPC, but same AZs)
- Supports cross-account cloning
- Uses a copy-on-write protocol
 - both source and clone share the same data initially
 - data that changes, is then copied at the time it changes either on the source or on the clone (i.e. stored separately from the shared data)
 - delta of writes after cloning is not shared



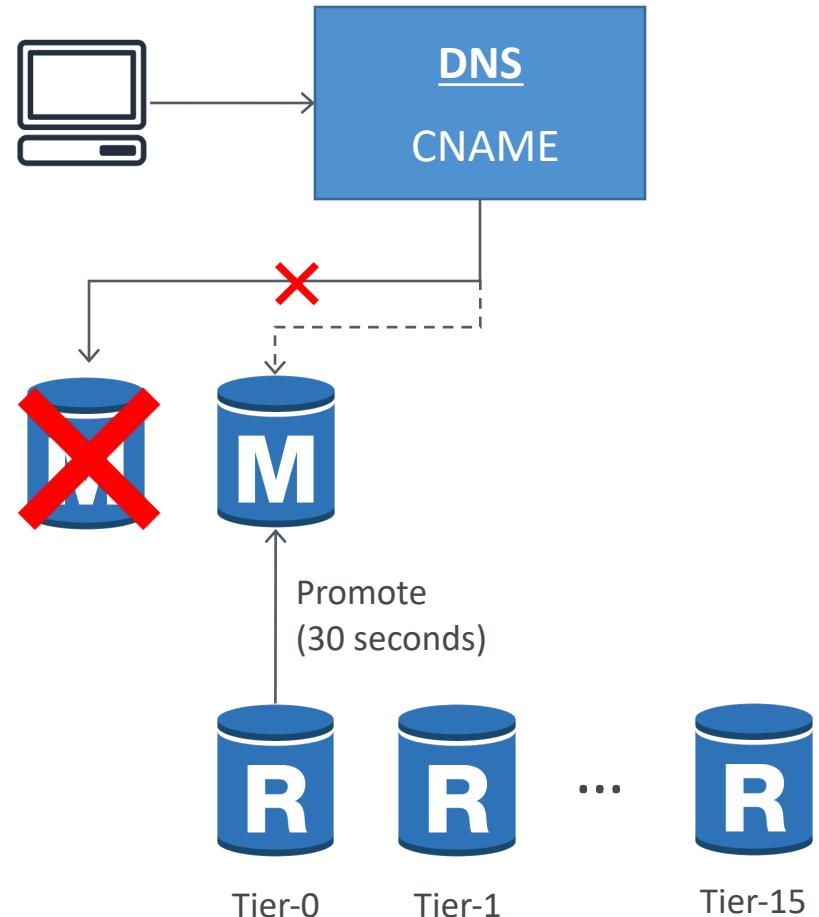
Cloning Databases in Aurora

- Use Cases
 - Create a copy of a production DB cluster for DEV/TEST/QA environment
 - Impact assessment of changes before applying to the main database – e.g. schema changes or parameter group changes
 - Perform workload-intensive ops – e.g. running analytical queries or exporting data for non-routine work
- Can't backtrack a clone to a time before that clone was created
- Cloning feature is only available in Aurora, not in RDS



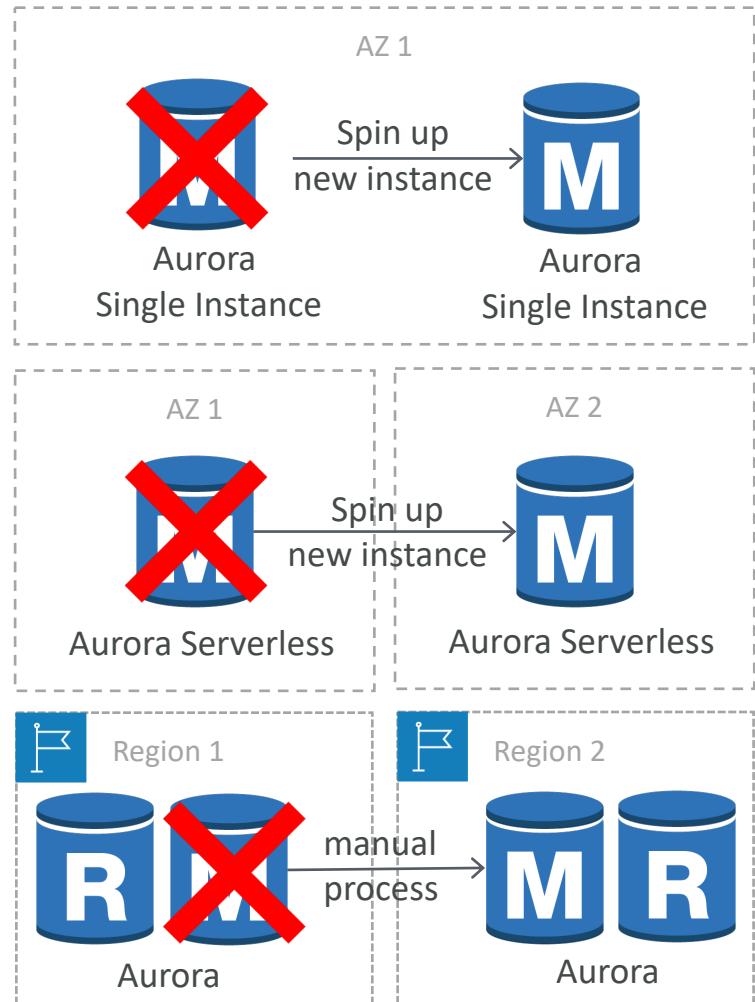
Aurora Failovers

- Failovers within region occur automatically
- A replica is promoted to be the new primary
- Which replica to promote is decided by replicas' failover priorities
- Failover priorities or failover tiers
 - Tier 0 (highest priority), through tier 15 (lowest priority)
 - Tier 0 replica gets promoted first
 - If two replicas have the same priority, then the replica that is largest in size gets promoted
 - If two replicas have same priority and size, then one of them gets promoted arbitrarily
- Aurora flips the CNAME of the DB instance to point to the replica and promotes it
- Typically takes 30 seconds (minimal downtime, 30-60 seconds of RTO)



Aurora Failovers

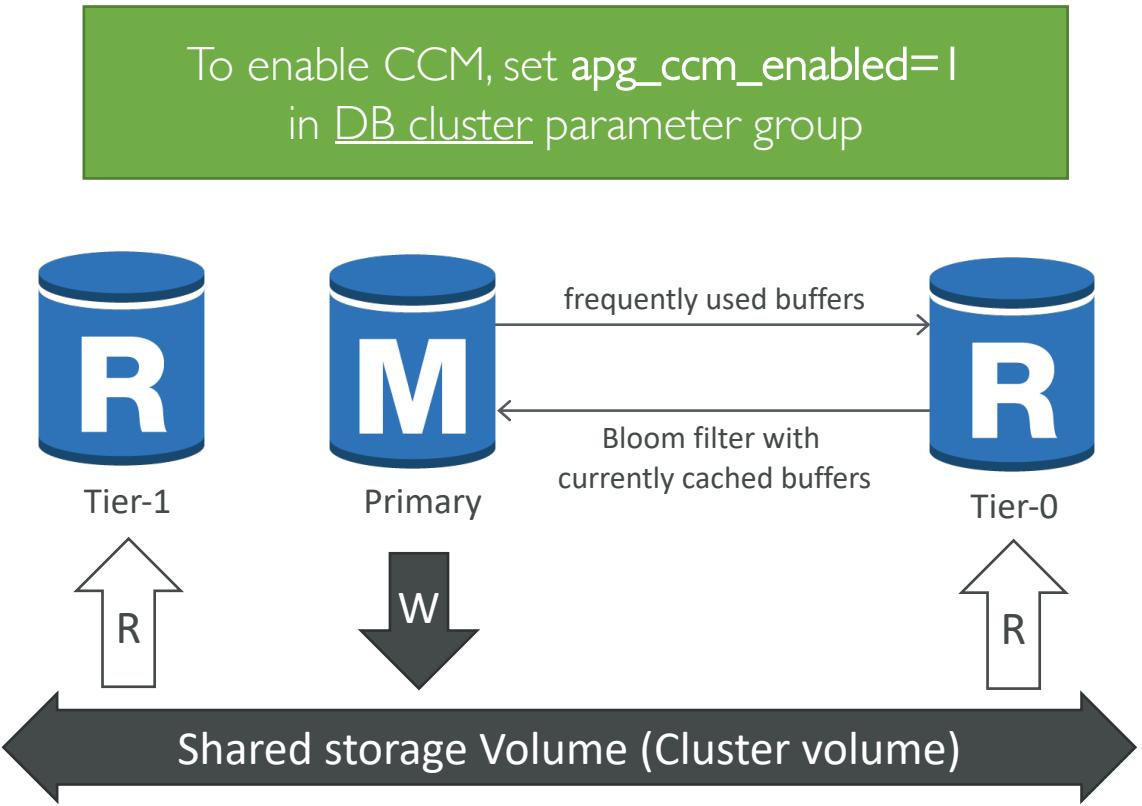
- In case of no replicas (single instance setup)
 - Aurora spins up a new master instance in the same AZ
 - results in additional downtime (as compared to an instance with replicas)
 - Best-effort basis (= may not succeed in case of AZ wide outage)
 - copying of data is not required due to shared storage architecture
- In case of Aurora Serverless
 - Aurora spins up a new instance in different AZ
 - DR across regions is a manual process
 - you promote a secondary region to take read/write workloads



Cluster cache management (CCM)

Aurora PostgreSQL Only

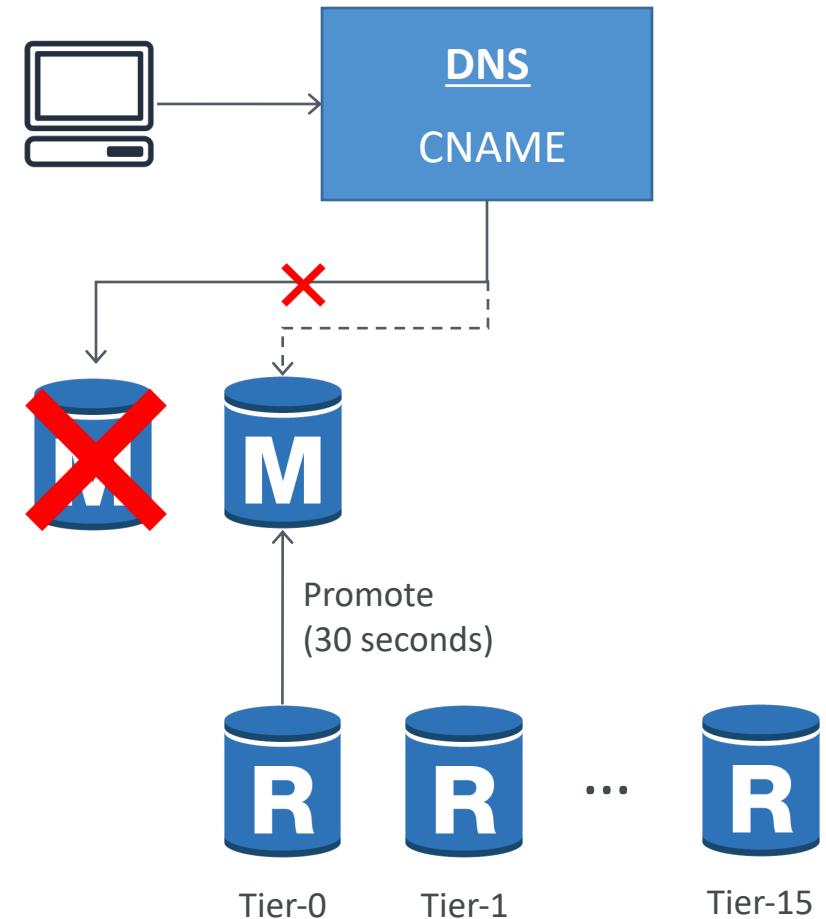
- Buffer cache is to reduce disk IO in RDBMS
- Cached content in primary and replica may be different
- Post a failover from primary to a replica, promoted replica takes some time to warm up its cache
- This causes slower response times post failover
- CCM improves the performance of the promoted instance post failover
- Replica preemptively reads frequently accessed buffers cached from the primary



A bloom filter is a space-efficient data structure

Simulating Fault Tolerance in Aurora

- Two ways to test/simulate fault tolerance
 - Manual failover
 - Fault injection queries
- Fault tolerance is synonymous to resiliency (or fault resiliency)
- You can use these options to simulate AZ Failure
- Can perform primary upgrade by force failover



Simulating fault tolerance w/ manual failover

- Select the master instance and choose Actions → Failover (or use `failover-db-cluster` command)
- Failover to the replica with highest failover priority will be triggered
- The read replica with highest failover priority will be the new master
- The master instance that failed over will become a replica when it comes online

DB identifier	Role	Engine	Region & AZ	Size	Status
database-1	Regional	Aurora MySQL	us-east-2	2 instances	Available
database-1-instance-1	Writer	Aurora MySQL	us-east-2a	db.r5.large	Available
database-1-instance-1-us-east-2b	Reader	Aurora MySQL	us-east-2b	db.r5.large	Available

- As each instance has its own endpoint address
- So you should clean up and re-establish any existing connections that use the old endpoint post a failover

Simulating failovers in Aurora



Demo

Aurora failover priorities



Demo

Simulating fault tolerance w/ fault injection queries

- Fault injection queries are issued as SQL commands
- You can schedule a simulated occurrence of different failure events
 - writer/reader crash
 - replica failure
 - disk failure
 - disk congestion



Fault injection queries – writer / reader crash

ALTER SYSTEM CRASH

[INSTANCE | DISPATCHER | NODE];



- Instance = DB instance (default crash type)
- Dispatcher = writes updates to the cluster volume
- Node = Instance + Dispatcher

Fault injection queries – replica failure

ALTER SYSTEM SIMULATE

percentage_of_failure PERCENT READ REPLICA FAILURE

[TO ALL | TO "replica name"]

FOR INTERVAL quantity

{YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE
SECOND };



- percentage_of_failure = % of requests to block
- TO ALL / TO = simulate failure of all or a specific replica
- quantity = duration of replica failure

Fault injection queries – disk failure

ALTER SYSTEM SIMULATE

percentage_of_failure PERCENT DISK FAILURE

[IN DISK index | NODE index]

FOR INTERVAL quantity

{ YEAR | QUARTER | MONTH | WEEK | DAY | HOUR |
MINUTE | SECOND };

- percentage_of_failure = % of the disk to mark as faulting
- DISK index = simulate failure of a specific logical block of data
- NODE index = simulate failure of a specific storage node
- quantity = duration of disk failure



Fault injection queries – disk congestion

ALTER SYSTEM SIMULATE

```
percentage_of_failure PERCENT DISK CONGESTION
BETWEEN minimum AND maximum MILLISECONDS
[ IN DISK index | NODE index ]
FOR INTERVAL quantity
{YEAR | QUARTER | MONTH | WEEK | DAY | HOUR |
MINUTE | SECOND};
```



- percentage_of_failure = % of the disk to mark as congested
- DISK index / NODE index = simulate failure of a specific disk or node
- minimum / maximum = min and max amount of congestion delay in milliseconds (a random number between the two will be used)
- quantity = duration of disk congestion

Fast failover in Aurora PostgreSQL

- Use CCM (apg_ccm_enabled=1)
- Use cluster / reader / custom endpoint (instead of instance endpoints)
 - Cleanup / re-establish connections, if using instance endpoints
 - Add automatic retry capability to the application
- **Aggressively set TCP keepalives (=low values)** - Quickly closes active connections if client is no longer able to connect to the DB
- **Reduce Java DNS Cache timeout value (low value for DNS TTL)**
 - Reader endpoint cycles through available readers. If a client caches DNS info, requests might go to the old instance until DNS cache times out
- Use separate connection objects for long and short running queries
- Use Aurora PostgreSQL connection string with multiple hosts

Recommended TCP keepalive settings

- `tcp_keepalive_time = 1 (second)`
- `tcp_keepalive_intvl = 1 (second)`
- `tcp_keepalive_probes = 5`

Fast failover in Aurora PostgreSQL (contd.)

- Use a list of hosts in your JDBC connection string
 - JDBC connection driver will loop through all nodes on this list to find a valid connection
 - Set connection parameters (**in red**) aggressively so your app doesn't wait too long on any host
 - Alternatively, you can also maintain a file containing instance endpoints
 - When you add/remove nodes, you must update this file
- Or maintain a file containing cluster endpoints (reader / writer)
- Your application can read this file to populate the host section of the connection string

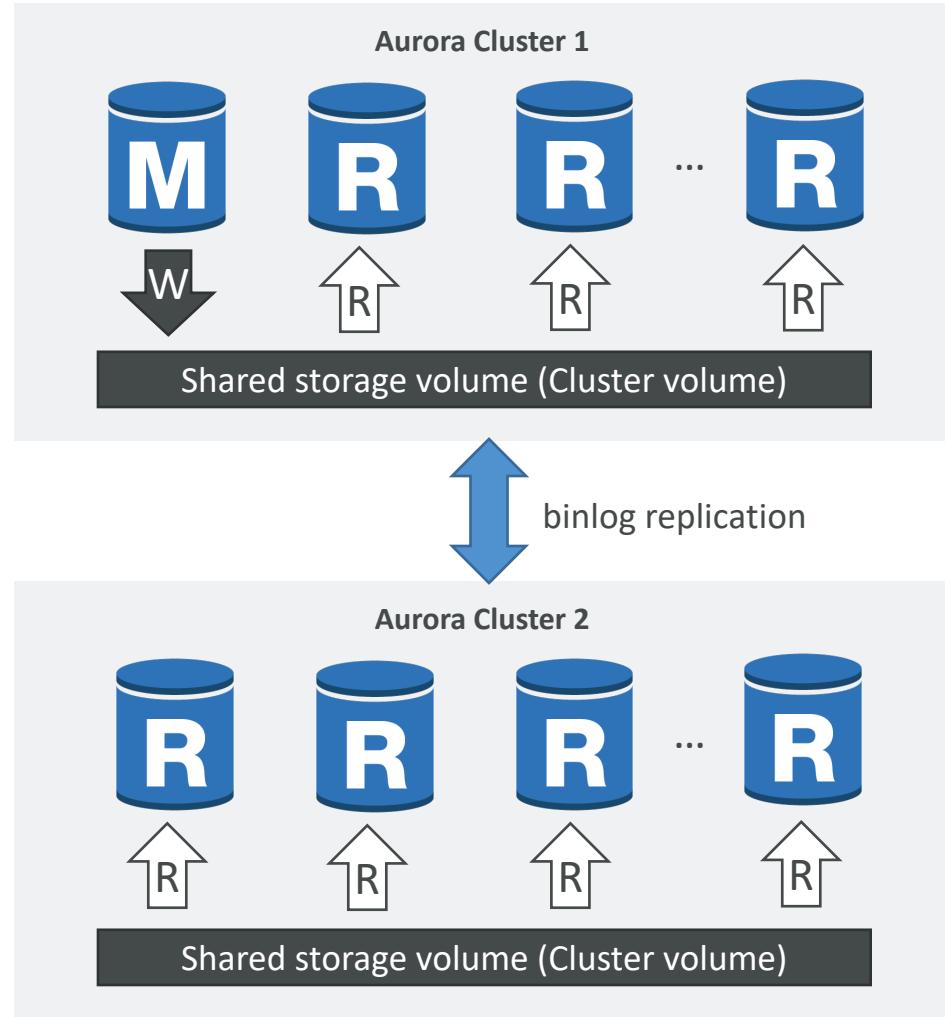
```
jdbc:postgresql://  
database-1.cluster-cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432,  
database-1.cluster-ro-cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432  
/postgres?user=<username>&password=<password>&loginTimeout=2&  
connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60&  
tcpKeepAlive=true&targetServerType=master&loadBalanceHosts=true
```

```
database-1.cluster-cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432,  
database-1.cluster-ro-cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432
```

```
database-1-instance-1.cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432,  
database-2-instance-1-us-east-2a.cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432,  
database-2-instance-1-us-east-2b.cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432,  
database-2-instance-1-us-east-2c.cqxc86vdny2e.us-east-2.rds.amazonaws.com:5432
```

Cluster Replication Options for Aurora MySQL

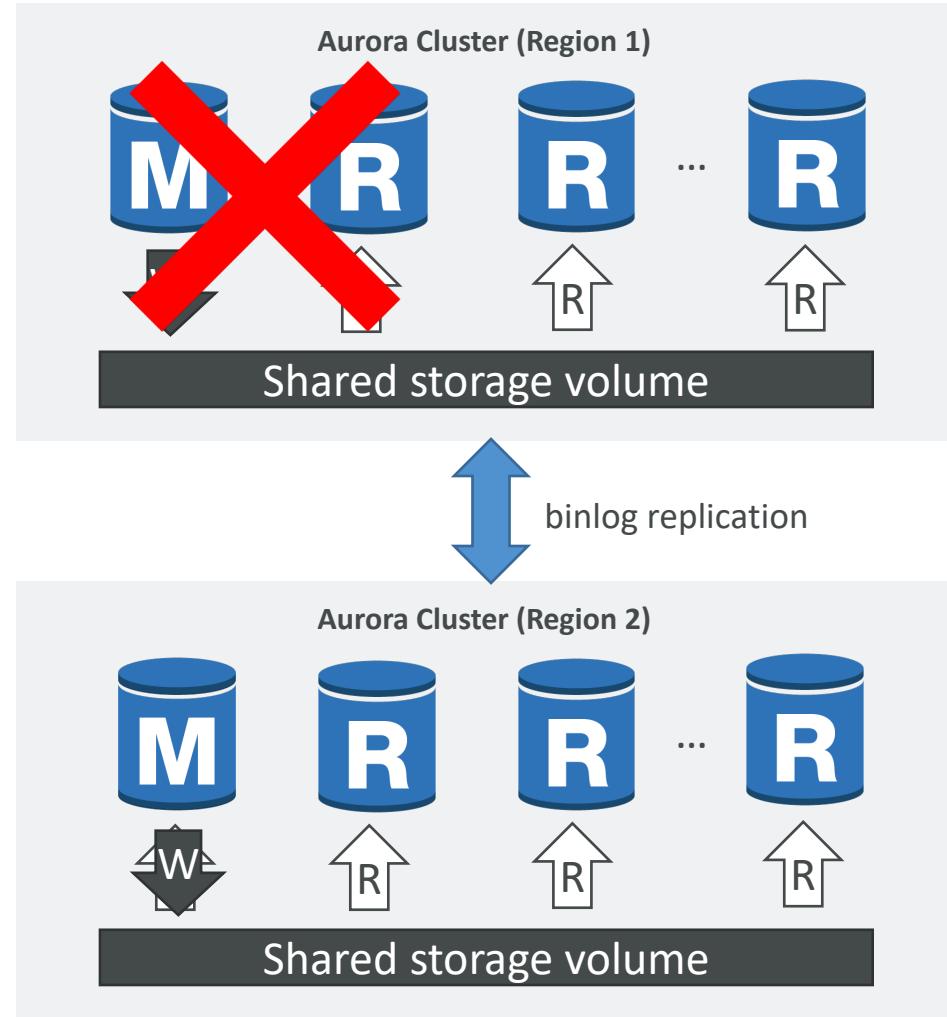
- Replication between clusters = can have more than 15 read replicas
- Replication
 - between two Aurora MySQL DB clusters in different regions (Cross-Region Replication)
 - between two Aurora MySQL DB clusters in same region
 - between RDS MySQL DB instance and an Aurora MySQL DB cluster



Cluster Replication Options for Aurora MySQL

Cross-Region Replication b/w two Aurora MySQL DB clusters

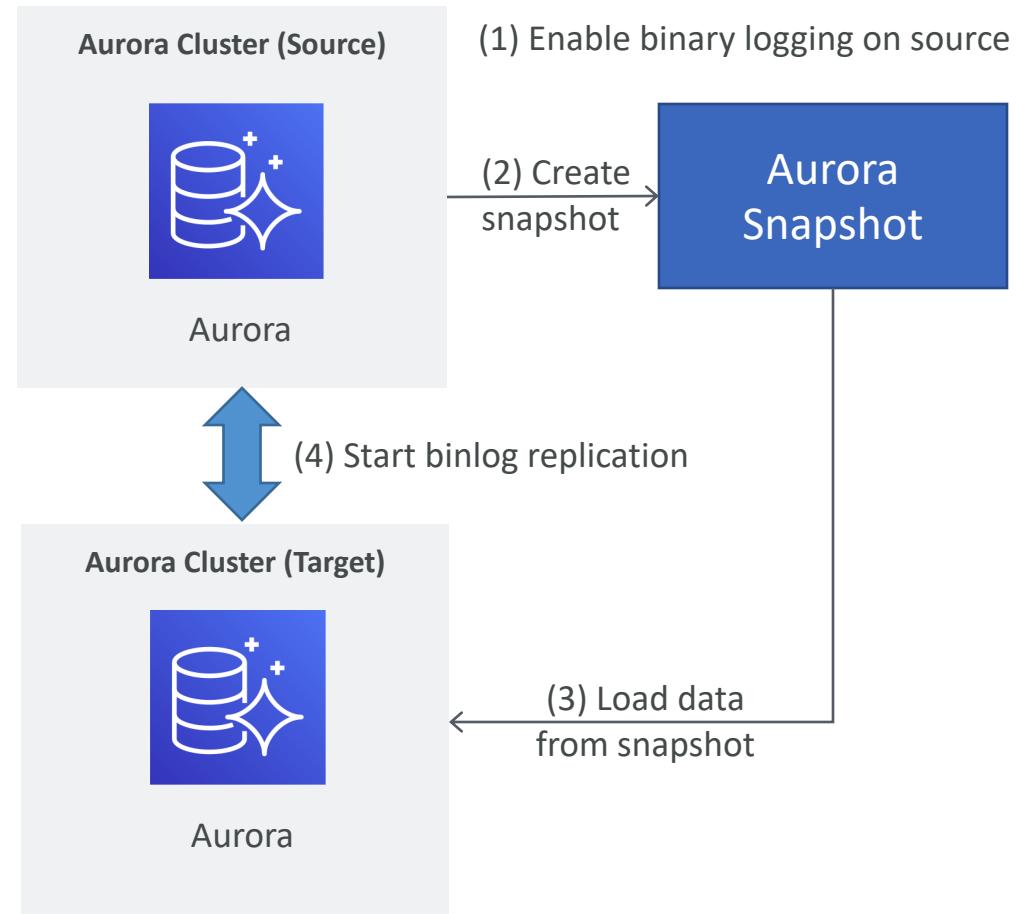
- enable binary logging (binlog_format parameter)
- then, create a cross-region read replica in another region
- you can promote the replica to a standalone DB cluster (typically, for DR purposes)



Cluster Replication Options for Aurora MySQL

Replication b/w two Aurora MySQL DB clusters in same region

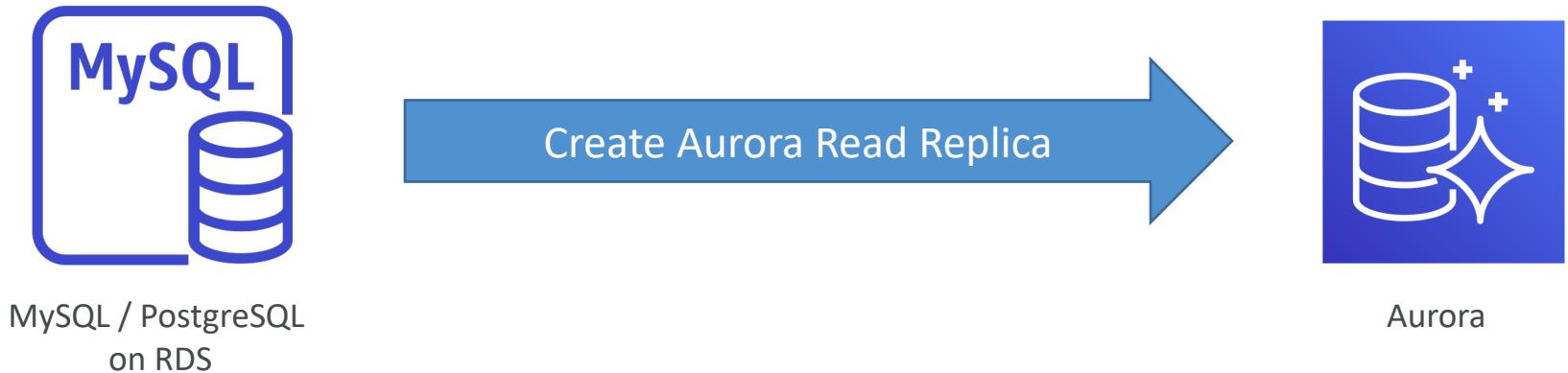
- enable binary logging (binlog_format parameter) on source
- then, replicate using a snapshot of the replication master



Cluster Replication Options for Aurora

Replication b/w RDS DB instance and an Aurora DB cluster

- By creating an Aurora read replica of RDS DB instance
- Typically used for migration to Aurora rather than ongoing replication
 - To migrate, stop the writes on master. After replication lag is zero, promote the Aurora replica as a standalone Aurora DB cluster



Aurora Replicas vs MySQL Replicas

Feature	Amazon Aurora Replicas	MySQL Replicas
Number of replicas	Up to 15	Up to 5
Replication type	Asynchronous (milliseconds)	Asynchronous (seconds)
Performance impact on primary	Low	High
Replica location	In-region	Cross-region
Act as failover target	Yes (no data loss)	Yes (potentially minutes of data loss)
Automated failover	Yes	No
Support for user-defined replication delay	No	Yes
Support for different data or schema vs. primary	No	Yes

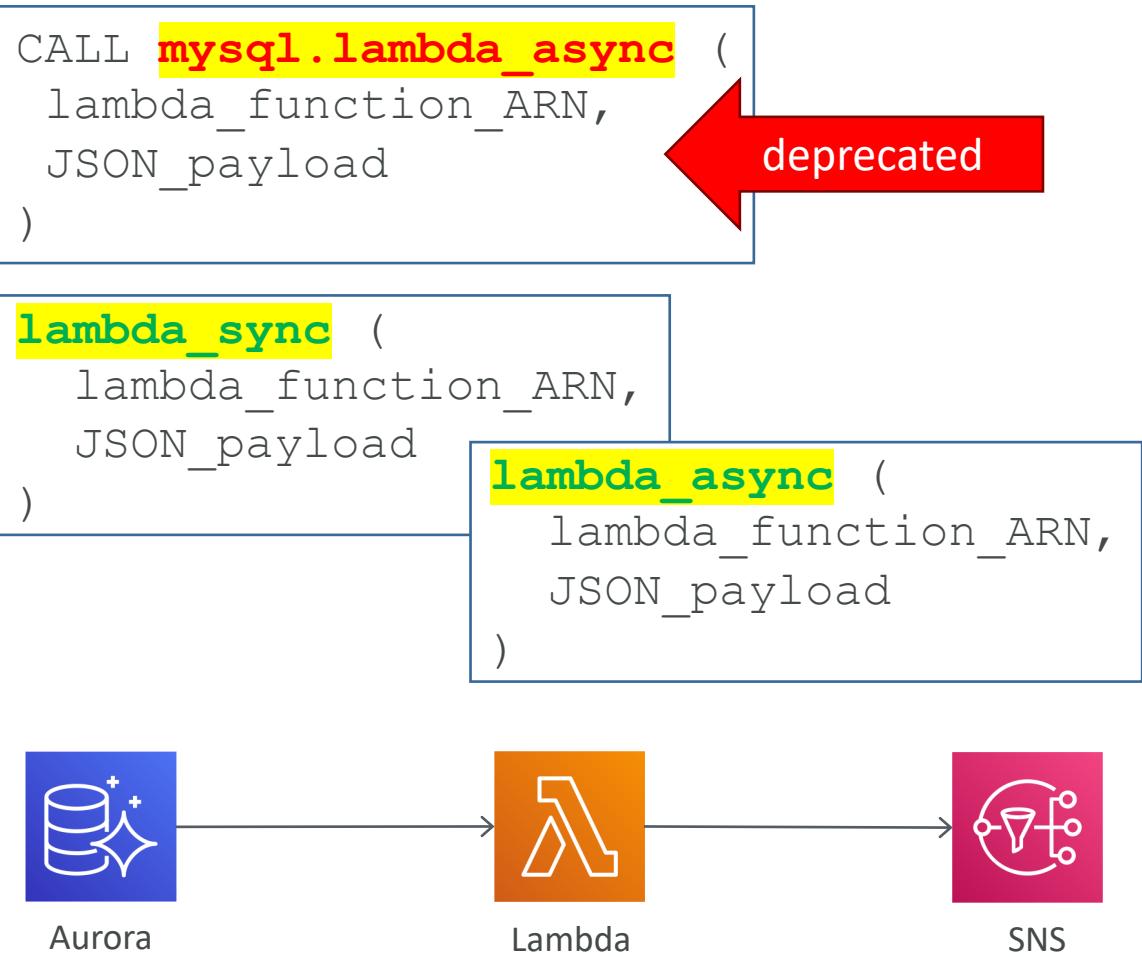
Comparison of RDS Deployments

	Read replicas	Multi-AZ deployments (Single-Region)	Multi-Region deployments
Main purpose	Scalability	HA	DR and performance
Replication method	Asynchronous (eventual consistency)	Synchronous	Asynchronous (eventual consistency)
		Asynchronous (Aurora)	
Accessibility	All replicas can be used for read scaling	Active-Passive (standby not accessible)	All regions can be used for reads
		Active-Active (Aurora)	
Automated backups	No backups configured by default	Taken from standby	Can be taken in each region
		Taken from shared storage layer (Aurora)	
Instance placement	Within-AZ, Cross-AZ, or Cross-Region	At least two AZs within region	Each region can have a Multi-AZ deployment
Upgrades	Independent from source instance	On primary	Independent in each region
		All instances together (Aurora)	
DR (Disaster Recovery)	Can be manually promoted to a standalone instance	Automatic failover to standby	Aurora allows promotion of a secondary region to be the master
	Can be promoted to primary (Aurora)	Automatic failover to read replica (Aurora)	

<https://aws.amazon.com/rds/features/multi-az/>

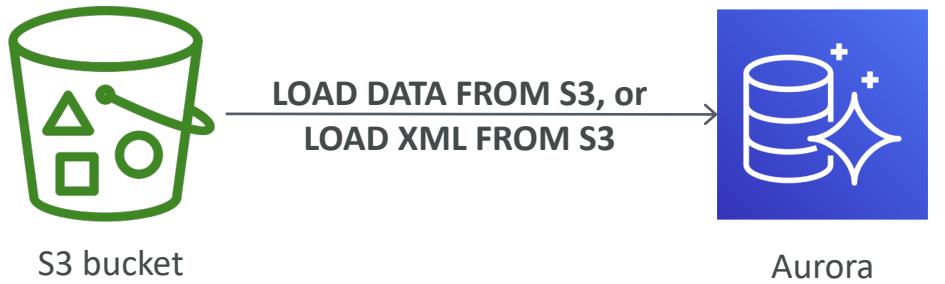
Invoke Lambda functions from Aurora MySQL

- Give Aurora MySQL access to Lambda by setting DB cluster parameter
`aws_default_lambda_role` = IAM role ARN
- Option 1 – Using `mysql.lambda_async` procedure (deprecated)
 - Wrap calls in a stored procedure and call through triggers or application code
- Option 2 – Using native functions `lambda_sync` and `lambda_async`
 - User must have `INVOKE LAMBDA` privilege
 - `GRANT INVOKE LAMBDA ON *.* TO user@host`



Load data from S3 into Aurora MySQL

- Use SQL statements
 - LOAD DATA FROM S3, or
 - LOAD XML FROM S3
- Must give the Aurora cluster access to S3 by setting DB cluster parameter
 - `aurora_load_from_s3_role` = IAM role ARN, or
 - `aws_default_s3_role` = IAM role ARN
- User must have **LOAD FROM S3** privilege
 - `GRANT LOAD FROM S3 ON *.* TO user@host`

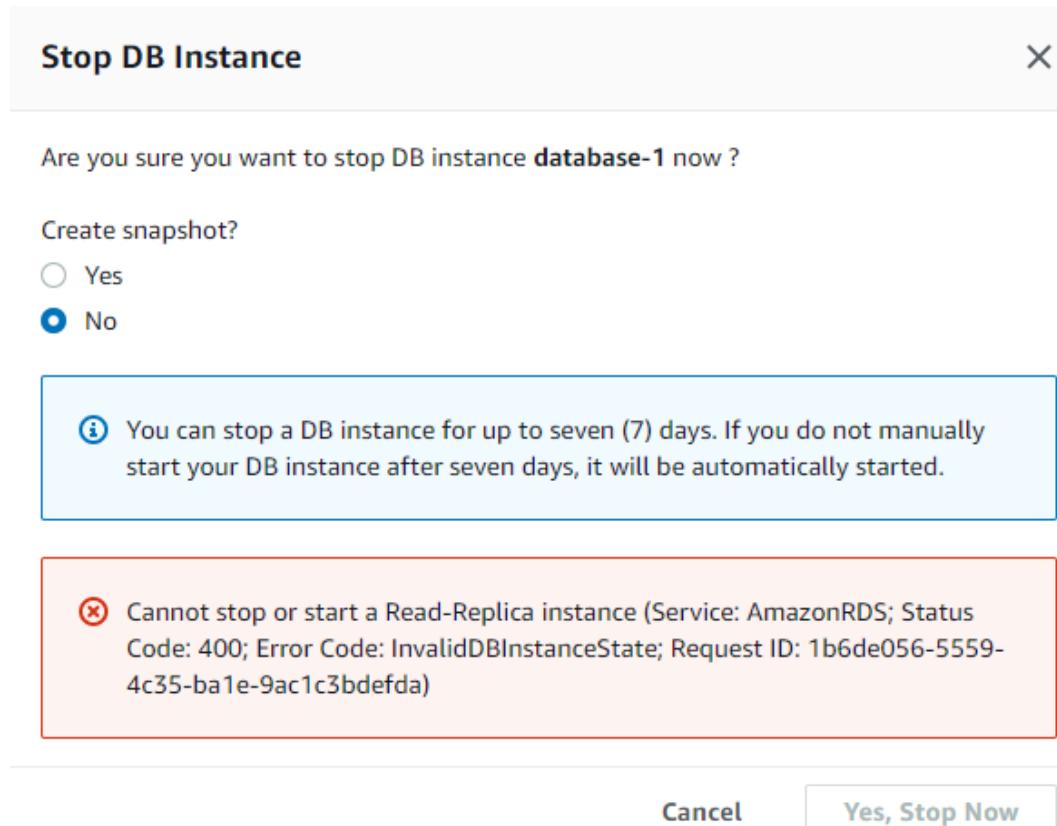


Example:

```
LOAD DATA FROM S3 's3://bucket/users.csv'  
INTO TABLE table_name  
FIELDS TERMINATED BY ','  
LINES TERMINATED BY '\n'  
(ID, FIRSTNAME, LASTNAME, EMAIL);
```

RDS / Aurora – Good things to know

- Stopping an RDS DB
 - Can stop an RDS instance only if it does not have a replica
 - Cannot stop an RDS replica
 - Cannot stop RDS for SQL Server DB instance if it's in Multi-AZ
- Stopping an Aurora DB
 - Can stop the cluster; not the individual instances
 - Cannot manually stop Aurora Serverless
 - Cannot stop Aurora Multi-Master or Aurora Global DB cluster
 - Cannot stop a cluster if it uses parallel query
 - Cannot delete a stopped cluster without starting it first
- If you don't manually start your DB instance/Cluster after seven days, it will be automatically started



RDS / Aurora – Good things to know

- Maximum Connections in RDS or Aurora is controlled via parameter groups
- Each DB engine has a specified formula for the default max connections value
- You can override this value using a custom parameter group
- Ideally, you'd want to scale the instance to get higher max connections

The image displays three separate screenshots of the AWS RDS Parameter Groups interface, each showing the configuration for the `max_connections` parameter.

default.aurora5.6

Name	Values	Allowed values	Apply type
max_connections	GREATEST({log(DBInstanceClassMemory/805306368)*45}, {log(DBInstanceClassMemory/8187281408)*1000})	1-16000	dynamic

default.aurora-postgresql11

Name	Values	Allowed values	Apply type
max_connections	LEAST({DBInstanceClassMemory/9531392},5000)	6-262143	static

default.mysql8.0

Name	Values	Allowed values	Apply type
max_connections	{DBInstanceClassMemory/12582880}	1-100000	dynamic
mysqlx_max_connections	100	1-65535	dynamic

DynamoDB and DAX

High Performance at any scale!

Amazon DynamoDB – Overview

KEY-VALUE



DynamoDB



DAX

- Non-relational Key-Value store
- Fully Managed, Serverless, NoSQL database in the cloud
- Fast, Flexible, Cost-effective, Fault Tolerant, Secure
- Multi-region, multi-master database (Global Tables)
- Backup and restore with PITR (Point-in-time Recovery)
- **Single-digit millisecond performance** at any scale
- In-memory caching with DAX (DynamoDB Accelerator; **microsecond latency**)
- Supports CRUD (Create/Read/Update/Delete) operations through APIs
- Supports transactions across multiple tables (ACID support)
- No direct analytical queries (No joins)
- Access patterns must be known ahead of time for efficient design and performance

Creating a DynamoDB Table



Demo



DataCumulus | RIZMAXed

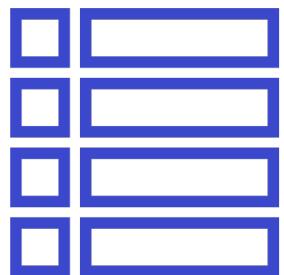
DynamoDB Terminology Compared to SQL

SQL / RDBMS	DynamoDB
Tables	Tables
Rows	Items
Columns	Attributes
Primary Keys – Multicolumn and optional	Primary Keys – Mandatory, minimum one and maximum two attributes
Indexes	Local Secondary Indexes
Views	Global Secondary Indexes

DynamoDB Tables

- Tables are top-level entities
- No strict inter-table relationships (Independent Entities)
- You control performance at the table level
- Table items stored as JSON (DynamoDB-specific JSON)
- Primary keys are mandatory, rest of the schema is flexible
- Primary Key can be simple or composite
- Simple Key has a single attribute (=partition key or hash key)
- Composite Key has two attributes
(=partition/hash key + sort/range key)
- Non-key attributes (including secondary key attributes) are optional

DynamoDB



DynamoDB Table

DynamoDB Table Items

Table with Simple Primary Key

```
{  
    "partition_key" :"...","  
    "attribute_1" :"...","  
    "attribute_2" :"..."  
}
```

```
{  
    "partition_key" :"...","  
    "attribute_1" :"...","  
    "attribute_4" :"...","  
    "attribute_5" :"..."  
}
```

Table with Composite Primary Key

```
{  
    "partition_key" :"...","  
    "sort_key" :"...","  
    "attribute_1" :"...","  
    "attribute_3" :"..."  
}
```

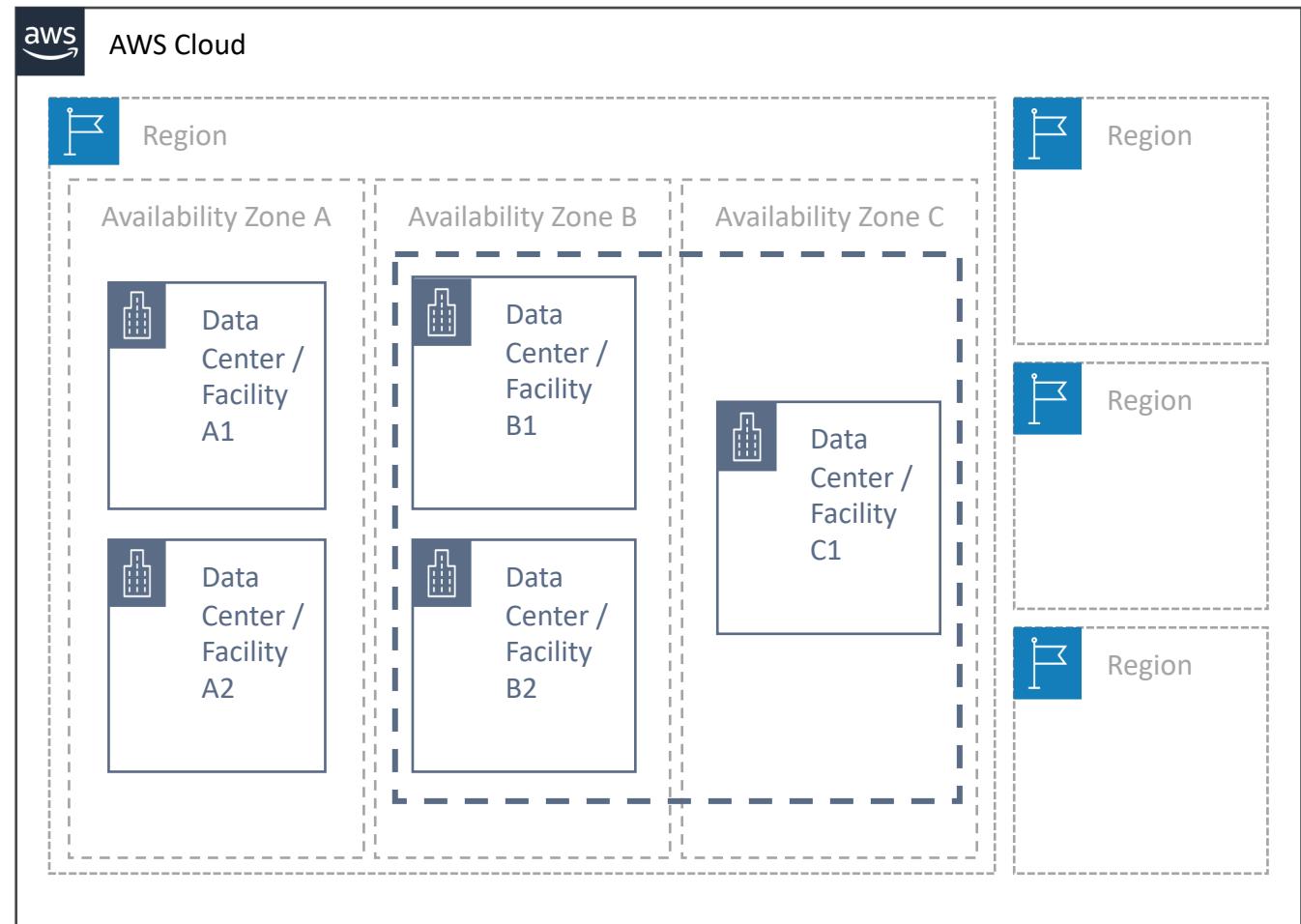
```
{  
    "partition_key" :"...","  
    "sort_key" :"...","  
    "attribute_3" :"...","  
    "attribute_4" :"..."  
}
```

Data Types in DynamoDB

- Scalar Types
 - Exactly one value
 - e.g. string, number, binary, boolean, and null
 - Keys or index attributes only support string, number and binary scalar types
- Set Types
 - Multiple scalar values
 - e.g. string set, number set and binary set
- Document Types
 - Complex structure with nested attributes
 - e.g. list and map

AWS Global Infrastructure

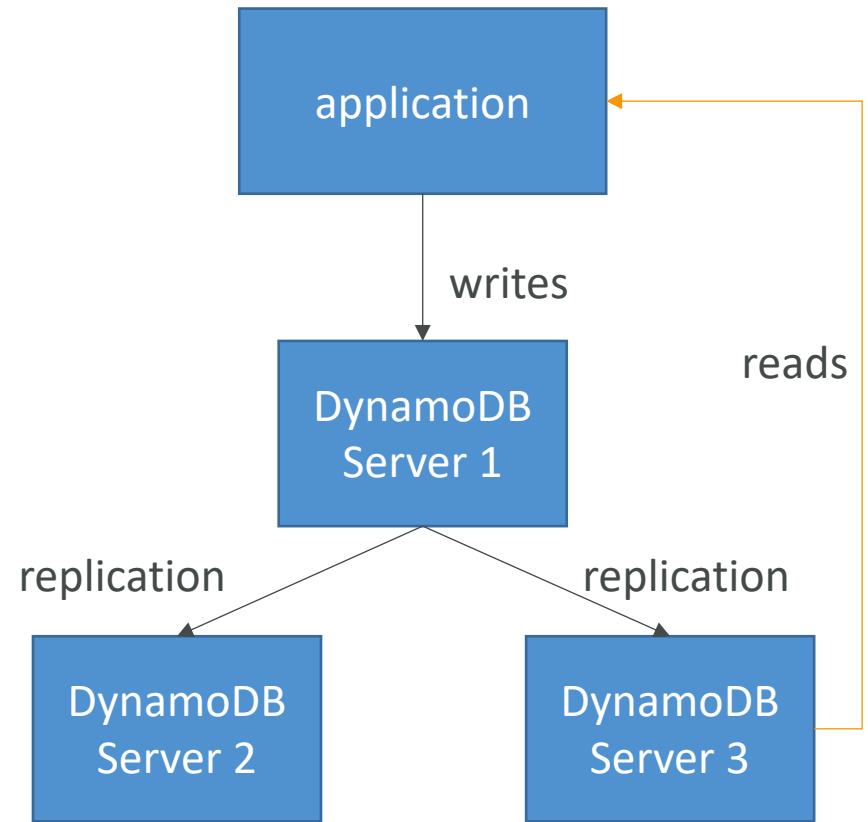
- Has multiple AWS Regions across the globe
- Each region has one or more AZs (Availability Zones)
- Each AZ has one or more facilities (= Data Centers)
- DynamoDB automatically replicates data between multiple facilities within the AWS region
- Near Real-time Replication
- AZs act as independent failure domains



DynamoDB Consistency

- Read Consistency: **strong consistency**, **eventual consistency**, and **transactional**
- Write Consistency: **standard** and **transactional**
- Strong Consistency
 - The most up-to-date data
 - Must be requested explicitly
- Eventual Consistency
 - May or may not reflect the latest copy of data
 - Default consistency for all operations
 - 50% cheaper than strong consistency
- Transactional Reads and Writes
 - For ACID support across one or more tables within a single AWS account and region
 - 2x the cost of strongly consistent reads
 - 2x the cost of standard writes

Strongly Consistent Read vs Eventually Consistent Read



- **Eventually Consistent Read:** If we read just after a write, it's possible we'll get unexpected response because of replication
- **Strongly Consistent Read:** If we read just after a write, we will get the correct data
- **By default:** DynamoDB uses Eventually Consistent Reads, but GetItem, Query & Scan provide a "ConsistentRead" parameter you can set to True

DynamoDB Transactions – transactional consistency

- AccountBalance Table

Account_id	balance	Last_transaction_ts
Acct_21	230	1562503085
Acct_45	120	1562503085

- BankTransactions Table

Transaction_id	Transaction_time	From_account_id	To_account_id	value
Tx_12345	1561483349	Acct_45	Acct_21	45
Tx_23456	1562503085	Acct_21	Acct_45	100

- A transaction is a write to both tables, or none!

DynamoDB Pricing Model

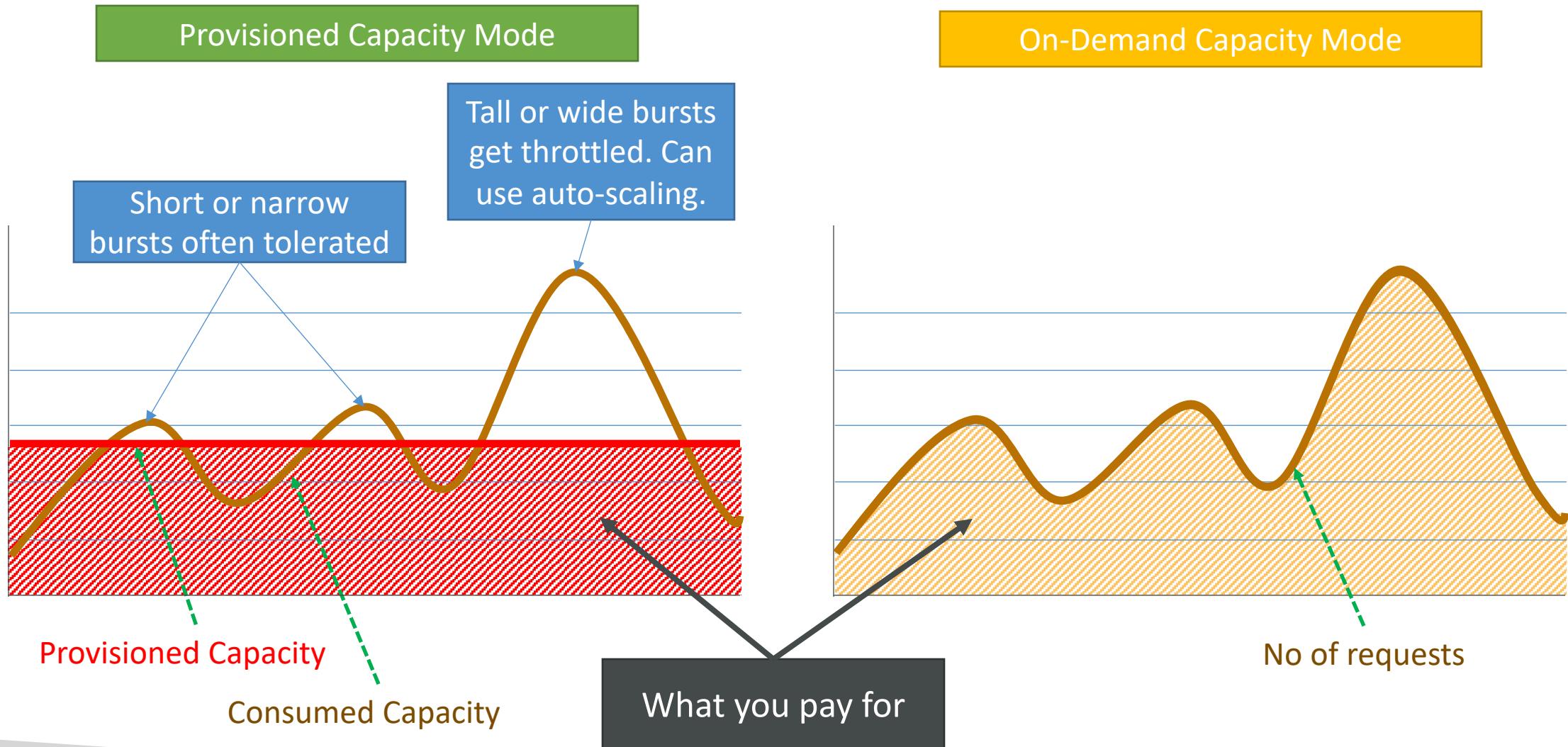
Provisioned Capacity

- You pay for the capacity you provision (= number of reads and writes **per second**)
- You can use auto-scaling to adjust the provisioned capacity
- Uses Capacity Units: Read Capacity Units (RCUs) and Write Capacity Units (WCUs)
- Consumption beyond provisioned capacity may result in throttling
- Use **Reserved Capacity** for discounts over 1 or 3-year term contracts (you're charged a one-time fee + an hourly fee per 100 RCUs and WCUs)
- for throughput calculation purposes, request units are equivalent to capacity units
- storage, backup, replication, streams, caching, data transfer out charged additionally

On-Demand Capacity

- You pay **per request** (= number of read and write requests your application makes)
- No need to provision capacity units
- DynamoDB instantly accommodates your workloads as they ramp up or down
- Uses Request Units: Read Request Units and Write Request Units
- Cannot use reserved capacity with On-Demand mode

Provisioned Capacity vs On-Demand Mode



DynamoDB Throughput

Provisioned Capacity mode

- Uses Capacity Units
 - 1 capacity unit = 1 request/sec
- RCUs (Read Capacity Units)
 - In blocks of 4KB, last block always rounded up
 - 1 strongly consistent table read/sec = 1 RCU
 - 2 eventually consistent table reads/sec = 1 RCU
 - 1 transactional read/sec = 2 RCUs
- WCUs (Write Capacity Units)
 - In blocks of 1KB, last block always rounded up
 - 1 table write/sec = 1 WCU
 - 1 transactional write/sec = 2 WCUs

On-Demand Capacity mode

- Uses Request Units
 - Same as Capacity Units for calculation purposes
- Read Request Units
 - In blocks of 4KB, last block always rounded up
 - 1 strongly consistent table read request = 1 RRU
 - 2 eventually consistent table read request = 1 RRU
 - 1 transactional read request = 2 RRUs
- Write Request Units
 - In blocks of 1KB, last block always rounded up
 - 1 table write request = 1 WRU
 - 1 transactional write request = 2 WRUs

Provisioned Capacity vs On-Demand Mode

Provisioned Capacity Mode

- Typically used in production environment
- Use this when you have **predictable traffic**
- Consider using reserved capacity if you have **steady and predictable traffic** for **cost savings**
- Can result in throttling when consumption shoots up (use auto-scaling)
- Tends to be cost-effective as compared to the on-demand capacity mode

On-Demand Capacity Mode

- Typically used in dev/test environments or for small applications
- Use this when you have **variable, unpredictable traffic**
- Instantly accommodates up to **2x** the previous peak traffic on a table
- Throttling can occur if you exceed 2x the previous peak within 30 minutes
- Recommended to space traffic growth over at least 30 mins before driving more than 2x

Example 1: Calculating Capacity Units

Calculate capacity units to read and write a 15KB item

- RCUs with strong consistency:
 - $15\text{KB}/4\text{KB} = 3.75 \Rightarrow$ rounded up $\Rightarrow 4 \text{ RCUs}$
- RCUs with eventual consistency:
 - $(1/2) \times 4 \text{ RCUs} = 2 \text{ RCUs}$
- RCUs for transactional read:
 - $2 \times 4 \text{ RCUs} = 8 \text{ RCUs}$
- WCUs:
 - $15\text{KB}/1\text{KB} = 15 \text{ WCUs}$
- WCUs for transactional write:
 - $2 \times 15 \text{ WCUs} = 30 \text{ WCUs}$



Example 2: Calculating Capacity Units

Calculate capacity units to read and write a 1.5KB item

- RCUs with strong consistency:
 - $1.5\text{KB}/4\text{KB} = 0.375 \Rightarrow$ rounded up $\Rightarrow 1 \text{ RCU}$
- RCUs with eventual consistency:
 - $(1/2) \times 1 \text{ RCU} = 0.5 \text{ RCU} \Rightarrow$ rounded up $= 1 \text{ RCU}$
- RCUs for transactional read:
 - $2 \times 1 \text{ RCU} = 2 \text{ RCUs}$
- WCUs:
 - $1.5\text{KB}/1\text{KB} = 1.5 \Rightarrow$ rounded up $\Rightarrow 2 \text{ WCUs}$
- WCUs for transactional write:
 - $2 \times 2 \text{ WCUs} = 4 \text{ WCUs}$



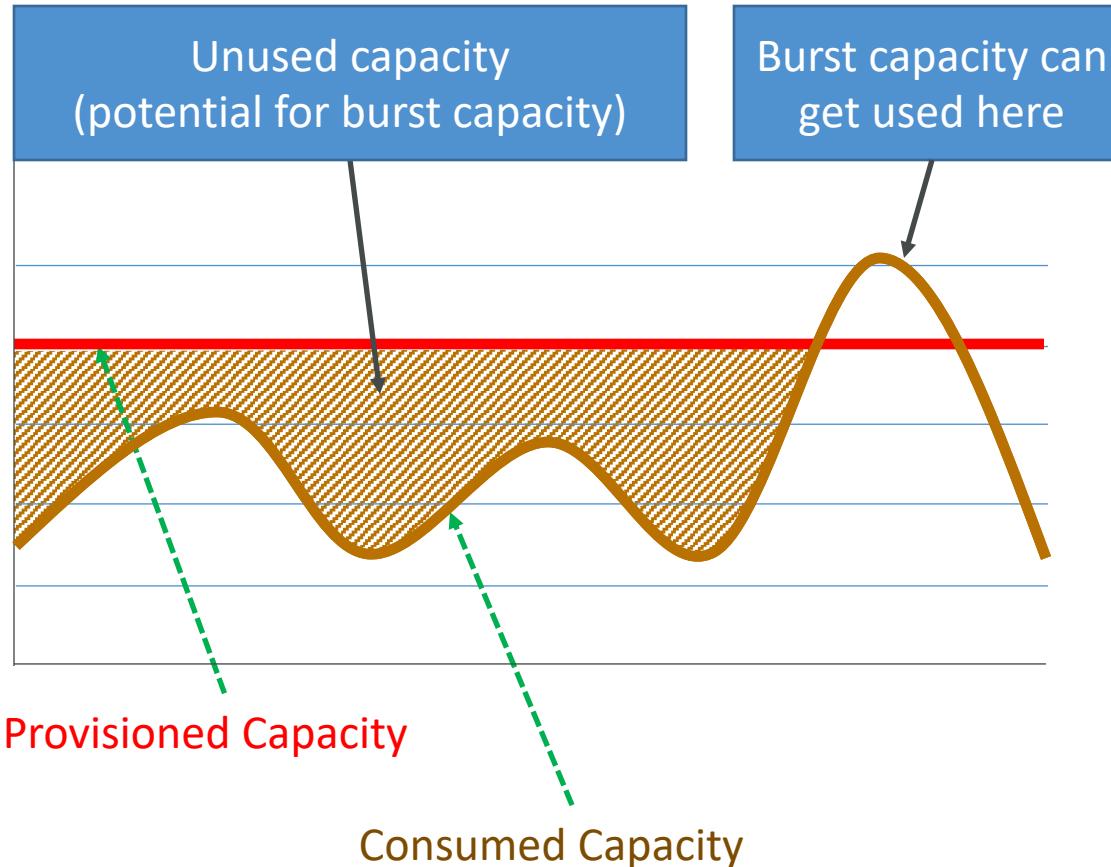
Example 3: Calculating Throughput

A DynamoDB table has provisioned capacity of 10 RCUs and 10WCUs. Calculate the throughput that your application can support:

- Read throughput with strong consistency = $4\text{KB} \times 10 = 40\text{KB/sec}$
- Read throughput (eventual) = $2 (40\text{KB/sec}) = 80\text{KB/sec}$
- Transactional read throughput = $(1/2) \times (40\text{KB/sec}) = 20\text{KB/sec}$
- Write throughput = $1\text{KB} \times 10 = 10\text{KB/sec}$
- Transactional write throughput = $(1/2) \times (10\text{KB/sec}) = 5\text{KB/sec}$

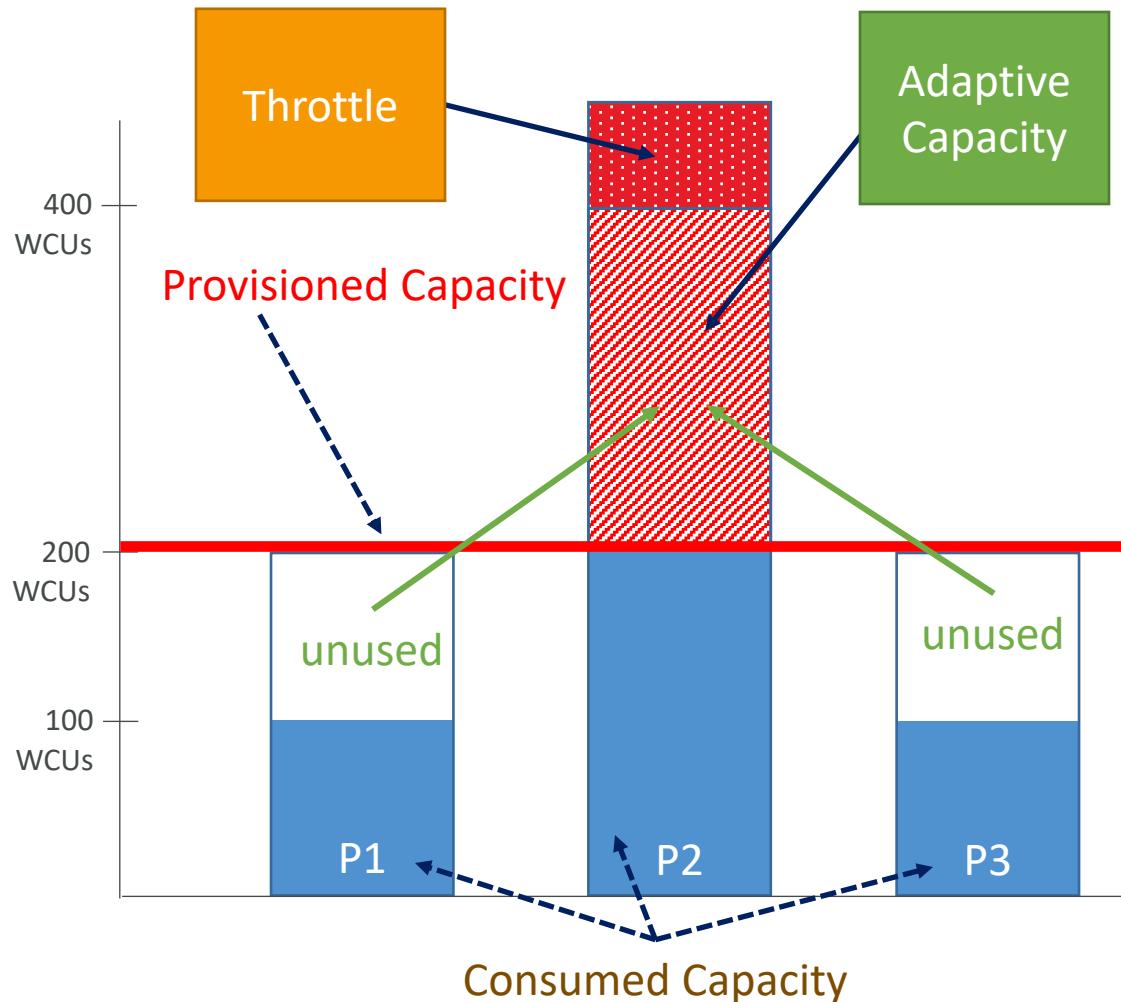


DynamoDB Burst Capacity



- To provide for occasional bursts or spikes
- 5 minutes or 300 seconds of unused read and write capacity
- Can get consumed quickly
- Must not be relied upon

DynamoDB Adaptive Capacity



- Total provisioned capacity = 600 WCUs per sec
- Provisioned capacity per partition = 200 WCUs per sec
- Unused capacity = 200 WCUs per sec
- So the hot partition can consume these unused 200 WCUs per sec above its allocated capacity
- Consumption beyond this results in throttling
- For Non-uniform Workloads
- Works automatically and applied in real time
- No Guarantees

DynamoDB LSI (Local Secondary Index)

- Can define up to 5 LSIs
- Has same partition/hash key attribute as the primary index of the table
- Has different sort/range key than the primary index of the table
- Must have a sort/range key (=composite key)
- Indexed items must be \leq 10 GB
- Can only be created at the time of creating the table and cannot be deleted later
- Can only query single partition (specified by hash key)
- Support eventual / strong / transactional consistency
- Consumes provisioned throughput of the base table
- Can query any table attributes even if the attributes are not projected on to the index (can project all or up to 20 individual attributes)

user_id	game_id	game_ts	result	duration
12broiu45	1234	“2018-03-15T17:43:08”	win	45
12broiu45	3456	“2018-06-20T19:02:32”	lose	33
34oiusd21	4567	“2018-02-11T-04:11:31”	lose	45

Primary Key (user_id + game_id)
LSI Examples

- user_id + game_ts
- user_id + result

DynamoDB GSI (Global Secondary Index)

- Can define up to 20 GSIs (soft limit)
- Can have same or different partition/hash key than the table's primary index
- Can have same or different sort/range key than the table's primary index
- Can omit sort/range key (=simple and composite)
- No size restrictions for indexed items
- Can be created or deleted any time. Can delete only one GSI at a time
- Can query across partitions (over entire table)
- Support only eventual consistency
- Has its own provisioned throughput
- Can only query projected attributes (attributes included in the index)

user_id	game_id	game_ts	result	duration
12broiu45	1234	“2020-03-15T17:43:08”	win	45
12broiu45	3456	“2020-06-20T19:02:32”	lose	33
34oiusd21	4567	“2020-02-11T-04:11:31”	lose	45

Primary Key
(user_id + game_id)

GSI Examples

- user_id
- game_id
- user_id + result
- game_ts + game_id
- game_ts + duration

When to choose which index?

Local Secondary Indexes

- When application needs **same partition key** as the table
- When you need to **avoid additional costs**
- When application needs **strongly consistent index reads**

LSI Examples

- user_id + game_ts
- user_id + result

user_id	game_id	game_ts	result	duration
12broiu45	1234	“2020-03-15T17:43:08”	win	45
12broiu45	3456	“2020-06-20T19:02:32”	lose	33
34oiusd21	4567	“2020-02-11T-04:11:31”	lose	45

Global Secondary Indexes

- When application needs **different or same partition key** as the table
- When application needs **finer throughput control**
- When application only needs **eventually consistent index reads**

GSI Examples

- user_id
- game_id
- **user_id + result**
- game_ts + game_id
- game_ts + duration

DynamoDB Indexes and Throttling

Local Secondary Indexes

- Uses the WCU and RCU of the main table
- No special throttling considerations

Global Secondary Indexes

- If the writes are throttled on the GSI, then the main table will be throttled! (even if the WCU on the main tables are fine)
- Choose your GSI partition key carefully!
- Assign your WCU capacity carefully!

LSI Examples

- user_id + game_ts
- user_id + result

user_id	game_id	game_ts	result	duration
12broiu45	1234	“2020-03-15T17:43:08”	win	45
12broiu45	3456	“2020-06-20T19:02:32”	lose	33
34oiusd21	4567	“2020-02-11T-04:11:31”	lose	45

GSI Examples

- user_id
- game_id
- user_id + result
- game_ts + game_id
- game_ts + duration

Simple design patterns with DynamoDB

- You can model different entity relationships like 1:1, 1:N, N:M
- Store players' game states – 1:1 modeling, 1:N modeling
 - user_id as PK, game_id as SK (1:N modeling)
- Players' gaming history – 1:N modeling
 - user_id as PK, game_ts as SK (1:N modeling)
- Gaming leaderboard – N:M modeling
 - GSI with game_id as PK and score as SK

user_id	game_id	game_ts	result	score	duration
12broiu45	1234	"2020-03-15T17:43:08"	win	98	45
12broiu45	3456	"2020-06-20T19:02:32"	lose	91	33
34oiusd21	4567	"2020-02-11T-04:11:31"	lose	88	45

DynamoDB Write Sharding

- Imagine we have a voting application with two candidates, candidate A and candidate B.
- If we use a partition key of candidate_id, we will run into partitions issues, as we only have two partitions
- Solution: add a suffix (usually random suffix, sometimes calculated suffix)

Partition Key	Sort_Key	Attributes
CandidateID + RandomSuffix	Vote_date	Voter_id
Candidate_A-1	2020-07-18 01.36.45	235343
Candidate_A-2	2020-07-18 01.36.30	232312
Candidate_B-1	2020-07-18 01.36.20	098432
Candidate_B-2	2020-07-18 01.36.15	340983

Error and Exceptions in DynamoDB



HTTP 5XX

(Server-side errors)

service unavailable etc.



HTTP 4XX

(Client-side errors)

authentication failure, missing required parameters etc.

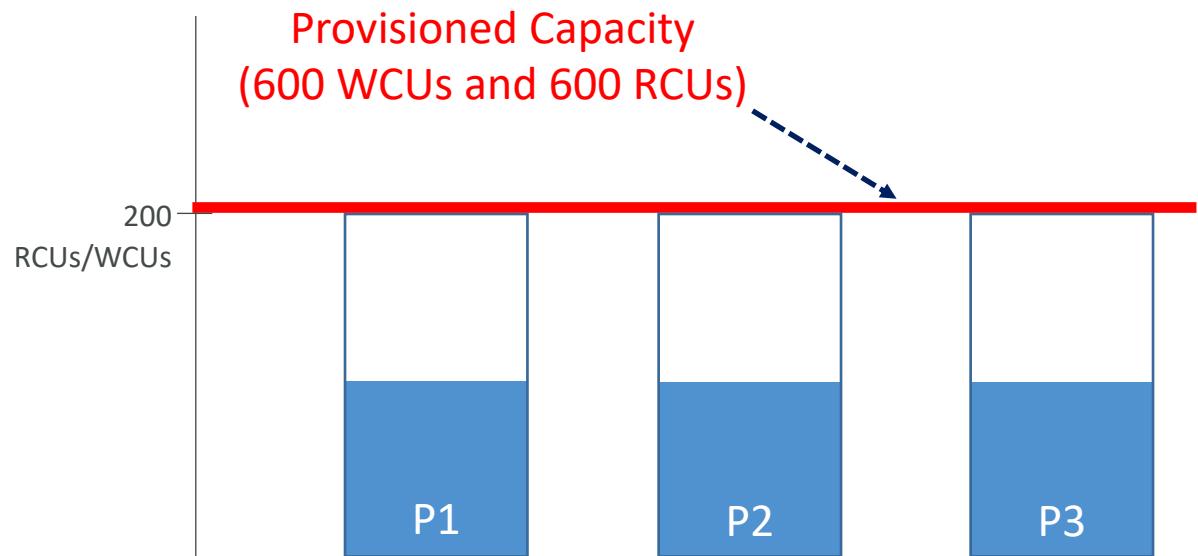
- Common Exceptions

- Access Denied Exception
- Conditional Check Failed Exception
- Item Collection Size Limit Exceeded Exception
- Limit Exceeded Exception
- Resource In Use Exception
- Validation Exception
- Provisioned Throughput Exceeded Exception
 - Error Retries
 - Exponential Backoff

DynamoDB Partitions

- Store DynamoDB table data (physically)
- Each (physical) partition = 10GB SSD volume
- Not to be confused with table's partition/hash key (which is a logical partition)
- One partition can store items with multiple partition keys
- A table can have multiple partitions
- Number of table partitions depend on its size and provisioned capacity
- Managed internally by DynamoDB

- Provisioned capacity is evenly distributed across table partitions
- Partitions once allocated, **cannot be deallocated** (important!)



Calculating DynamoDB Partitions

- 1 partition = 1000 WCUs or 3000 RCUs (Maximum supported throughput per partition)
- 1 partition = 10GB of data
- No. of Partitions = Either the number of partitions based on throughput or the number of partitions based on size, whichever is higher



$$P_T = \text{Round up} [(\text{RCUs} / 3000) + (\text{WCUs} / 1000)]$$

$$P_S = \text{Round up} [(\text{Storage Required in GB} / 10 \text{ GB})]$$

$$P = \max (P_T, P_S)$$

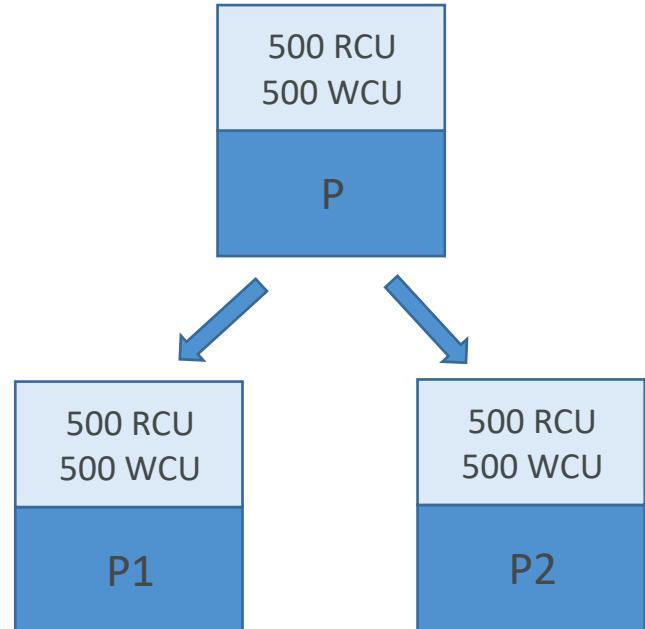
Partition Behavior Example (Scaling up Capacity)

- Provisioned Capacity: 500 RCUs and 500 WCUs
- Storage requirement < 10 GB
- Number of Partitions:

$$\begin{aligned} P_T &= (500 \text{ RCUs}/3000 + 500 \text{ WCUs}/1000) \\ &= 0.67 \Rightarrow \text{rounded up} \Rightarrow 1 \text{ partition} \end{aligned}$$

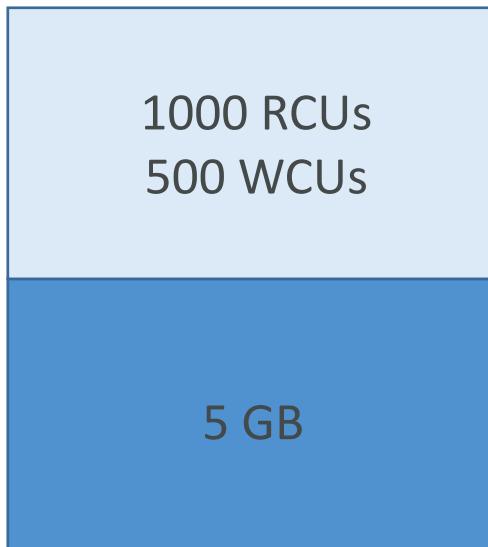
- Say, we scale up the provisioned capacity
- New Capacity: 1000 RCUs and 1000 WCUs

$$\begin{aligned} P_T &= (1000 \text{ RCUs}/3000 + 1000 \text{ WCUs}/1000) \\ &= 1.33 \Rightarrow \text{rounded up} \Rightarrow 2 \text{ partitions} \end{aligned}$$



Partition Behavior Example (Scaling up Storage)

- Provisioned Capacity = 1000 RCU and 500 WCU
- Storage size = 5 GB



$$\begin{aligned}P_T &= \text{Round up} [(1000 / 3000) + (500 / 1000)] \\&= \text{Round up} [0.67]\end{aligned}$$

$$P_T = 1$$

$$P_S = \text{Round up} [(5 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [0.5]$$

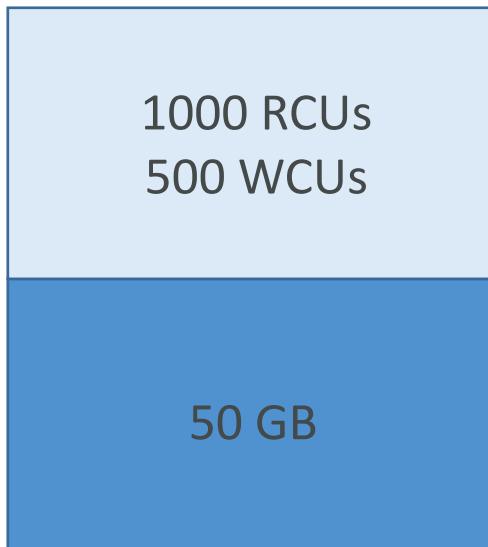
$$P_S = 1$$

$$P = \max (1, 1)$$

$$P = 1$$

Partition Behavior Example (Scaling up Storage)

- Provisioned Capacity = 1000 RCU and 500 WCU
- New Storage size = 50 GB



$$\begin{aligned}P_T &= \text{Round up} [(1000 / 3000) + (500 / 1000)] \\&= \text{Round up} [0.67]\end{aligned}$$

$$P_T = 1$$

$$P_S = \text{Round up} [(50 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [5]$$

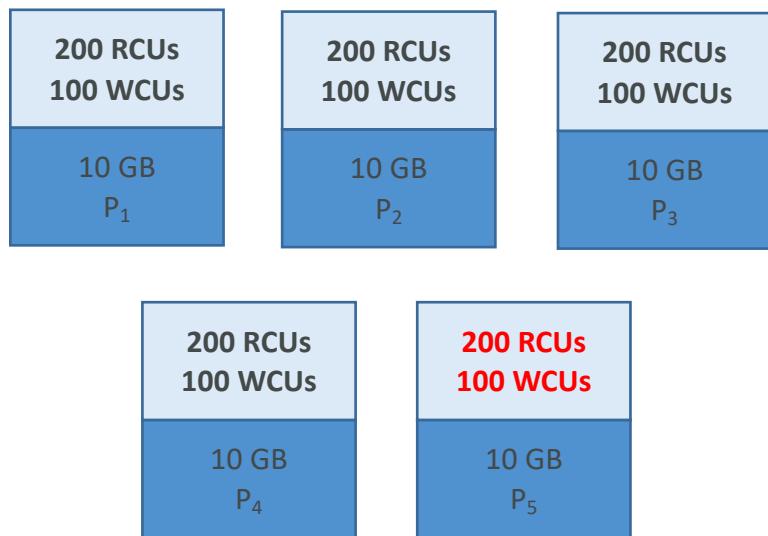
$$P_S = 5$$

$$P = \max (1, 5)$$

$$P = 5$$

Partition Behavior Example (Scaling up Storage)

- Provisioned Capacity = 1000 RCU and 500 WCU
- New Storage size = 50 GB



$$\begin{aligned}P_T &= \text{Round up} [(1000 / 3000) + (500 / 1000)] \\&= \text{Round up} [0.67] \\P_T &= 1\end{aligned}$$

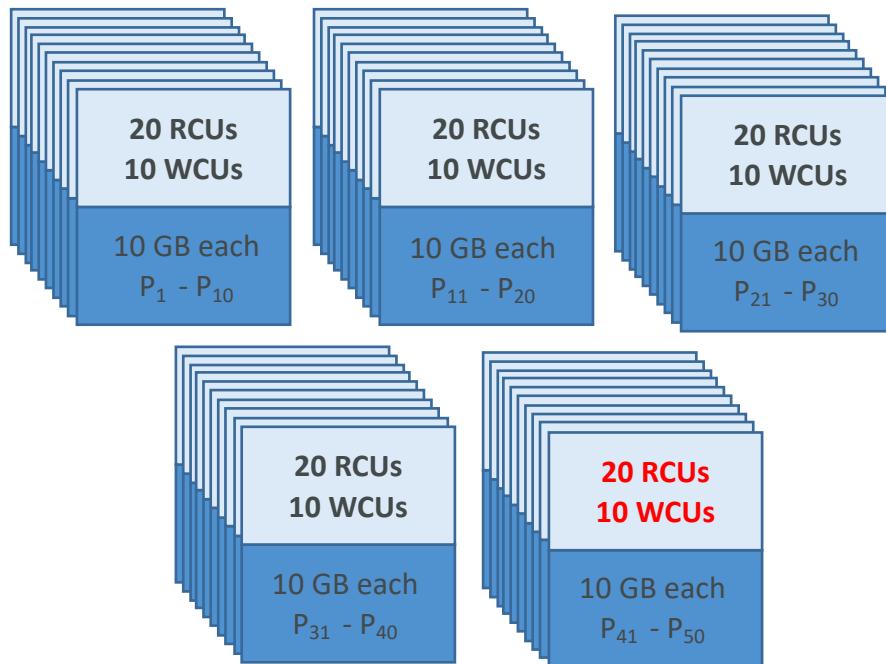
$$\begin{aligned}P_S &= \text{Round up} [(50 \text{ GB} / 10 \text{ GB})] \\P_S &= \text{Round up} [5] \\P_S &= 5\end{aligned}$$

$$\begin{aligned}P &= \max (1, 5) \\P &= 5\end{aligned}$$

Each partition now receives only
1/5th of the provisioned capacity!

Partition Behavior Example (Scaling up Storage)

- Provisioned Capacity = 1000 RCU and 500 WCU
- New Storage size = 500 GB



$$\begin{aligned}P_T &= \text{Round up } [(1000 / 3000) + (500 / 1000)] \\&= \text{Round up } [0.67]\end{aligned}$$

$$P_T = 1$$

$$P_S = \text{Round up } [(500 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up } [50]$$

$$P_S = 50$$

Each partition now receives only
1/50th of the provisioned capacity!

$$P = \max (1, 50)$$

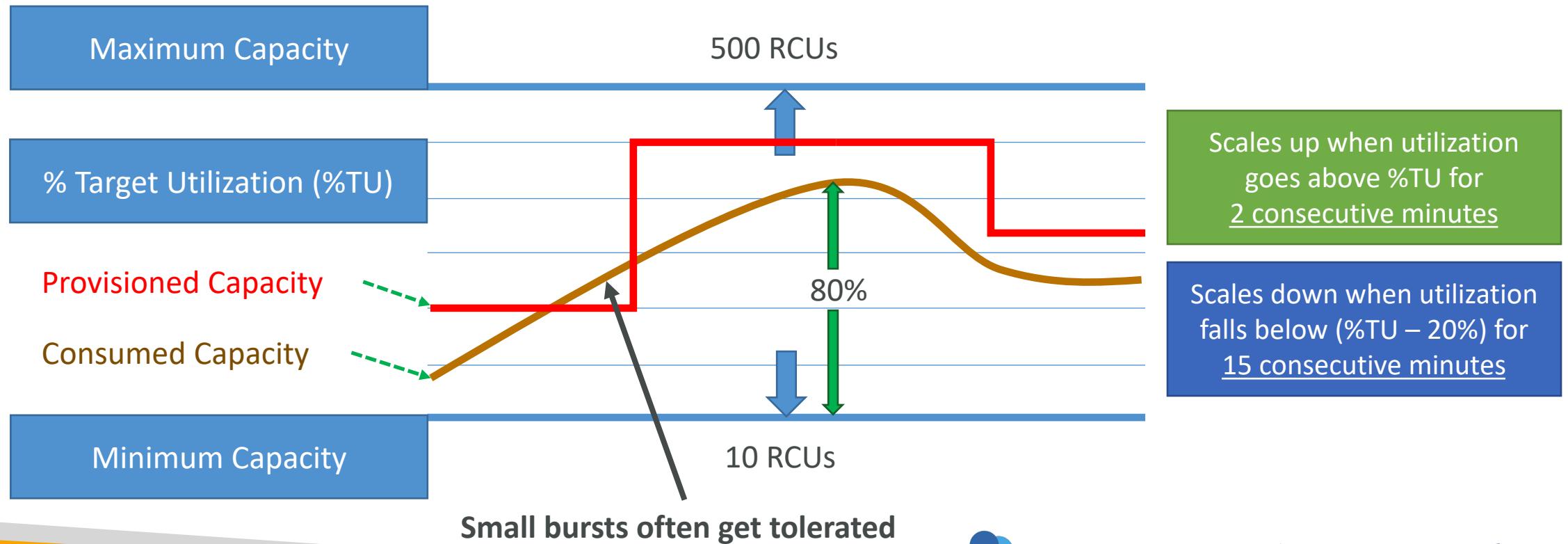
$$P = 50$$

DynamoDB Scaling

- You can manually scale up provisioned capacity as and when needed
- You can only scale down up to 4 times in a day
- Additional one scale down if no scale downs in last 4 hours
- Effectively 9 scale downs per day
- Scaling affects partition behavior
- Any increase in partitions on scale up will not result in decrease on scale down (Important!)
- Partitions once allocated will not get deallocated later

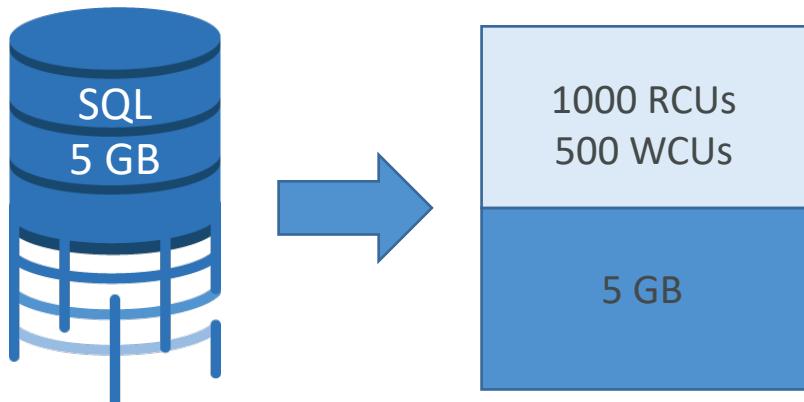
Auto Scaling in DynamoDB

- No additional costs, uses AWS Application Auto Scaling service
- You set the desired target utilization, minimum and maximum provisioned capacity



DynamoDB Scaling and Partition Behavior

- Say, we want to migrate a 5 GB SQL database to DynamoDB
- Provisioned Capacity needed during BAU is 1000 RCU and 500 WCU



$$P_S = \text{Round up} [(5 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [0.5]$$

$$P_S = 1$$

$$P_T = \text{Round up} [(1000 / 3000) + (500 / 1000)]$$

$$= \text{Round up} [0.67]$$

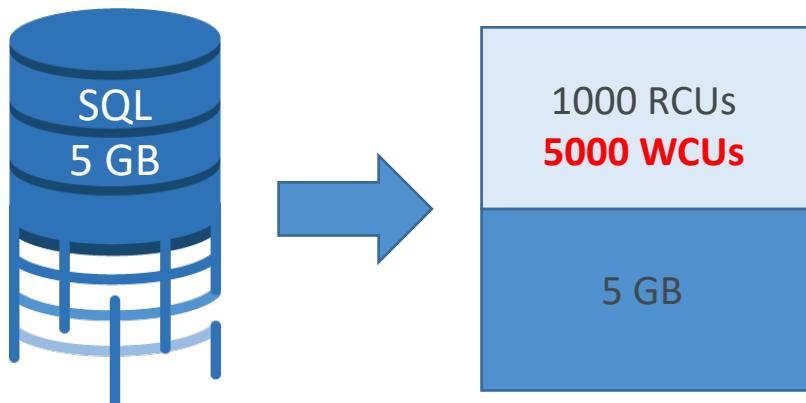
$$P_T = 1$$

$$P = \max (1, 1)$$

$$P = 1$$

DynamoDB Scaling and Partition Behavior

- Say, we want to migrate a 5 GB SQL database to DynamoDB
- Provisioned Capacity needed during BAU is 1000 RCU and 500 WCU
- To speed up the migration, we scale up the write capacity temporarily to 5000 WCU



$$P_S = \text{Round up} [(5 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [0.5]$$

$$P_S = 1$$

$$P_T = \text{Round up} [(1000 / 3000) + (5000 / 1000)]$$

$$= \text{Round up} [5.33]$$

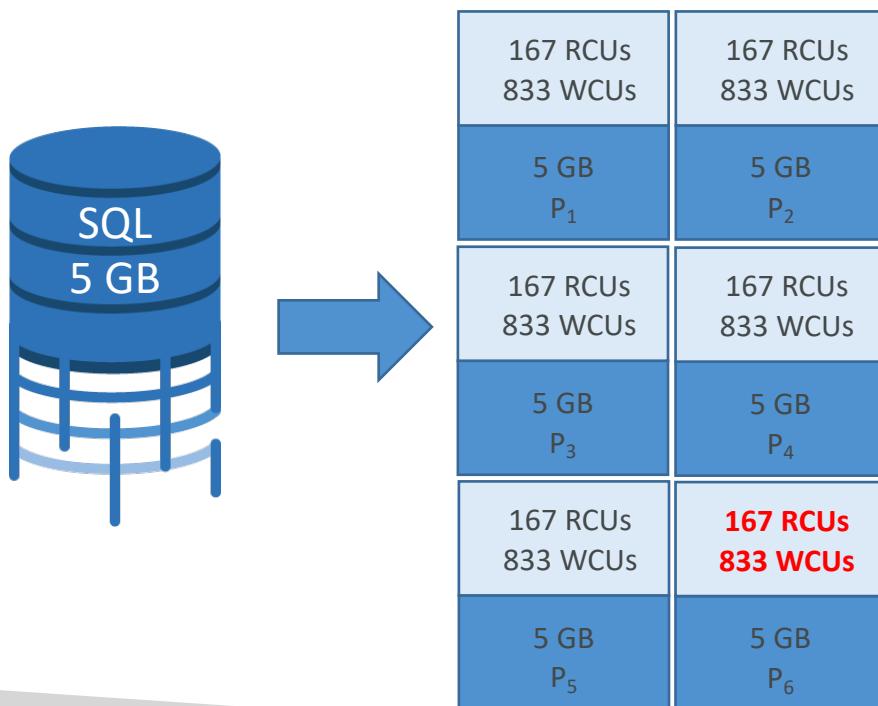
$$P_T = 6$$

$$P = \max (1, 6)$$

$$P = 6$$

DynamoDB Scaling and Partition Behavior

- Say, we want to migrate a 5 GB SQL database to DynamoDB
- Provisioned Capacity needed during BAU is 1000 RCU and 500 WCU
- To speed up the migration, we scale up the write capacity temporarily to 5000 WCUs



$$P_S = \text{Round up} [(5 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [0.5]$$

$$P_S = 1$$

$$\begin{aligned}P_T &= \text{Round up} [(1000 / 3000) + (5000 / 1000)] \\&= \text{Round up} [5.33]\end{aligned}$$

$$P_T = 6$$

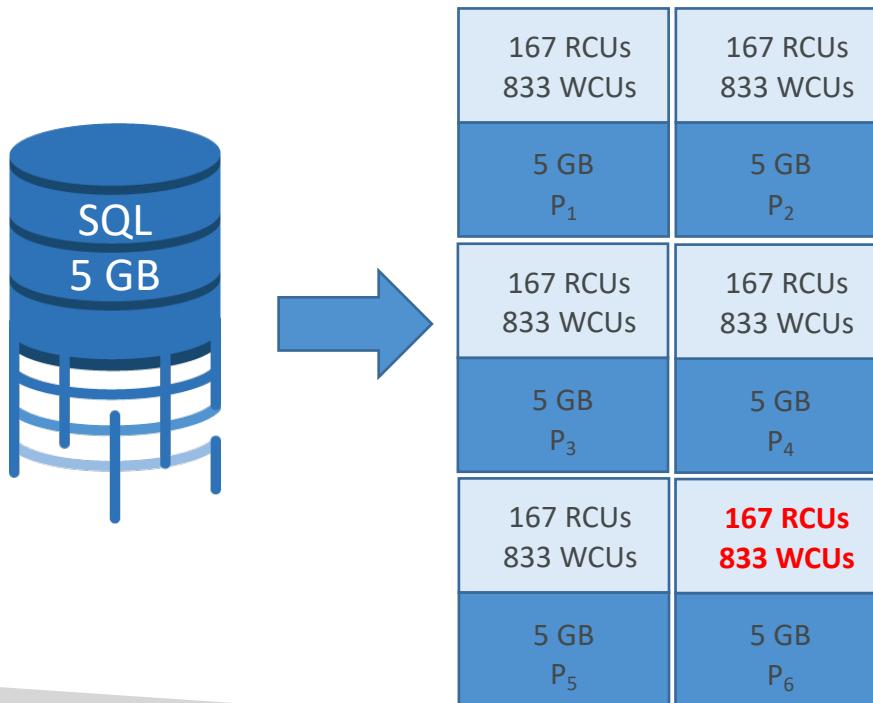
$$P = \max (1, 6)$$

$$P = 6$$

WCUs per partition
increased from 500 to 833!

DynamoDB Scaling and Partition Behavior

- Say, we want to migrate a 5 GB SQL database to DynamoDB
- Provisioned Capacity needed during BAU is 1000 RCU and 500 WCU
- To speed up the migration, we scale up the write capacity temporarily to 5000 WCU
- Post migration, we scale back to 500 WCU



$$P_S = \text{Round up} [(5 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [0.5]$$

$$P_S = 1$$

$$P_T = \text{Round up} [(1000 / 3000) + (5000 - 500 / 1000)] \\ = \text{Round up} [0.67]$$

$$P_T = 1$$

$$P_{\text{PREVIOUS}} = \max (6, 1)$$

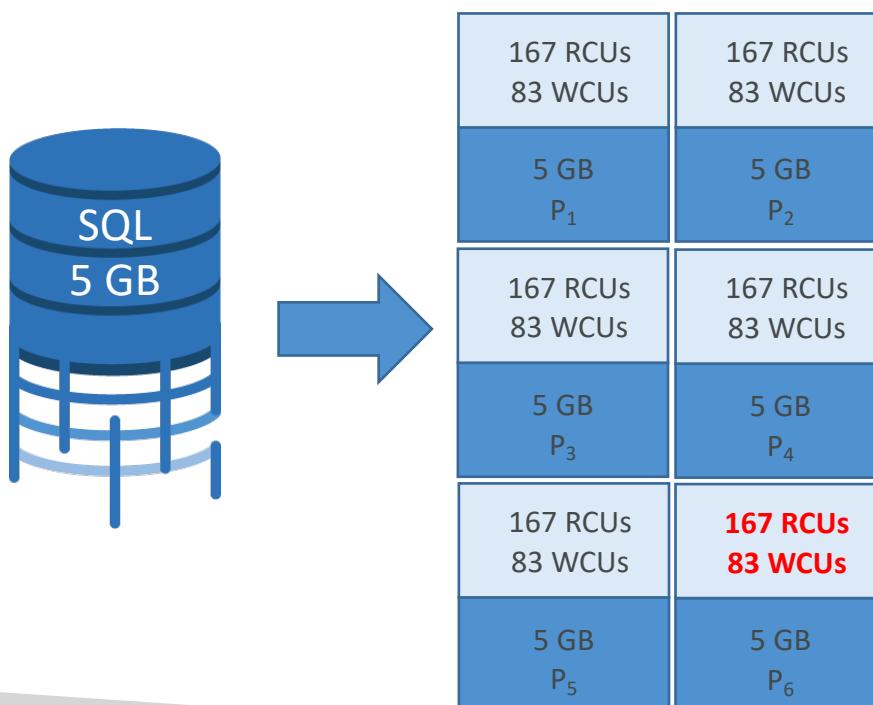
$$P_{\text{PREVIOUS}} = 6$$

$$P_{\text{NEW}} = \max (1, 1)$$

$$P_{\text{NEW}} = 1$$

DynamoDB Scaling and Partition Behavior

- Say, we want to migrate a 5 GB SQL database to DynamoDB
- Provisioned Capacity needed during BAU is 1000 RCU and 500 WCU
- To speed up the migration, we scale up the write capacity temporarily to 5000 WCU
- Post migration, we scale back to 500 WCU



$$P_S = \text{Round up} [(5 \text{ GB} / 10 \text{ GB})]$$

$$P_S = \text{Round up} [0.5]$$

$$P_S = 1$$

$$\begin{aligned} P_T &= \text{Round up} [(1000 / 3000) + (500 / 1000)] \\ &= \text{Round up} [0.67] \end{aligned}$$

$$P_T = 1$$

$$P = \max (P_T, P_S, P_{\text{PREVIOUS}})$$

$$P = \max (1, 1, 6)$$

$$P = 6$$

WCUs per partition
dropped from 833 to 83!

DynamoDB Best Practices

Efficient Key Design

- Partition key should have many unique values
- Distribute reads / writes uniformly across partitions to **avoid hot partitions**
- Store hot and cold data in separate tables
- Consider all possible query patterns to eliminate use of scans and filters
- Choose sort key depending on your application's needs
- Use indexes based on when your application's query patterns
- Use primary key or LSIs when strong consistency is desired
- Use GSIs for finer control over throughput or when your application needs to query using a different partition key
- Use shorter (yet intuitive!) attribute names



DynamoDB Best Practices (contd.)

Storing Large Item Attributes

- Use compression (GZIP)
- Use S3 to store large attributes
- Split large attributes across multiple items

Reading

- Avoid scans and filters
- Use eventual consistency for reads

LSIs

- Use LSIs Sparingly
- Project fewer attributes
- Watch for expanding item collections (10 GB size limit!)

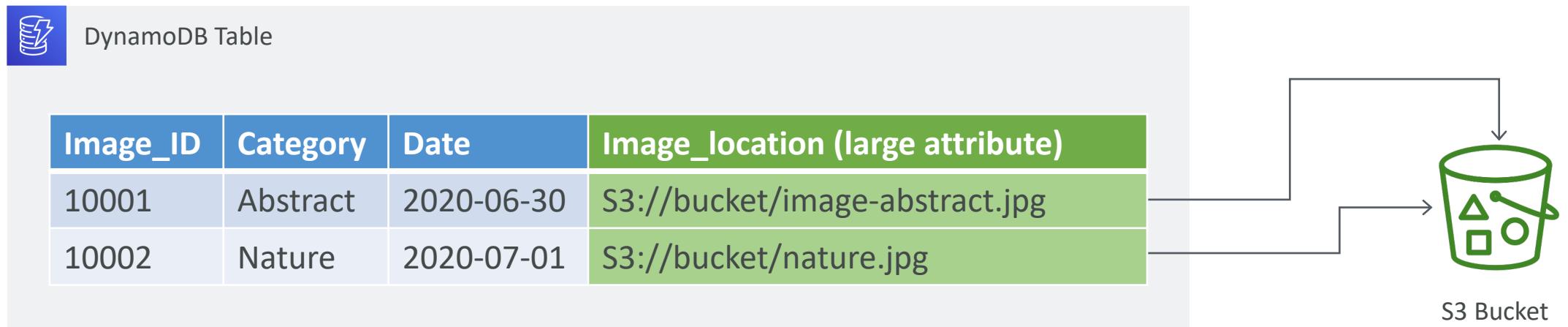
GSIs

- Project fewer attributes
- Can be used for eventually consistent read replicas

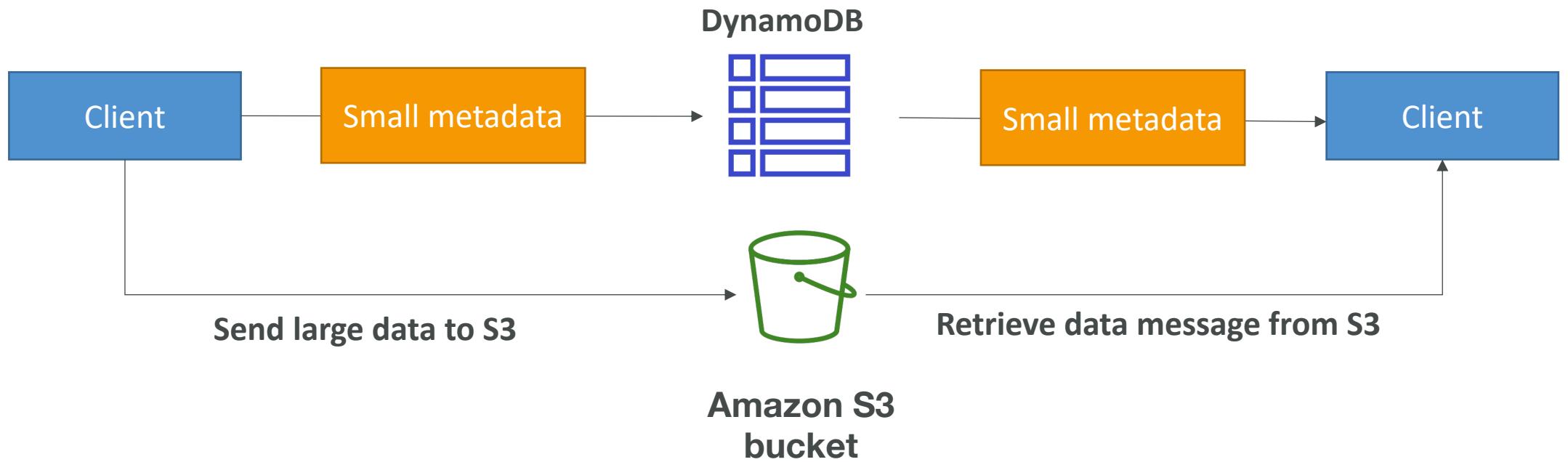


DynamoDB – Storing larger items

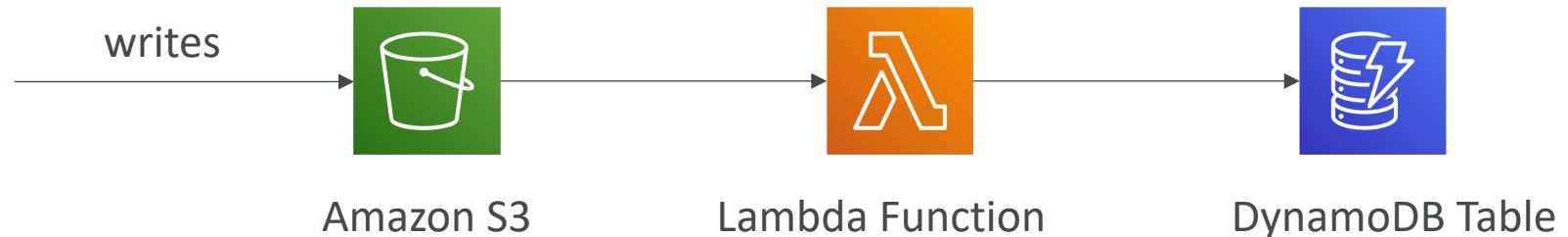
- DynamoDB supports item sizes up to 400 KB each
- This includes attribute name and attribute value (=entire JSON object)
- Options for storing larger items
 - Compress large attribute values, OR
 - Store large attribute values in S3 and store the corresponding S3 path in DynamoDB



DynamoDB - Large Objects Pattern



DynamoDB – Indexing S3 objects metadata

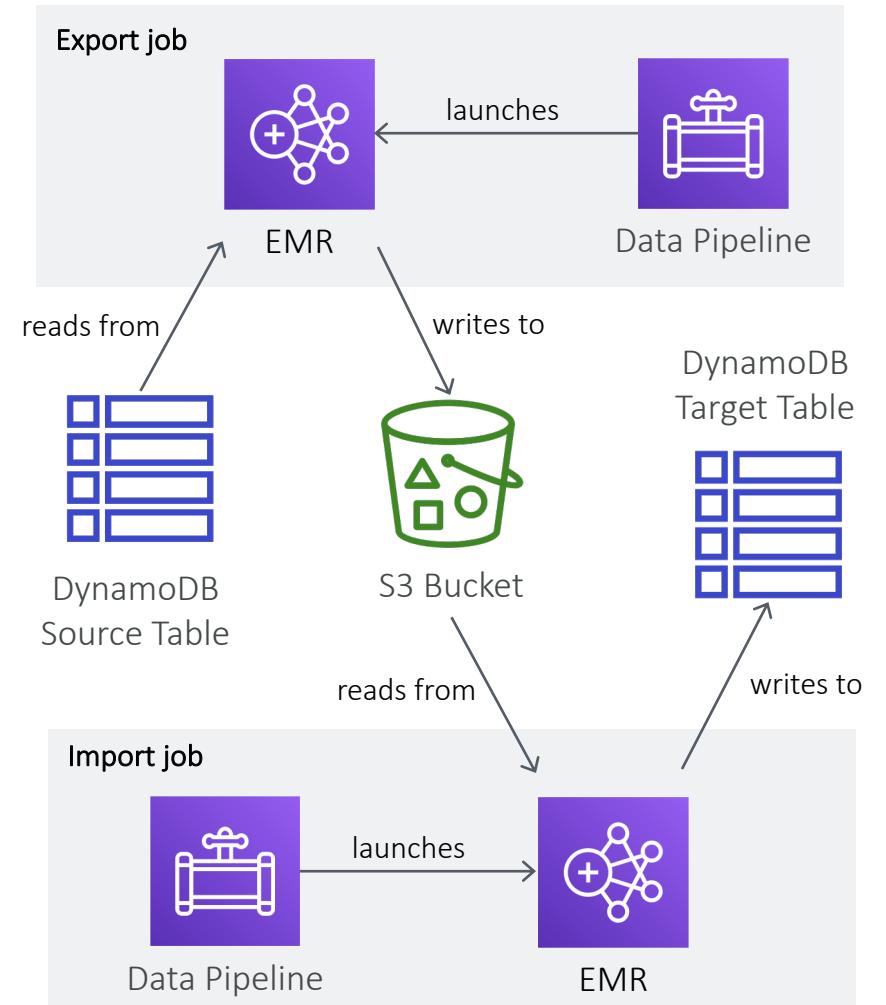


API for object metadata

- Search by date
- Total storage used by a customer
- List of all objects with certain attributes
- Find all objects uploaded within a date range

DynamoDB Operations

- **Table Cleanup:**
 - Option 1: Scan + Delete => very slow, expensive, consumes RCU & WCU
 - Option 2: Drop Table + Recreate table => fast, cheaper, efficient
- **Copying a DynamoDB Table:**
 - Option 1: Use AWS DataPipeline (uses EMR)
 - Option 2: Create a backup and restore the backup into a new table name (can take some time)
 - Option 3: Scan + Write => write own code



DynamoDB Accelerator (DAX)

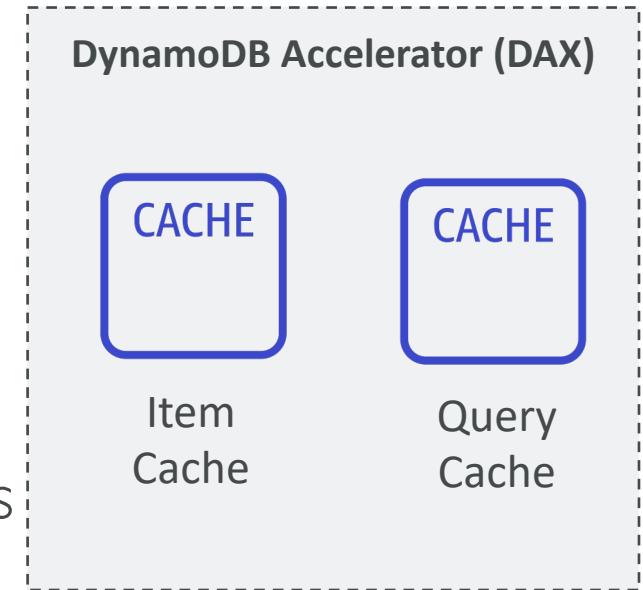
- In-Memory Caching, microsecond latency
- Sits between DynamoDB and Client Application (acts a proxy)
- Saves costs due to reduced read load on DynamoDB
- Helps prevent hot partitions
- Minimal code changes required to add DAX to your existing DynamoDB app
- Supports only eventual consistency (strong consistency requests pass-through to DynamoDB)
- Not for write-heavy applications
- Runs inside the VPC
- Multi AZ (3 nodes minimum recommended for production)
- Secure (Encryption at rest with KMS, VPC, IAM, CloudTrail...)



DAX

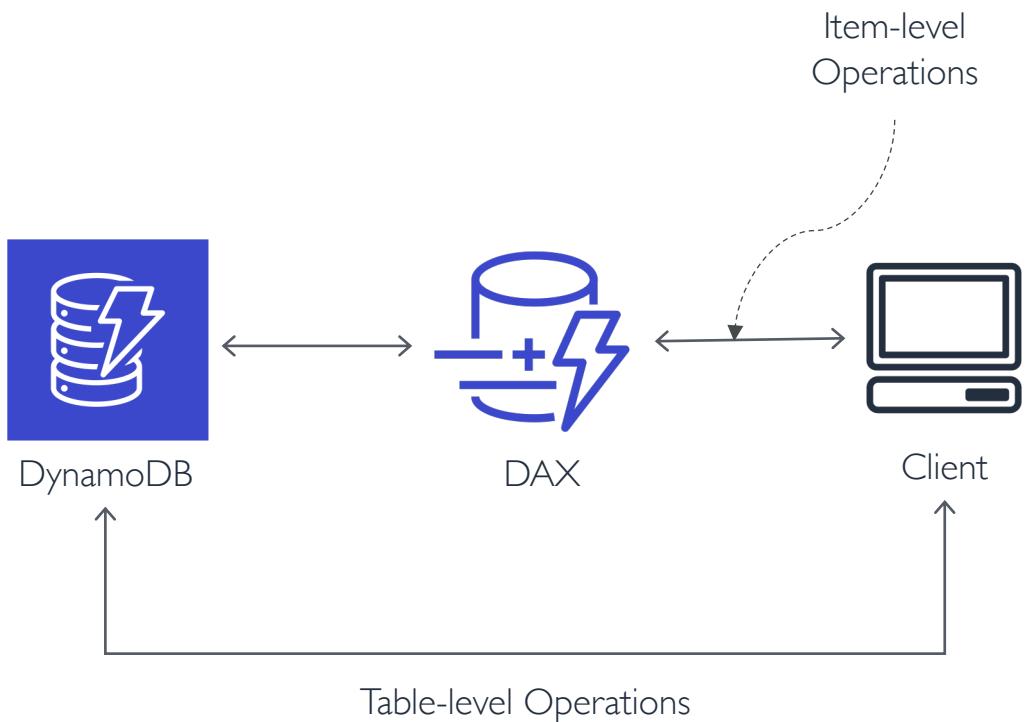
DAX architecture

- DAX has two types of caches (internally)
 - Item Cache
 - Query Cache
- Item cache stores results of index reads (=GetItem and BatchGetItem)
 - Default TTL of 5 min (specified while creating DAX cluster)
 - When cache becomes full, older and less popular items get removed
- Query cache stores results of Query and Scan operations
 - Default TTL of 5 min
 - Updates to the Item cache or to the underlying DynamoDB table do not invalidate the query cache. So, TTL value of the query cache should be chosen accordingly.



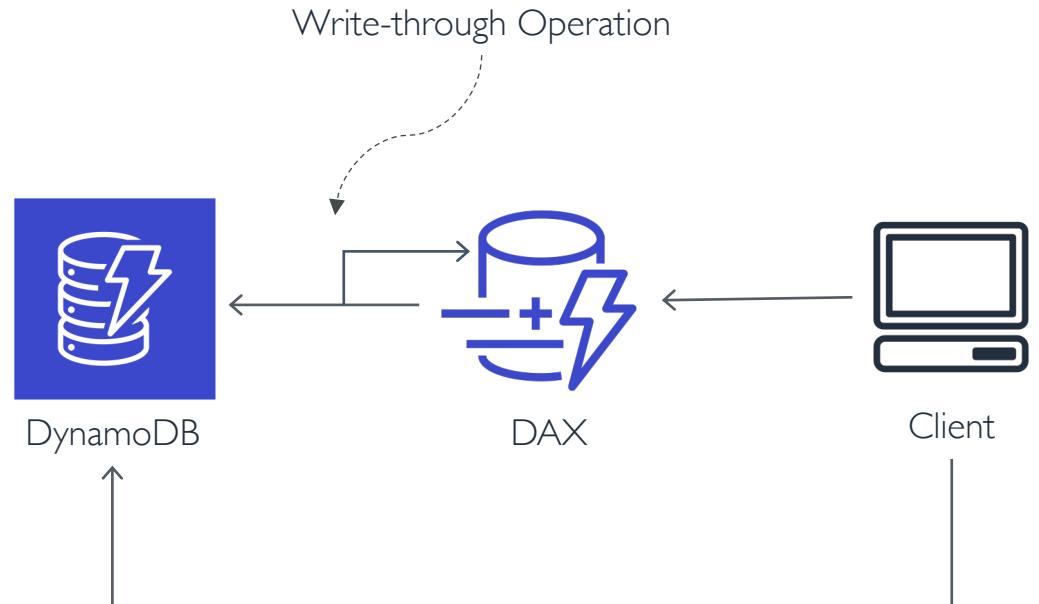
DAX Operations

- Only for item level operations
- Table level operations must be sent directly to DynamoDB



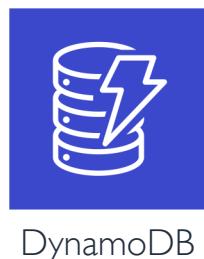
DAX Operations

- Only for item level operations
- Table level operations must be sent directly to DynamoDB
- Write Operations use **write-through approach**
- Data is first written to DynamoDB and then to DAX, and write operation is considered as successful only if both writes are successful
- You can use write-around approach to bypass DAX, e.g. for writing large amount of data, you can write directly to DynamoDB (Item cache goes out of sync)

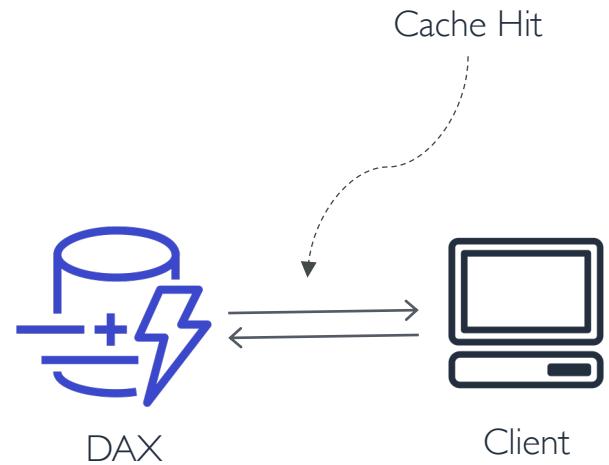


DAX Operations

- Only for item level operations
- Table level operations must be sent directly to DynamoDB
- Write Operations use **write-through approach**
- Data is first written to DynamoDB and then to DAX, and write operation is considered as successful only if both writes are successful
- You can use write-around approach to bypass DAX, e.g. for writing large amount of data, you can write directly to DynamoDB (Item cache goes out of sync)
- For reads, if DAX has the data (=Cache hit), it's simply returned without going through DynamoDB

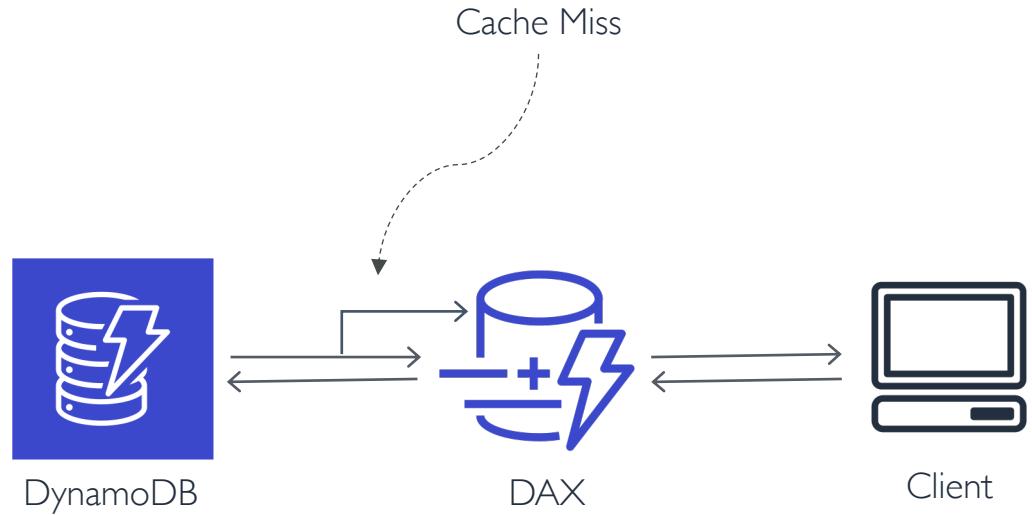


DynamoDB



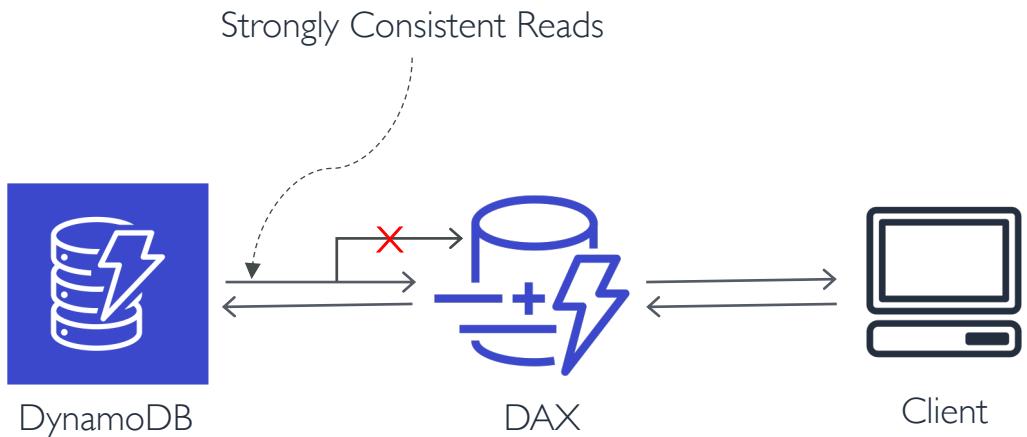
DAX Operations

- Only for item level operations
- Table level operations must be sent directly to DynamoDB
- Write Operations use **write-through approach**
- Data is first written to DynamoDB and then to DAX, and write operation is considered as successful only if both writes are successful
- You can use write-around approach to bypass DAX, e.g. for writing large amount of data, you can write directly to DynamoDB (Item cache goes out of sync)
- For reads, if DAX has the data (=Cache hit), it's simply returned without going through DynamoDB
- If DAX doesn't have the data (=Cache miss), it's returned from DynamoDB and updated in DAX on the master node

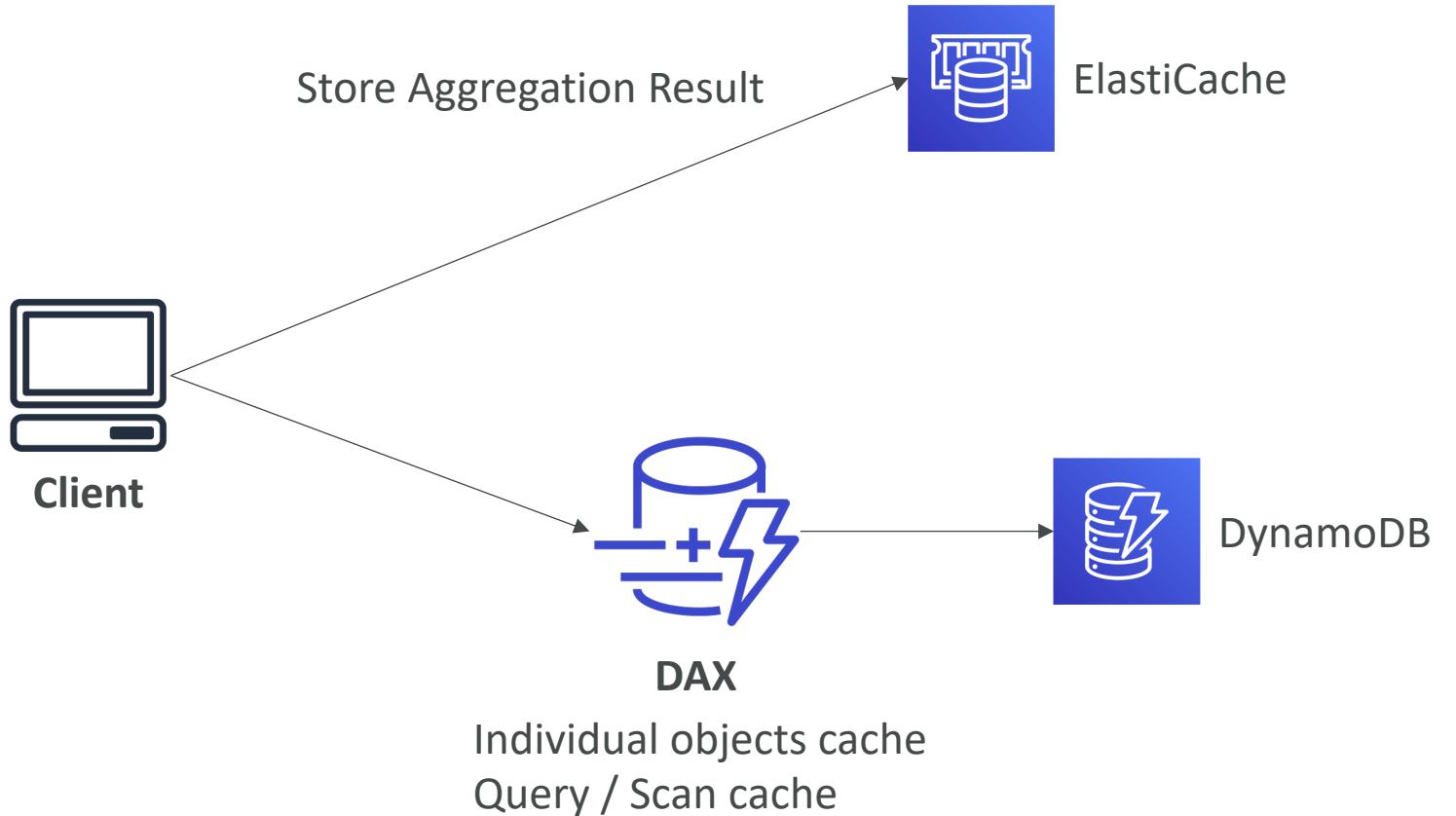


DAX Operations

- Only for item level operations
- Table level operations must be sent directly to DynamoDB
- Write Operations use **write-through approach**
- Data is first written to DynamoDB and then to DAX, and write operation is considered as successful only if both writes are successful
- You can use write-around approach to bypass DAX, e.g. for writing large amount of data, you can write directly to DynamoDB (Item cache goes out of sync)
- For reads, if DAX has the data (=Cache hit), it's simply returned without going through DynamoDB
- If DAX doesn't have the data (=Cache miss), it's returned from DynamoDB and updated in DAX on the master node
- Strongly consistent reads are served directly from DynamoDB and will not be updated in DAX.



DynamoDB – DAX vs ElastiCache



Implementing DAX

- To implement DAX, we create a DAX Cluster
- DAX Cluster consists of one or more nodes (up to 10 nodes per cluster)
- Each node is an instance of DAX
- One node is the master node or primary node
- Remaining nodes act as read replicas
- DAX internally handles load balancing between these nodes
- 3 nodes minimum recommended for production



DAX

Implementing DAX



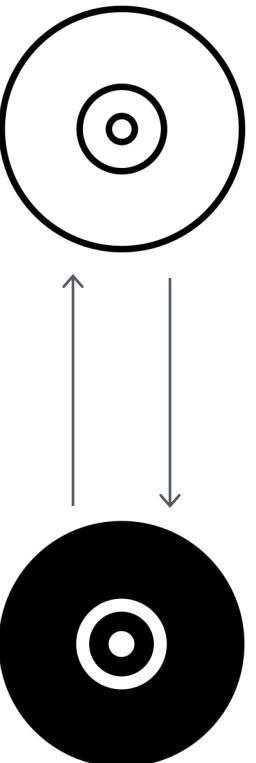
Demo



DataCumulus | RIZMAXed

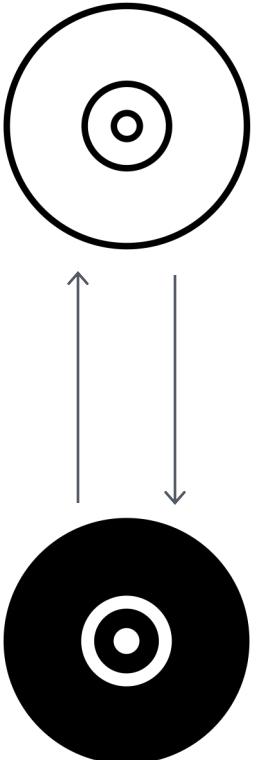
Backup and Restore in DynamoDB

- Automatically encrypted, cataloged and easily discoverable
- Highly Scalable - create or retain as many backups for tables of any size
- Backup operations complete in seconds
- Backups are consistent within seconds across thousands of partitions
- No provisioned capacity consumption
- Does not affect table performance or availability
- Backups are preserved regardless of table deletion



Backup and Restore in DynamoDB

- Can backup within the same AWS region as the table
- Restores can be within same region or cross region
- Integrated with AWS Backup service (can create periodic backup plans)
- Periodic backups can be scheduled using Lambda and CloudWatch triggers
- Cannot overwrite an existing table during restore, restores can be done only to a new table (=new name)
- To retain the original table name, delete the existing table before running restore
- You can use IAM policies for access control



Backup and Restore in DynamoDB

- Restored table gets the same provisioned RCUWs/WCUs as the source table, as recorded at the time of backup
- PITR RPO = 5 minutes approx.
- PITR RTO can be longer as restore operation creates a new table

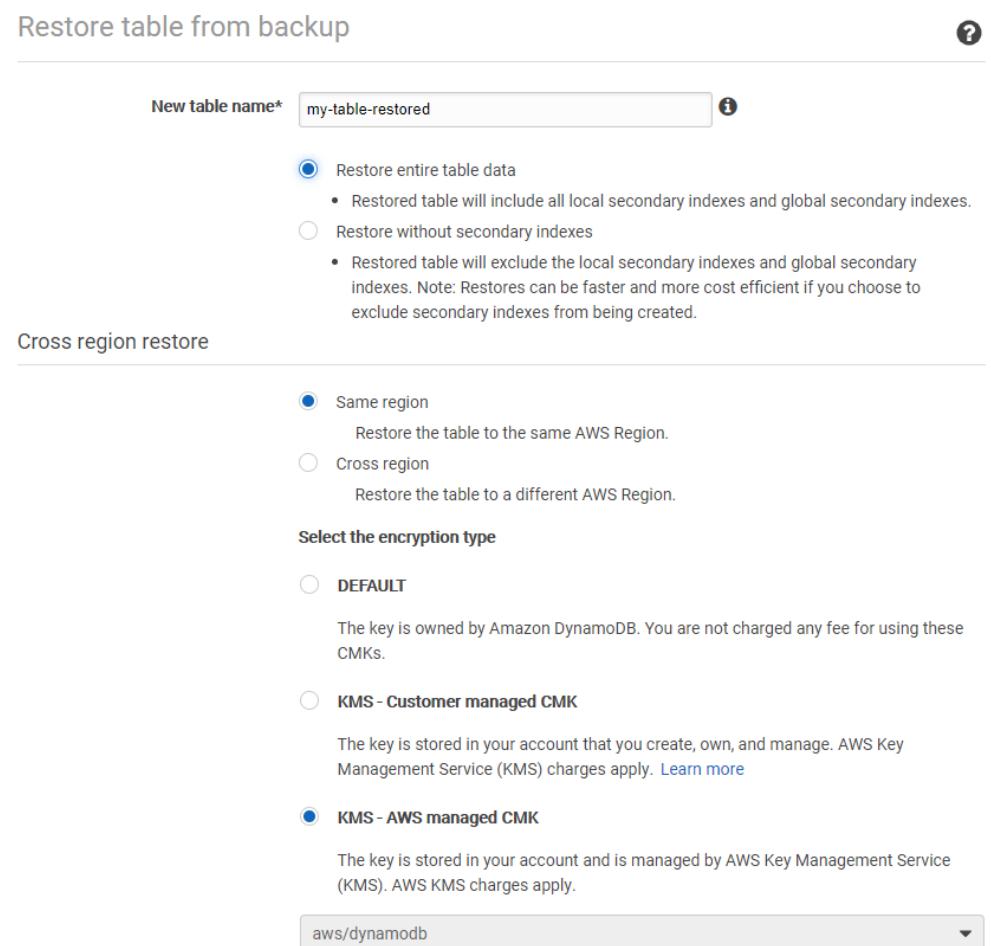
Two backup options

On-Demand Backup
and Restore

Continuous Backups
with PITR

Backup and Restore in DynamoDB

- What gets restored:
 - Table data
 - GSIs and LSIs (optional, you can choose)
 - Encryption settings (you can change)
 - Provisioned RCUs / WCUs (with values at the time when backup was created)
 - Billing mode (with value at the time when backup was created)
- What you must manually set up on the restored table:
 - Auto scaling policies, IAM policies
 - CloudWatch metrics and alarms
 - Stream and TTL settings
 - Tags



On-Demand Backup and Restore



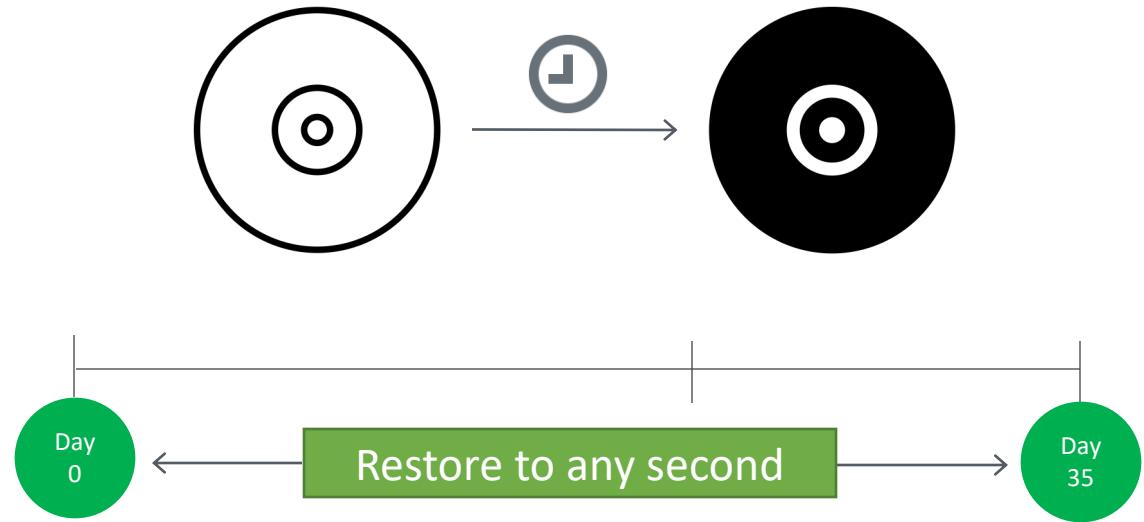
Demo



DataCumulus | RIZMAXed

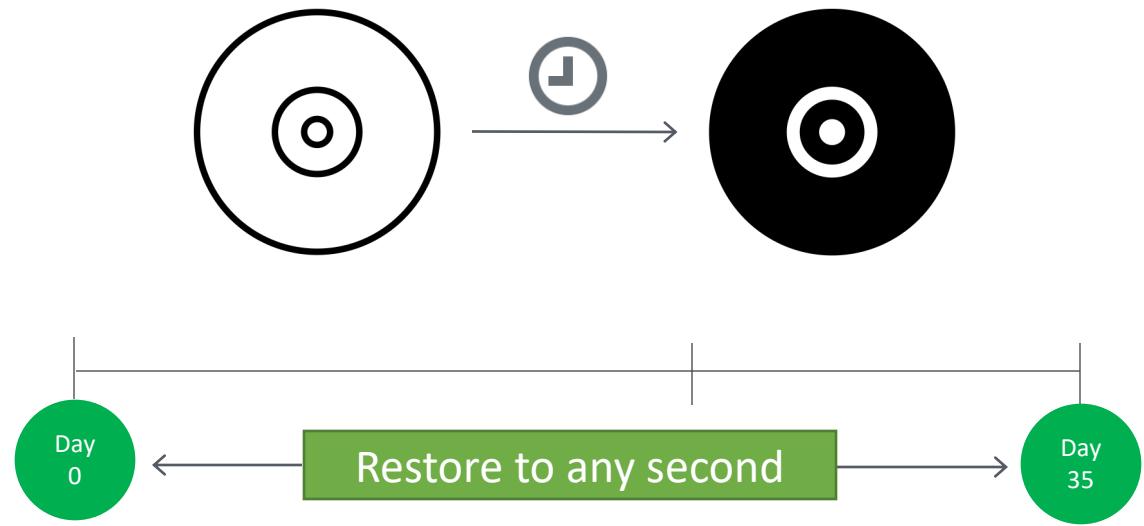
Continuous Backups with PITR

- Restore table data to any second in the last 35 days!
- Priced per GB based on the table size
- If you disable PITR and re-enable it, the 35 days clock gets reset
- Works with unencrypted, encrypted tables as well as global tables
- Can be enabled on each local replica of a global table



Continuous Backups with PITR

- If you restore a table which is part of global tables, the restored table will be an independent table (won't be a global table anymore!)
- Always restores data to a new table
- What cannot be restored
 - Stream settings
 - TTL options
 - Autoscaling config
 - PITR settings
 - Alarms and tags
- All PITR API calls get logged in CloudTrail



Continuous Backups with PITR



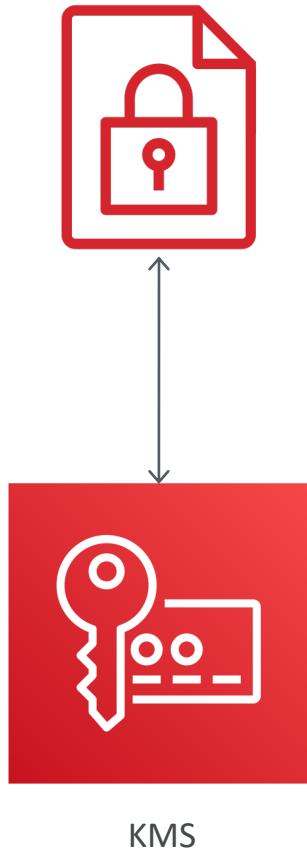
Demo



DataCumulus | RIZMAXed

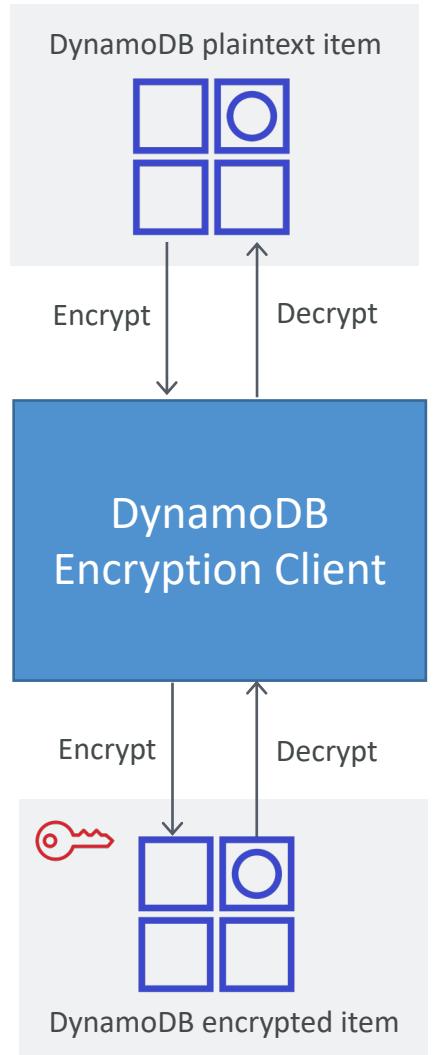
DynamoDB Encryption

- Server-side Encryption at Rest
 - Enabled by default
 - Uses KMS
 - 256-bit AES Encryption
 - Can use AWS owned CMK, AWS managed CMK, or customer managed CMK
 - Encrypts primary key, secondary indexes, streams, global tables, backups and DAX clusters
- Encryption in transit
 - Use VPC endpoints for applications running in a VPC
 - Use TLS endpoints for encrypting data in transit



DynamoDB Encryption Client

- For client-side encryption
- Added protection with encryption in-transit
- Results in end-to-end encryption
- Doesn't encrypt the entire table
- Encrypts the attribute values, but not the attribute names
- Doesn't encrypt values of the primary key attributes
- You can selectively encrypt other attribute values
- You can encrypt selected items in a table, or selected attribute values in some or all items

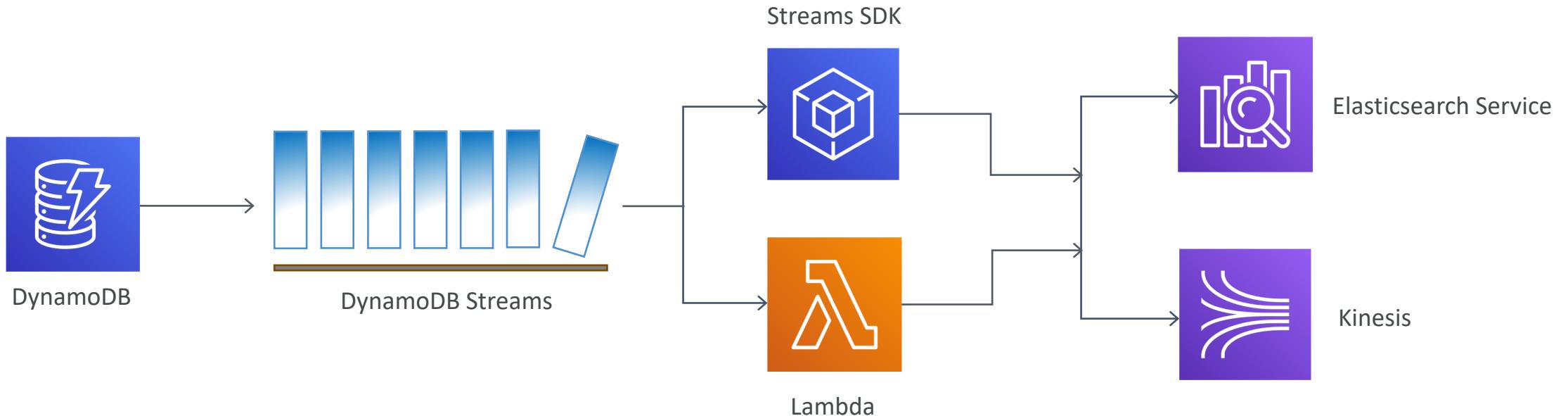


DynamoDB Streams

- 24 Hours time-ordered log of all table-write activity
- React to changes to DynamoDB tables in real time
- Can be read by AWS Lambda, EC2, ES, Kinesis...

Use cases

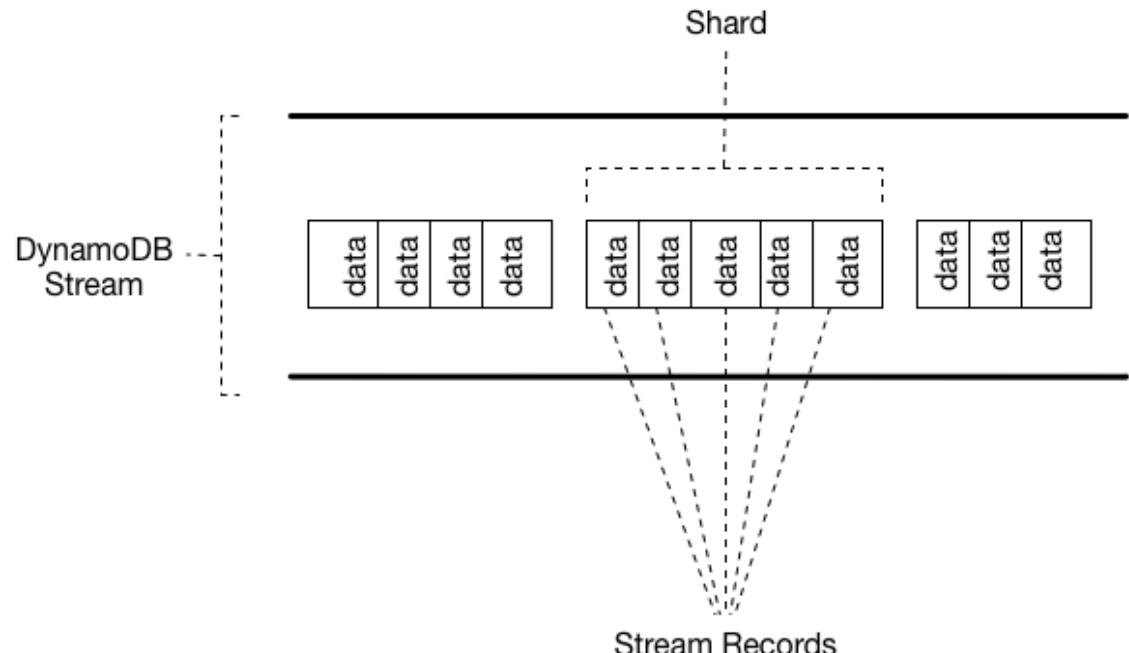
- Replication
- Archival
- Notifications
- Log processing



DynamoDB Streams (Contd.)

- DynamoDB Streams are organized into shards
- Records are not retroactively populated in a stream after enabling it
- Simply enable streams from DynamoDB console
- Four supported views:

Keys only	captures only the key attributes of the changed item
New image	captures the entire item after changes
Old image	captures the entire item before changes
New and old images	captures the entire item before and after changes



<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html>

Time to Live (TTL)

- Allows you to tell DynamoDB when to delete an item from the table
- Simply designate an item attribute as a TTL attribute
- TTL attribute should contain the expiry timestamp for the item (EPOCH or UNIX timestamp)
- Items get marked for deletion on expiry
- Expired items get removed from the table and indexes automatically within about 48 hrs
- Expired items can show up in the API responses until they get deleted
- Application should use filter operations to exclude items marked for deletion
- Deleted items appear in DynamoDB streams (if streams are enabled)

user_id	game_id	game_ts	result	duration	expires	TTL attribute
12broiu45	1234	"2020-03-15T17:43:08"	win	45	1585094400	
12broiu45	3456	"2020-03-20T19:02:32"	lose	33	1593475200	
34oiusd21	4567	"2020-06-11T-04:11:31"	lose	45	2229999999	

Time to Live (TTL)



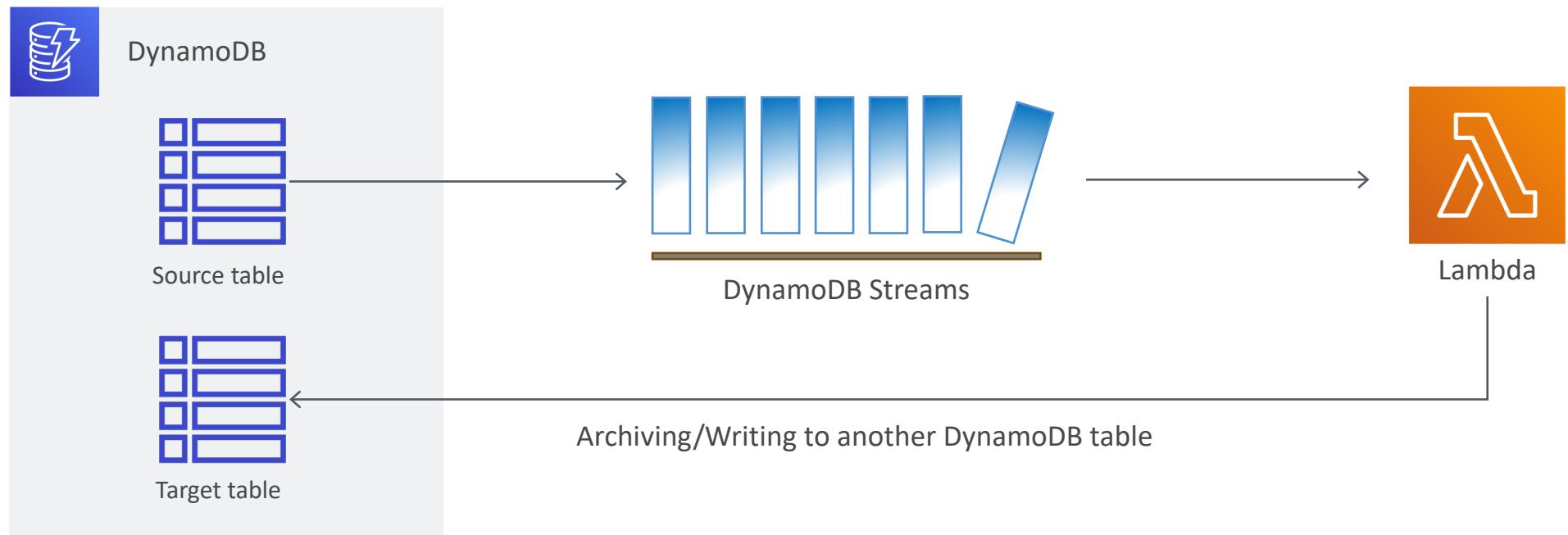
Demo



DataCumulus | RIZMAXed

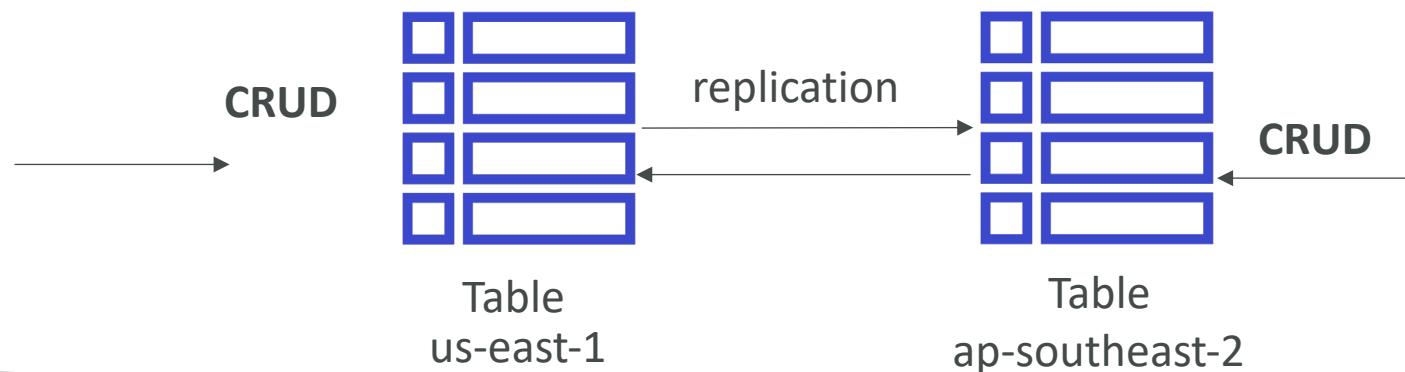
TTL Use Cases

- Data archival to another table (using DynamoDB streams)
- Separating hot and cold data in time-series data (using DynamoDB streams)



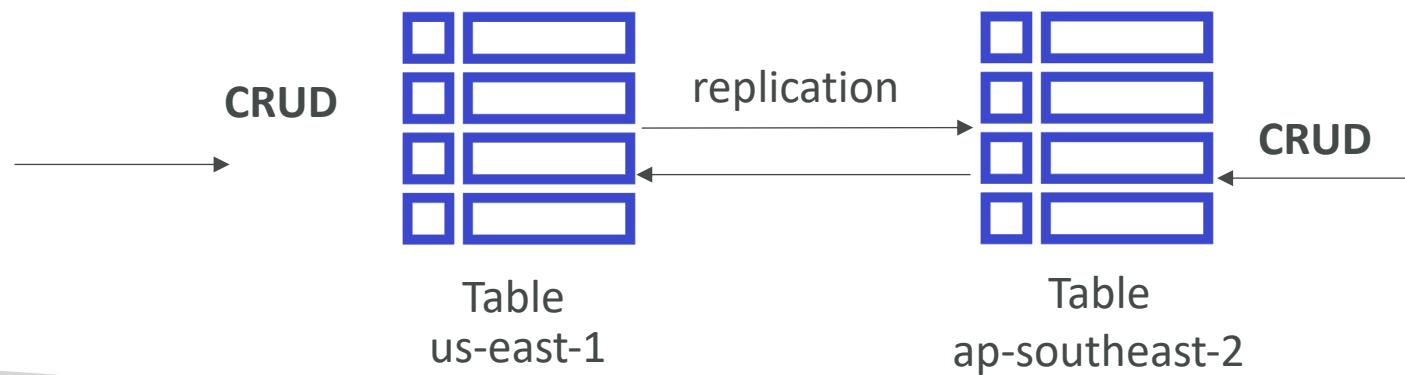
DynamoDB Global Tables

- Automatic, Multi-Master, Active-Active, Cross-region replication
- Useful for low latency, DR purposes
- Near real-time replication (< 1 second replication lag)
- Eventual consistency for cross-region reads
- Strong consistency for same region reads
- “Last Writer Wins” approach for conflict resolution
- Transactions are ACID-compliant only in the region where write occurs originally

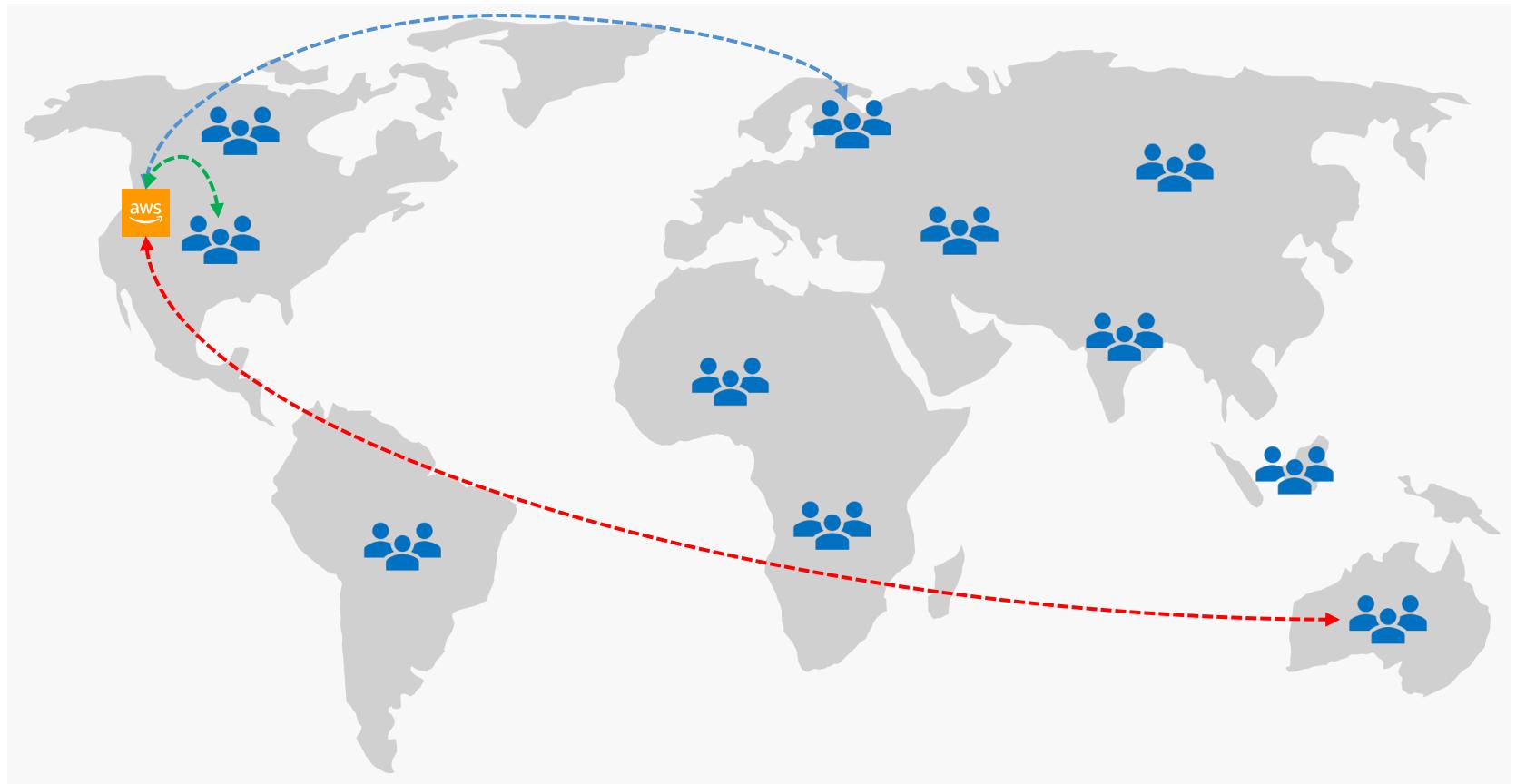


DynamoDB Global Tables

- To enable global tables for a table, the table must be empty across regions
- Only one replica per region
- Must enable DynamoDB Streams with New and Old Images
- Must have the same table name and primary keys across regions
- Recommended to use identical settings for table and indexes across regions



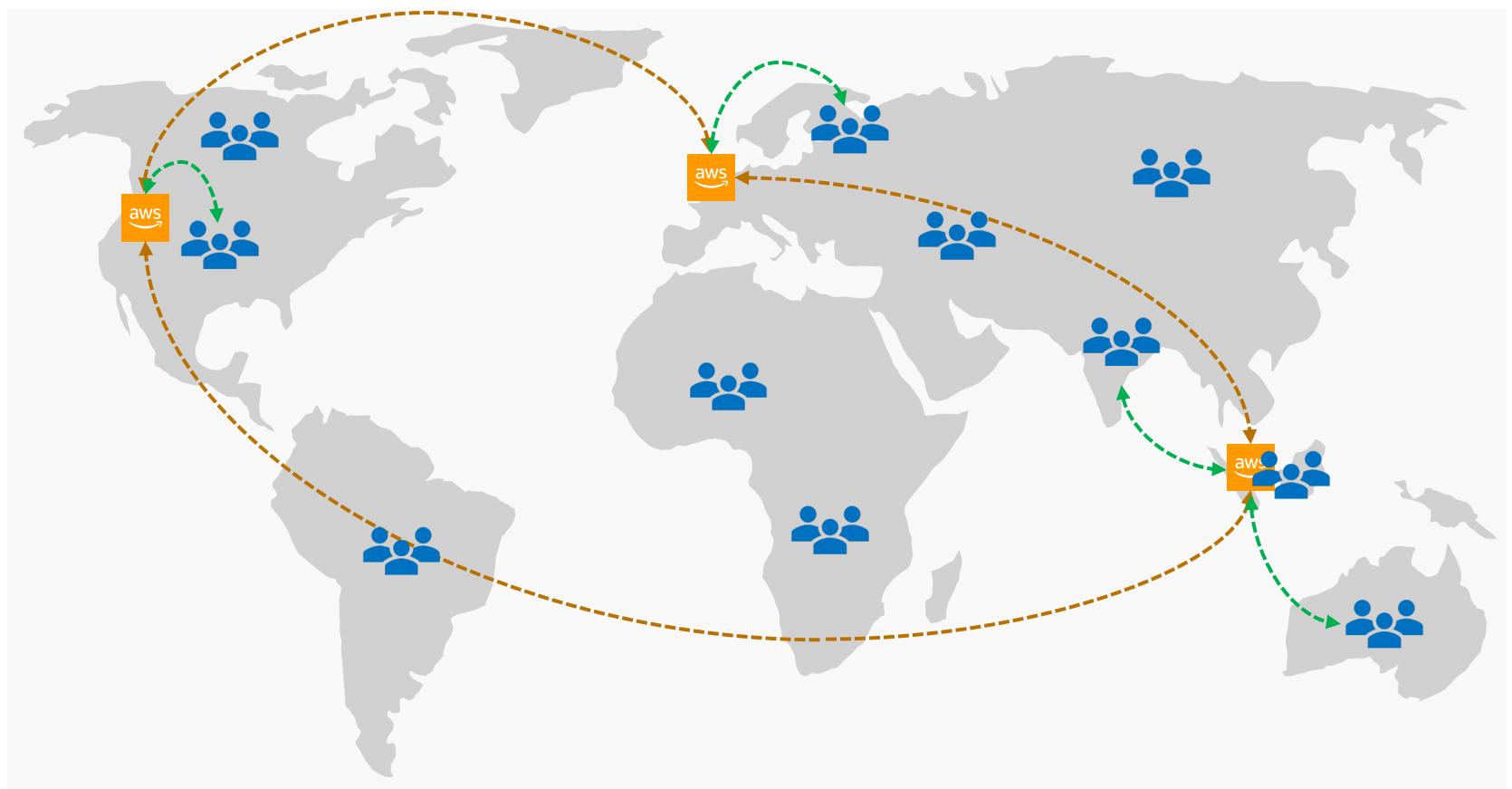
Why Global Tables



Why Global Tables



Why Global Tables



DynamoDB Global Tables



Demo



DataCumulus | RIZMAXed

Fine-grained access control in DynamoDB

- Can use IAM to control access to DynamoDB resources
- DynamoDB does not support tag-based conditions
- Can use condition keys in your IAM policy for fine-grained access control
 - Can restrict access to certain items / attributes based on user identity (in a table or a secondary index)
 - Example – allow users to access only the items that belong to them, based on certain primary key values

The diagram illustrates fine-grained access control in DynamoDB. On the left, two blue boxes represent IAM policies:

- User 12broiu45 can access
- User 34oiusd21 can access

Two arrows point from these boxes to a table on the right, indicating that each user can access specific items based on their user ID.

user_id	game_id	game_ts	result	duration
12broiu45	1234	"2020-03-15T17:43:08"	win	45
12broiu45	3456	"2020-06-20T19:02:32"	lose	33
34oiusd21	4567	"2020-02-11T-04:11:31"	lose	45

Fine-Grained Access Control for DynamoDB

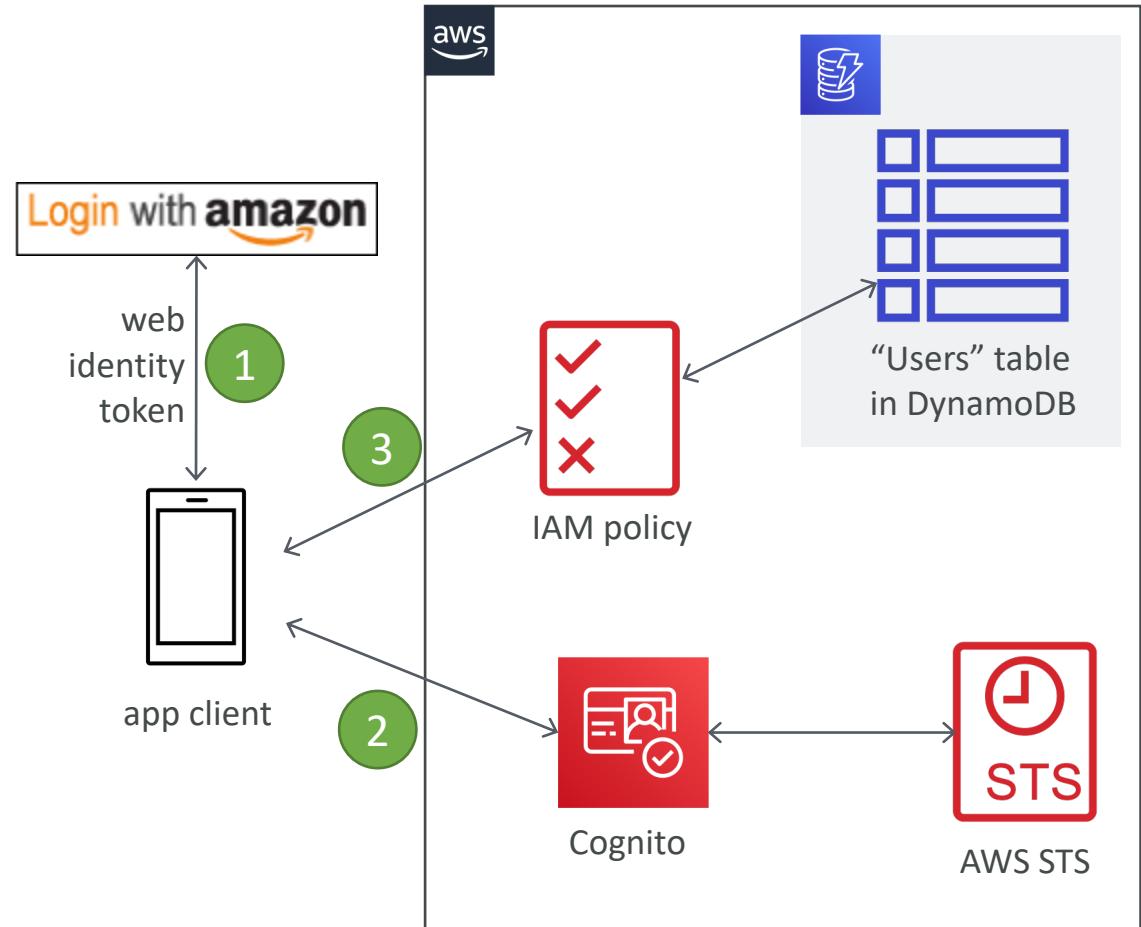
```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "FineGrainedAccessstoMessagesTable",  
      "Effect": "Allow",  
      "Action": [ "dynamodb:GetItem", "dynamodb:BatchGetItem",  
                 "dynamodb:Query", "dynamodb:PutItem" ],  
      "Resource": [  
        "arn:aws:dynamodb:us-west-2:0123456789:table/Messages"],  
      "Condition": {  
        "ForAllValues:StringEquals": {  
          "dynamodb:LeadingKeys": [  
            "${www.amazon.com:user_id}"  
          ],  
          "dynamodb:Attributes": [  
            "UserId", "MessageID", "Content", "Timestamp" ]  
        },  
        "StringEqualsIfExists": {  
          "dynamodb:Select": "SPECIFIC_ATTRIBUTES"  
        }  
      }  
    }  
  ]  
}
```

- ForAllValues:StringEquals – compares the requested attribute values with those in the table
- dynamodb:LeadingKeys – represents partition key. E.g. access will be allowed only if user's user_id matches the primary key value on the table
- dynamodb:Attributes – to limit access to specific attributes

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/specifying-conditions.html>

DynamoDB Web Identity Federation

- Also called as DynamoDB federated identities
- For authentication and authorization of app users
- No need to create individual IAM users
- Login with an identity provider (like Google / Facebook / Amazon) and get a web identity token
- Use Cognito to exchange the web identity token with temporary IAM credentials (STS token)
 - or you can also call STS API directly
- Use the temporary credentials to access DynamoDB (as per the role associated with the credentials)
- Can use fine-grained access control (with condition keys)



CloudWatch Contributor Insights

- Contributor Insights shows you the most accessed and throttled items in DynamoDB
- Also helps you analyze time-series data
- Supported for DynamoDB and CloudWatch Logs
- Identify outliers/contributors impacting system and application performance
- Find the heaviest traffic patterns
- Analyze the top system processes
- Displayed on CloudWatch dashboard
- Integrated with CloudWatch alarms



CloudWatch



DynamoDB

Redshift

Data = Understanding!

Amazon Redshift – Overview



Amazon Redshift

- OLAP database (Data warehousing solution) based on PostgreSQL
- OLAP = Online Analytical Processing
- Can query petabytes of **structured and semi-structured** data across your data warehouse and your data lake using standard SQL
- 10x performance than other data warehouses
- **Columnar** storage of data (instead of row based)
- Massively Parallel Query Execution (MPP), highly available
- Has a SQL interface for performing the queries
- BI tools such as AWS Quicksight or Tableau integrate with it

Redshift (contd.)

- Data is loaded from S3, Kinesis Firehose, DynamoDB, DMS...
- Can contain from 1 node to 128 compute nodes, up to 160 GB per node
- Can provision multiple nodes, but it's not Multi-AZ
- Leader node: for query planning, results aggregation
- Compute node: for performing the queries, send results to leader
- Backup & Restore, Security VPC / IAM / KMS, Monitoring
- Redshift Enhanced VPC Routing: COPY / UNLOAD goes through VPC
- Redshift is provisioned, so it's worth it when you have a sustained usage
(use Athena instead if the queries are sporadic)



Amazon Redshift

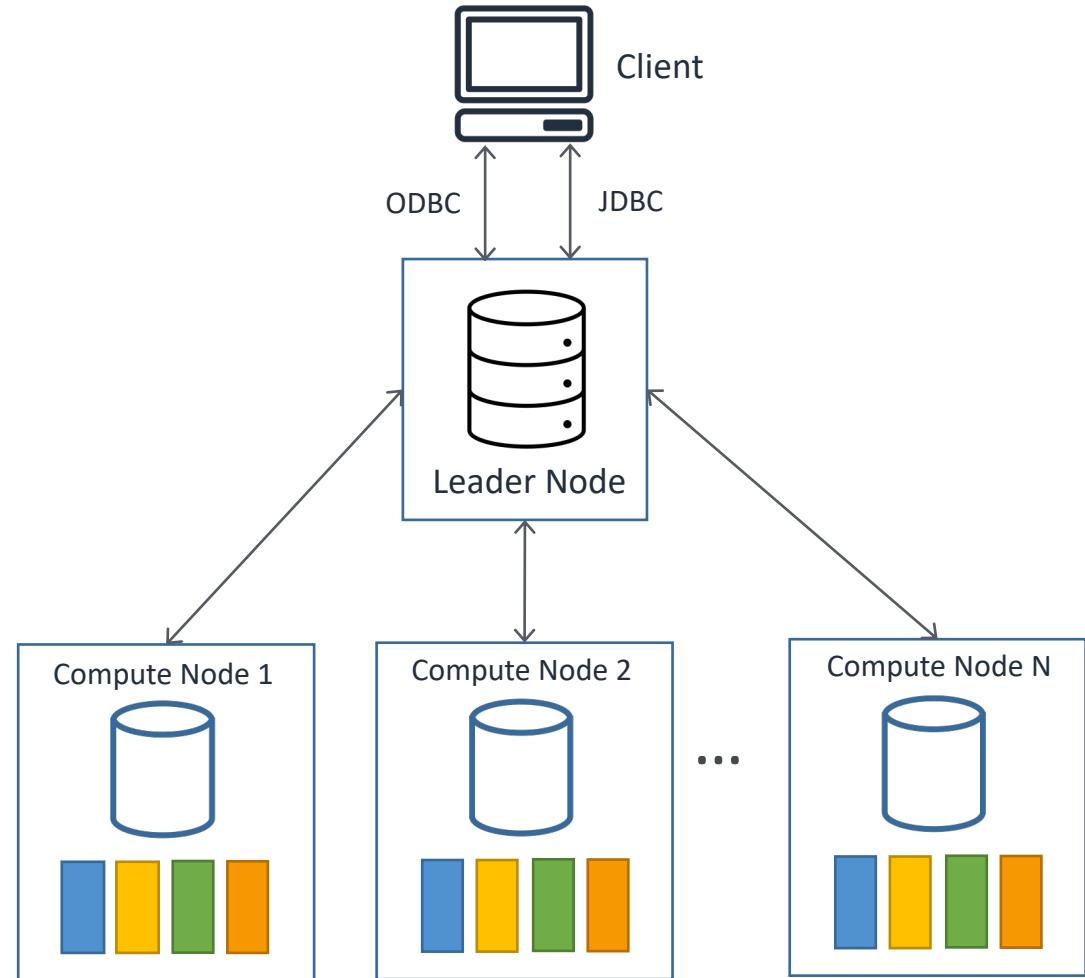
Creating a Redshift Cluster



Demo

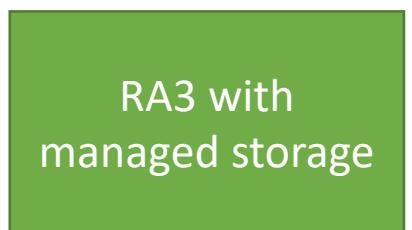
Redshift Architecture

- Massively parallel columnar database, runs within a VPC
- Single leader node and multiple compute nodes
- You can connect to Redshift using any application supporting JDBC or ODBC driver for PostgreSQL
- Clients query the leader node using SQL endpoint
- A job is distributed across compute nodes.
- Compute nodes partition the job into slices.
- Leader node then aggregates the results and returns them to the client



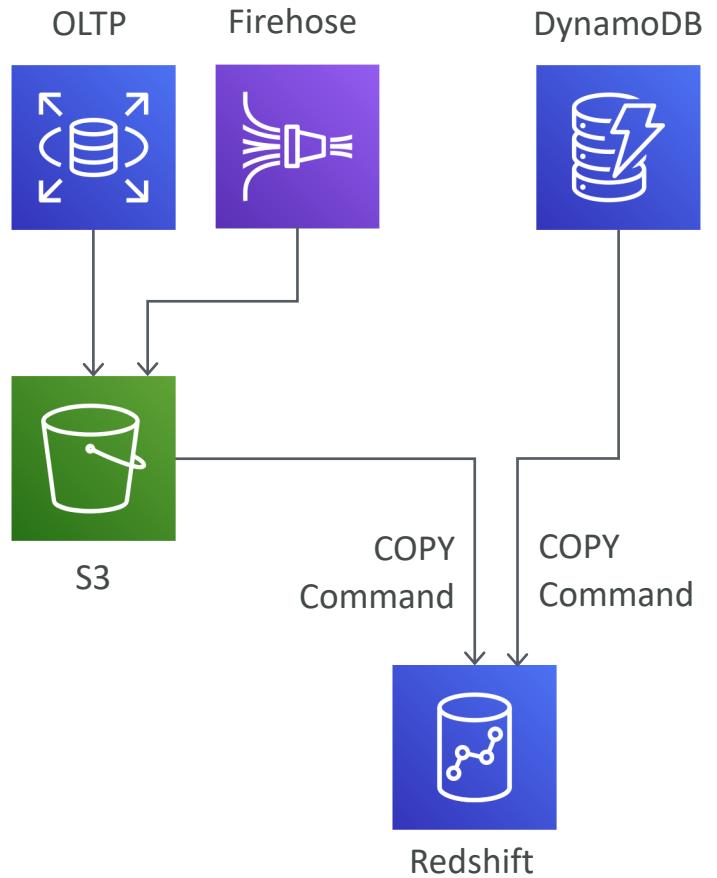
Redshift node types

- Dense compute nodes (DC2)
 - For compute-intensive DW workloads with local SSD storage
- Dense storage nodes (DS2)
 - For large DWs, uses hard disk drives (HDDs)
- RA3 nodes with managed storage
 - For large DWs, uses large local SSDs
 - Recommended over DS2
 - Automatically offloads data to S3 if node grows beyond its size
 - Compute and managed storage is billed independently



Loading data into Redshift

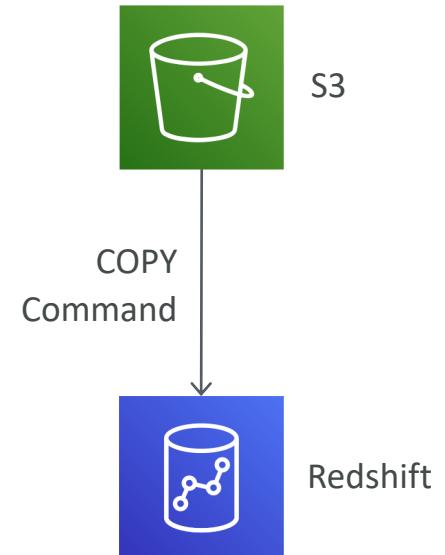
- Typically, data from OLTP systems is loaded into Redshift for analytics and BI purposes
 - Data from OLTP systems can be loaded into S3 and data from S3 can then be loaded into Redshift
 - Data from Kinesis Firehose can also be loaded in the same way
- COPY command
 - Loads data from files stored in S3 into Redshift
 - Data is stored locally in the Redshift cluster (persistent storage = cost)
 - DynamoDB table data and EMR data can also be loaded using COPY command



Loading data from S3 with COPY command

```
copy users from 's3://my_bucket/ticket/allusers_pipe.txt'  
credentials 'aws_iam_role=arn:aws:iam::0123456789:role/MyRedshiftRole'  
delimiter '|' region 'us-west-2';
```

- Create an IAM Role
- Create your Redshift cluster
- Attach the IAM role to the cluster
- The cluster can then temporarily assume the IAM role on your behalf
- Load data from S3 using COPY command



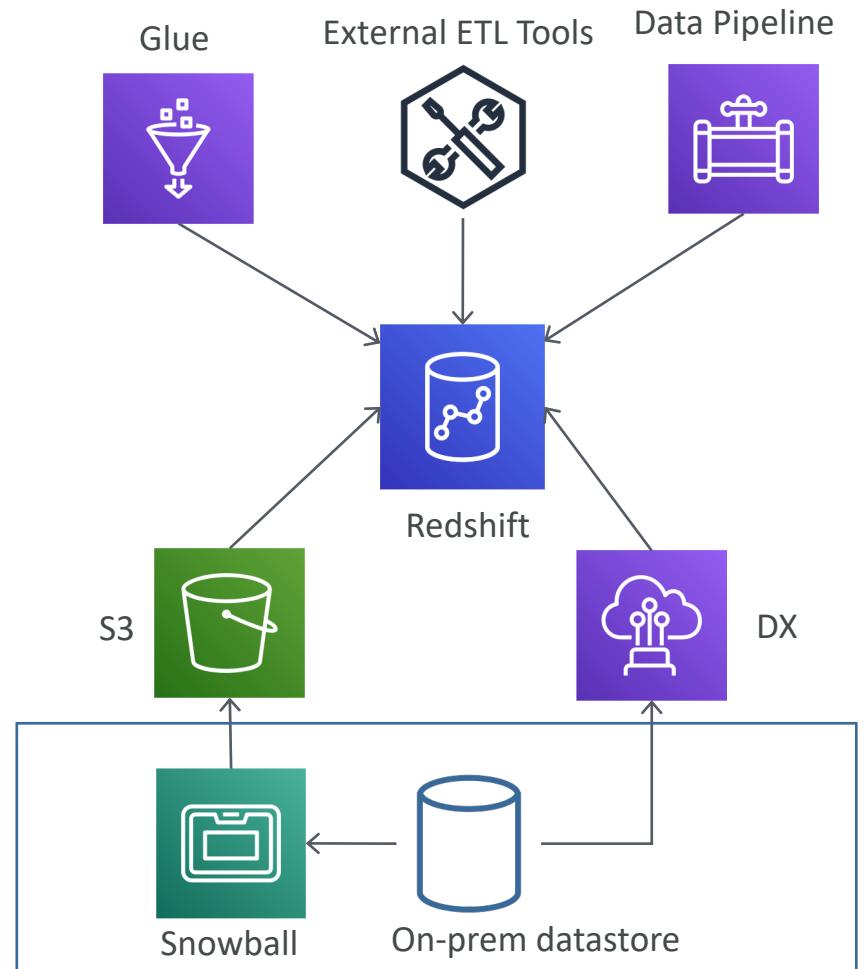
Loading data from S3 into Redshift



Demo

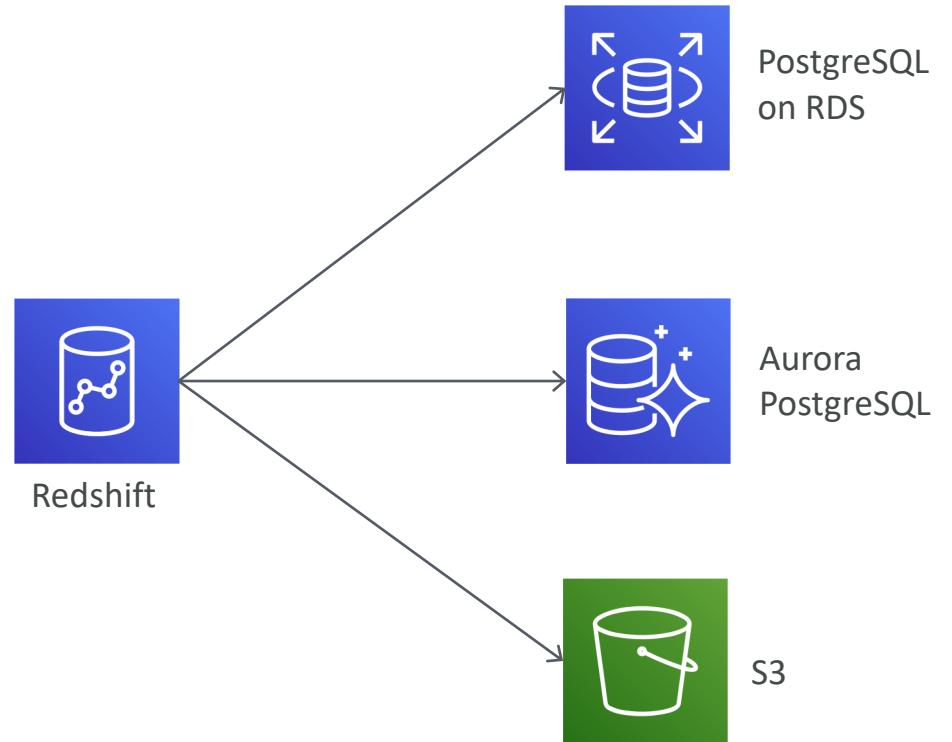
More ways to load data into Redshift

- Use AWS Glue – fully managed ETL service
- ETL = Extract, Transform, and Load
- Use ETL tools from APN partners
- Use Data Pipeline
- For migration from on-premise, use:
 - AWS Import/Export service (AWS Snowball)
 - AWS Direct Connect (private connection between your datacenter and AWS)



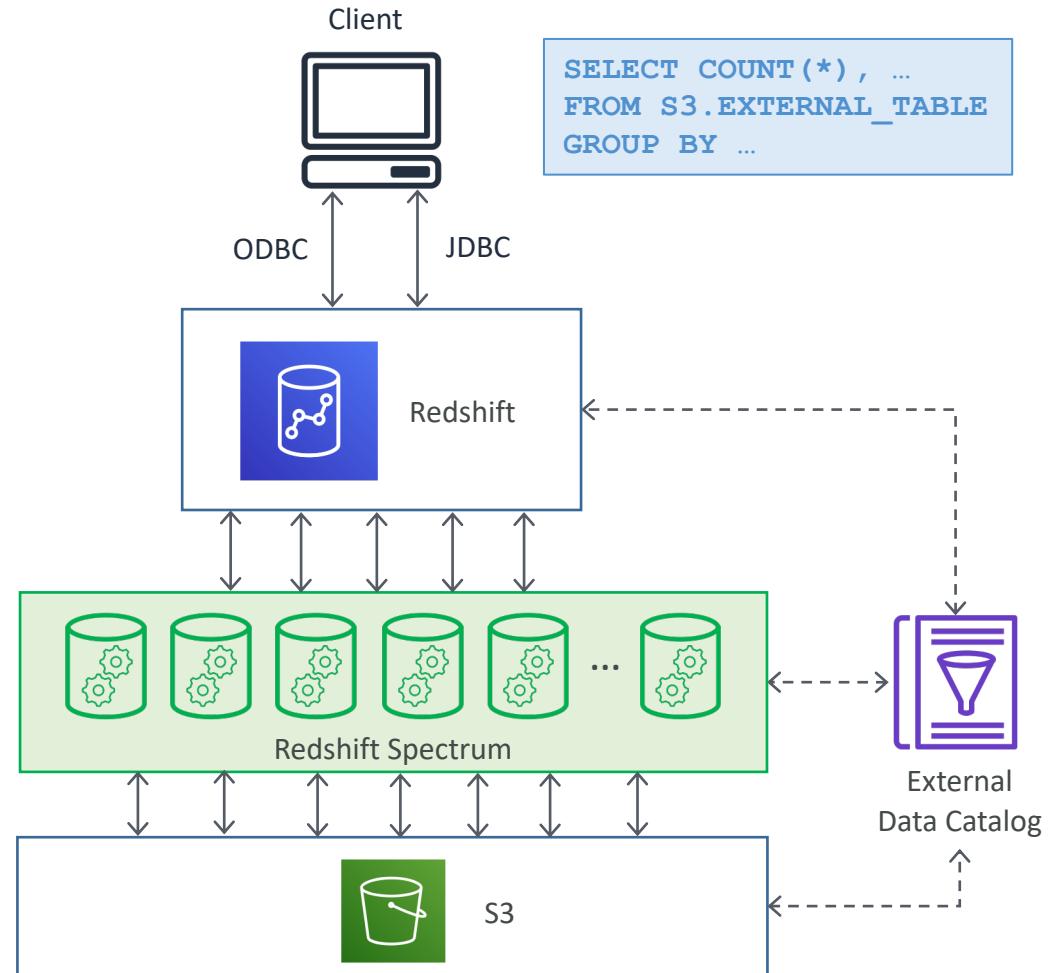
Querying external data with Redshift

- Two ways
 - Redshift Spectrum
 - Redshift Federated Query



Redshift Spectrum

- Query exabytes of data from S3 without loading it into Redshift
- Must have a Redshift cluster available to start the query
- The query is then submitted to thousands of Redshift Spectrum nodes
- External table structure/schemas can be created in external data catalog like Athena / Glue / Apache Hive metastore (EMR)
- These external tables are read-only (insert / update / delete operations not possible)
- Redshift cluster and S3 bucket must be in the same region



Querying S3 data with Redshift Spectrum



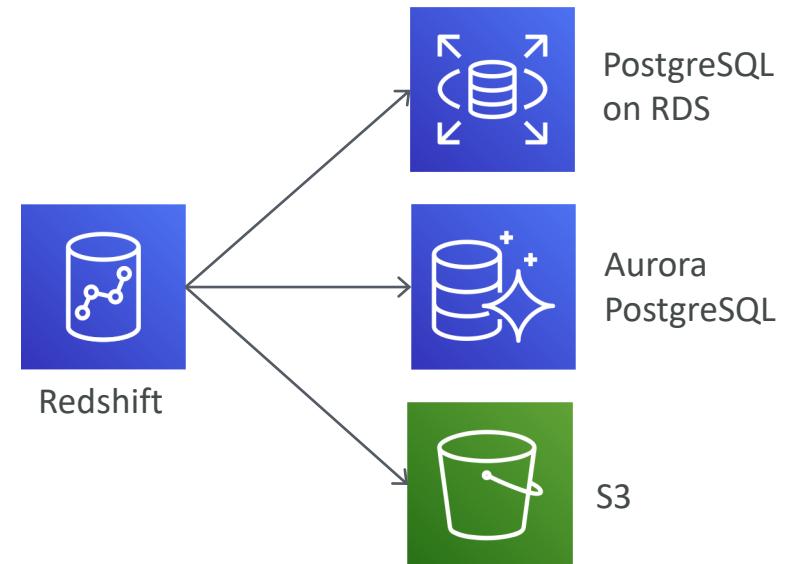
Demo

Redshift Federated Query

- Query and analyze data across different DBs, DWs, and data lakes
- Currently works with Redshift, PostgreSQL on RDS, Aurora PostgreSQL and S3

Example query referencing S3, Redshift and Aurora PostgreSQL

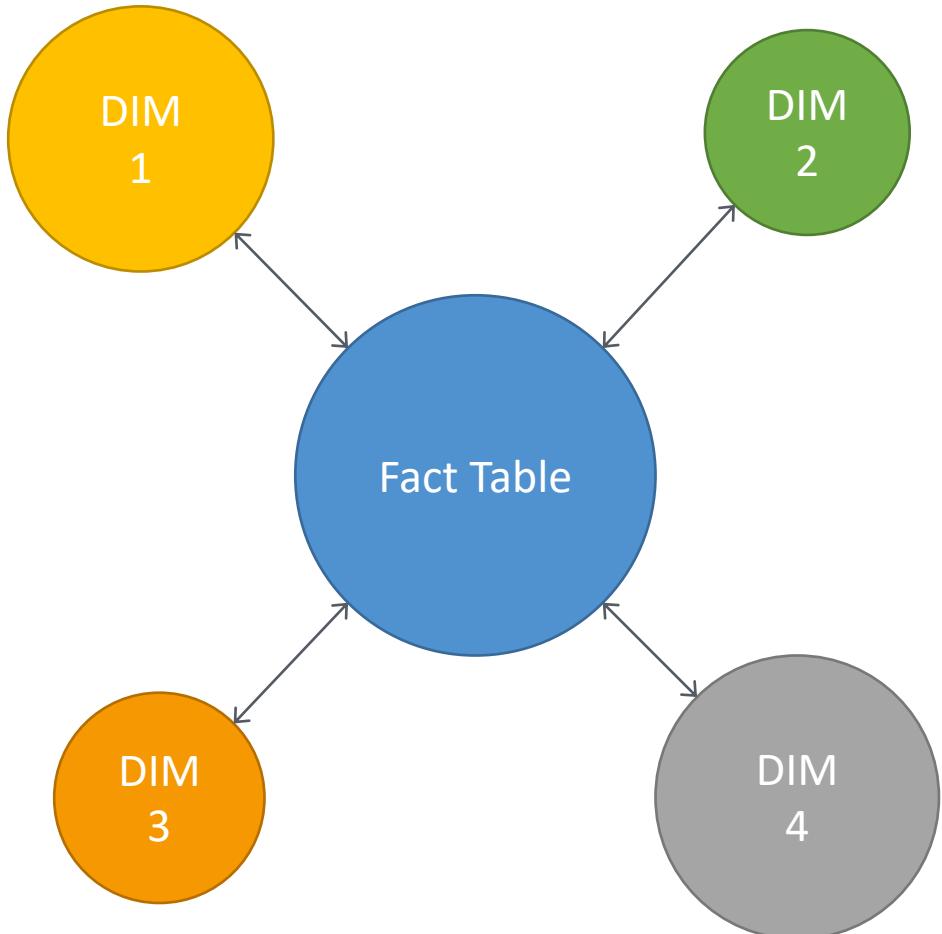
```
CREATE VIEW lineitem_all AS
    SELECT l_orderkey,l_partkey, ...
    FROM s3.lineitem_1t_part
    UNION ALL SELECT * FROM public.lineitem
    UNION ALL SELECT * FROM aurorapostgres.lineitem
        with no schema binding;
```



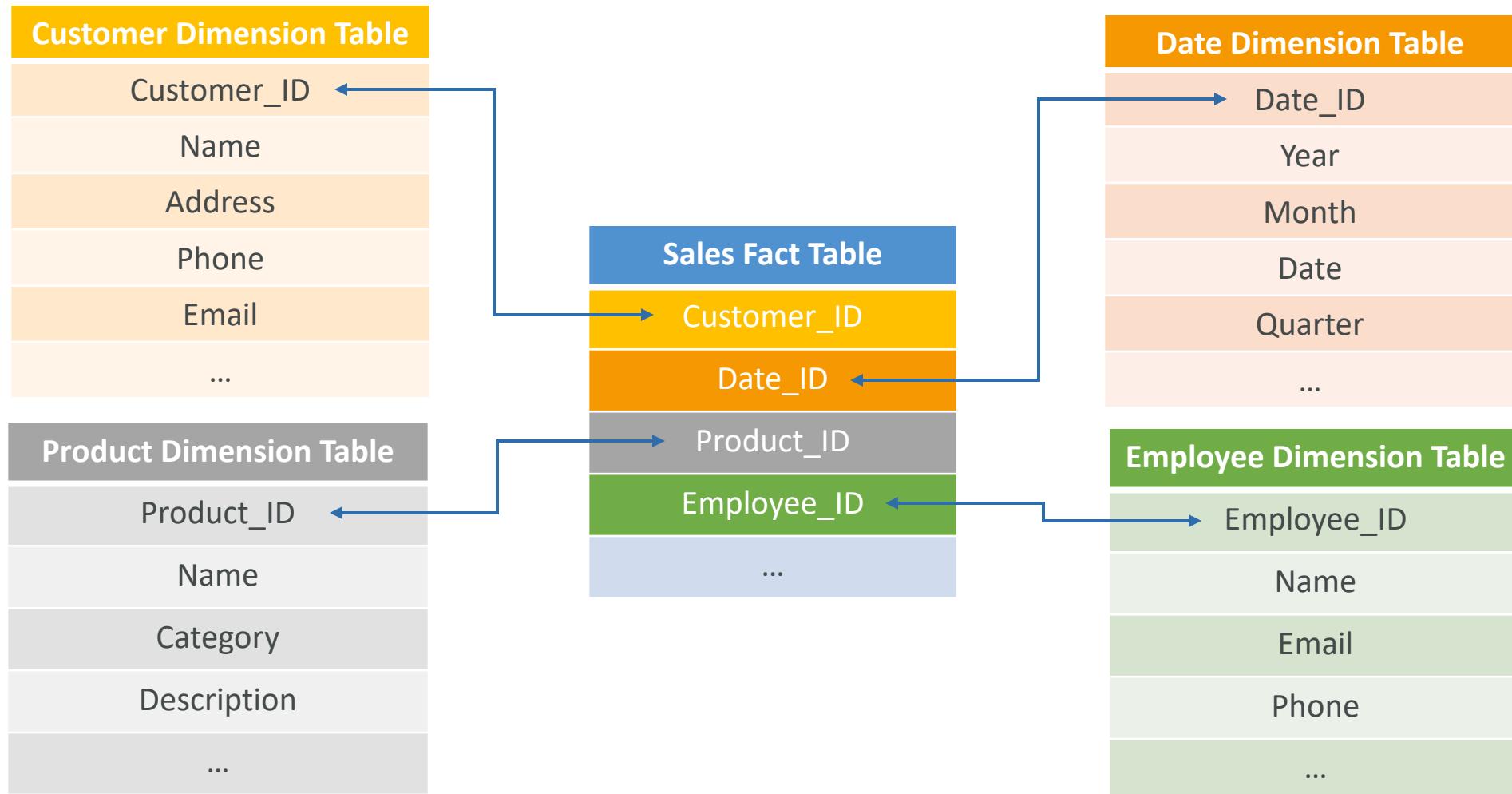
https://docs.aws.amazon.com/redshift/latest/dg/federated_query_example.html

Star Schema

- Typical way of organizing data in a data warehouse
 - Two types of tables – fact tables and dimension tables
 - A star pattern consists of a fact table and multiple dimension tables
 - Fact table has foreign key relationships with multiple dimension tables
 - Dimension tables are generally small (fewer records, but often with many fields)
- Recommended to store:
 - smaller dimension tables as local Redshift tables
 - larger fact tables as external Spectrum tables



Example of a Star Schema

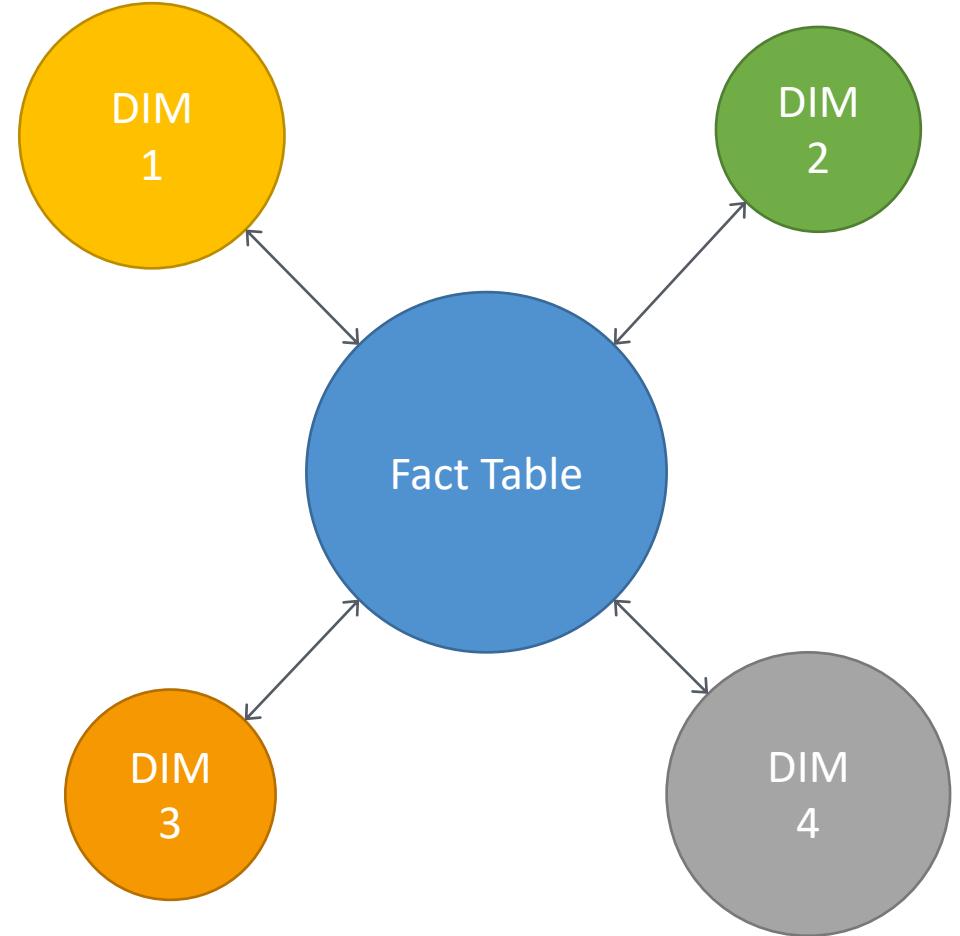


Redshift Key Constraints

- Redshift does not support indexes (clustered or non-clustered)
- Has a concept of sort keys
 - data is stored on disk in a sorted order by sort key
- Primary Key, Unique keys, and Foreign key constraints are not enforced (are informational only)
- Query optimizer uses these constraints to generate more efficient query plans
- You can enforce PK/FK relationships through your application

Redshift Table Design

- Key factors for efficient table design
 - Data distribution – how data is distributed across nodes
 - Sort strategies – how data is sorted in the tables
 - Compression – to reduce storage and I/O needs



Redshift Data Distribution

- Data is stored in columns (as against rows in a typical OLTP database)
- Data corresponding to a given field (across rows) is stored together and can be queried easily
- Data distribution refers to how data is distributed across nodes in the cluster
- When creating a Redshift table, you specify a distribution style

Distribution style	Description
EVEN	Spreads data evenly across all nodes in the cluster (default option, decent performance)
ALL	Table data is put on each node (good for smaller dimension tables, or for frequently used tables that are heavily joined)
KEY	Rows with the same DISTKEY column value are placed on the same node (good for known table relationships)
AUTO	Initially assigns ALL to a small table, changes to EVEN as table size grows

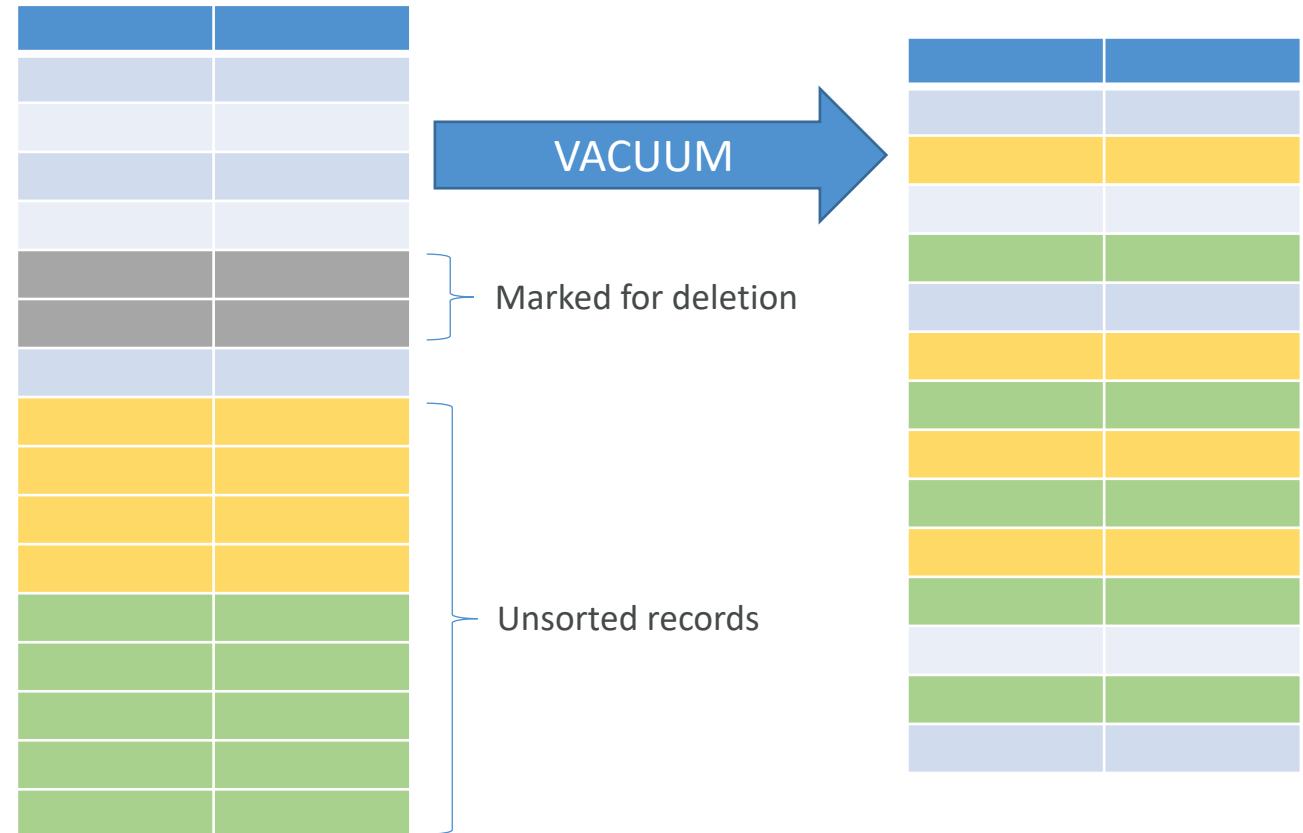
Redshift Sort Styles

- Single-column sort key (e.g. Dept)
- Compound sort key
 - more than one column as sort key
 - e.g. Dept + Location
 - Hierarchical (order of the column in the sort key is important)
- Interleaved sort key
 - gives equal weight to each column (or subset of columns) in the sort key
 - In effect, you can have multiple sort key combinations
 - e.g. Dept + Location, Location + Dept
- Must be defined at the table creation time

Dept	Location	Emp Name
HR	NY	Marie
HR	SFO	John
IT	NY	Tom
IT	LA	Tracy

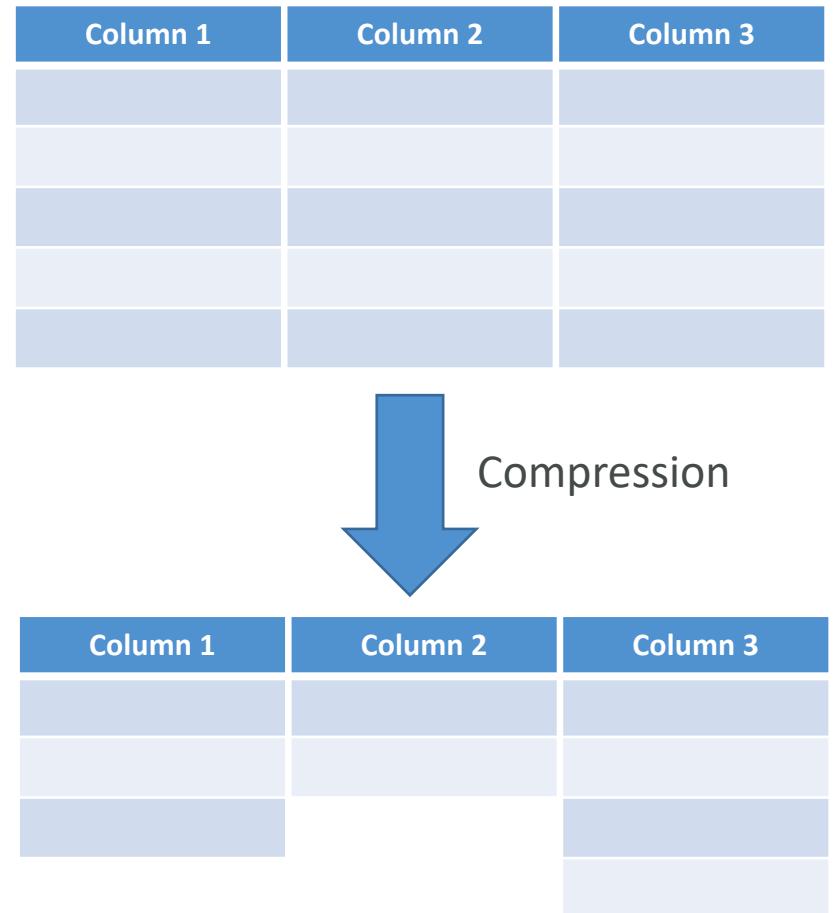
VACUUM operation in the tables

- As you delete row from or add more rows to a sorted table containing data, performance might deteriorate over time
- VACUUM operation re-sorts rows in one or all DB tables
- And reclaims space from table rows marked for deletion
- Redshift automatically sorts data and runs VACUUM DELETE in the background
- Need not run manually, but you can if required



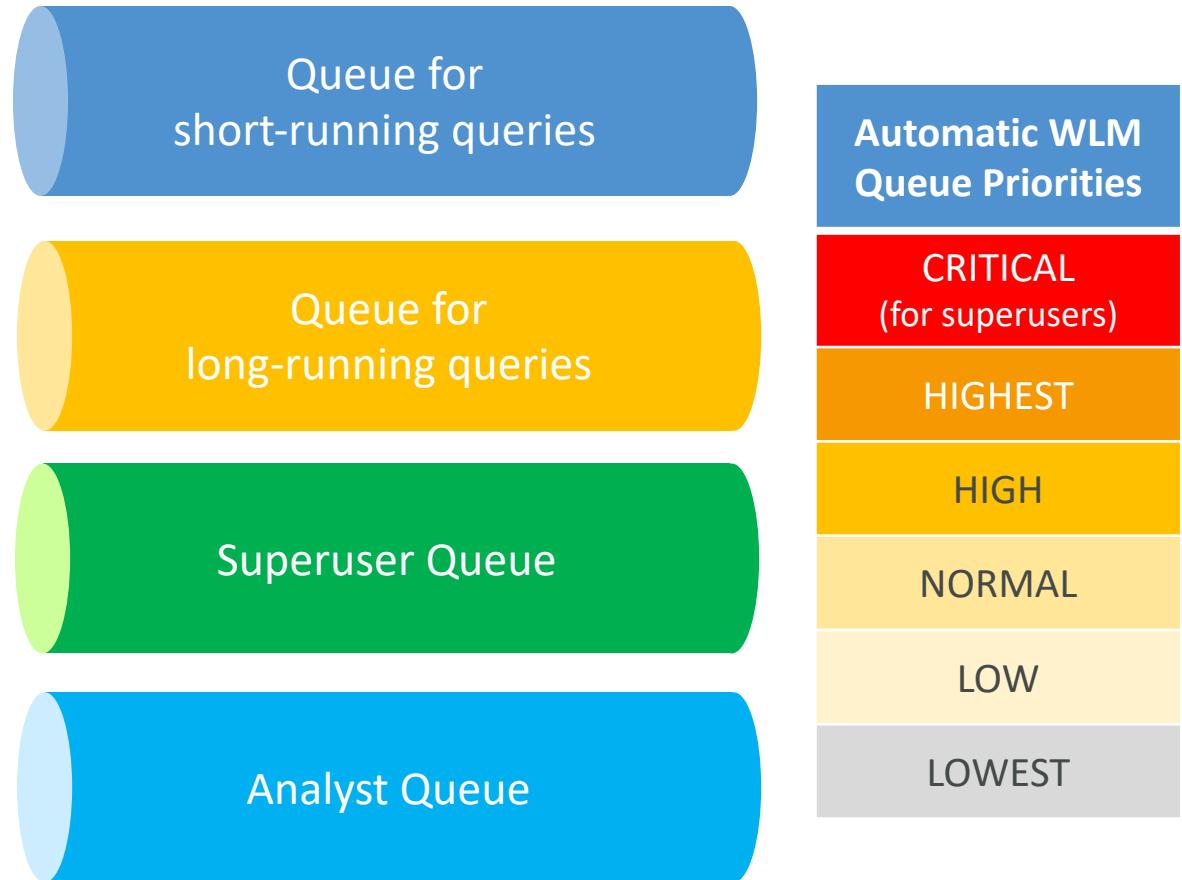
Redshift Compression

- Column-level operation to reduce size of stored data
- Reduces disk I/O and improves query performance
- Compression type = encoding
- Each column can be separately compressed with different encodings (manually / automatically)
- COPY command applies compression by default
- RAW = No compression (default for sort keys)
- LZO = Very high compression with good performance (default encoding)
- Encoding cannot be changed after table creation



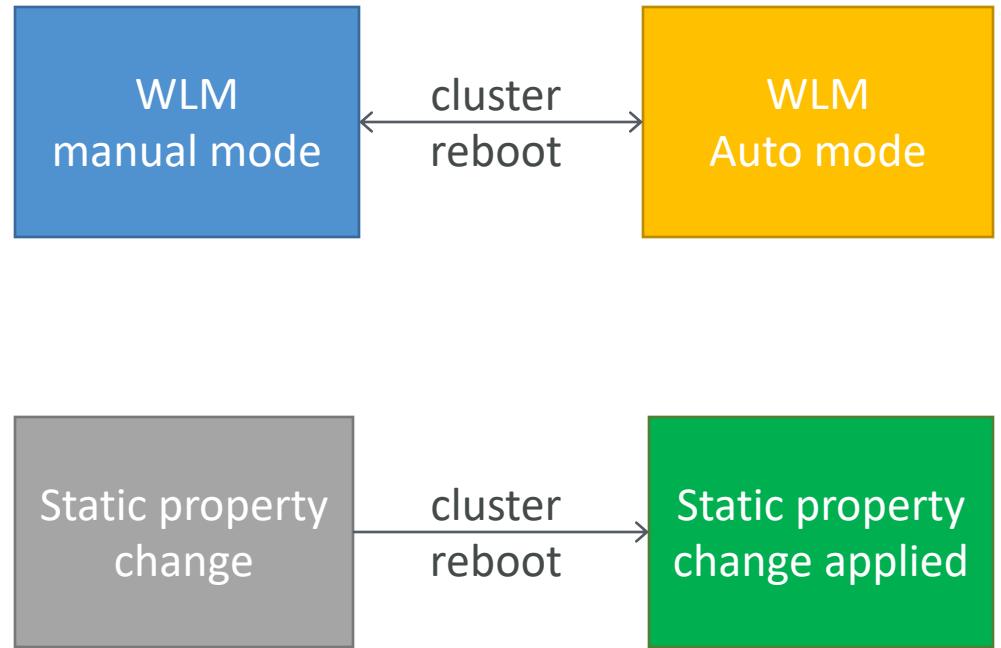
Redshift Workload Management (WLM)

- Helps you prioritize workloads
- Can prevent long-running queries from impacting short-running ones
- Two modes – Automatic WLM and Manual WLM
- Automatic WLM supports queue priorities
- SQA (Short query acceleration)
 - prioritize selected short running queries w/o creating dedicated queue



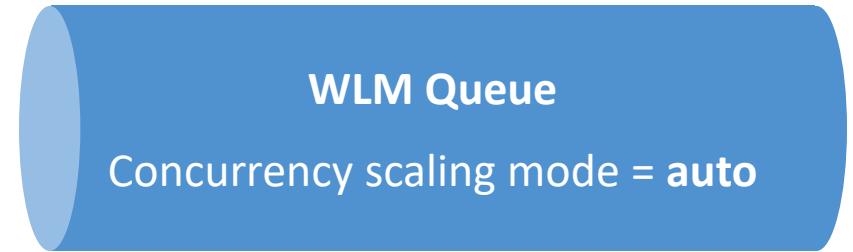
Modifying the WLM configuration

- Switching between Manual and Auto WLM modes requires cluster reboot
- WLM uses **parameter groups** for its config
- WLM configuration properties are either dynamic or static
- Dynamic property changes do not require cluster reboot
- Static property changes require cluster reboot



Redshift Concurrency Scaling

- Automatically scales cluster capacity to support increase in concurrent reads
- Can scale to a virtually unlimited # of queries
- It's a dynamic parameter in WLM queue config
- To enable, set **Concurrency Scaling mode = auto** for the given WLM queue
- Can continue R/W during concurrency scaling
- You get one-hour of free concurrency scaling credits per day



Scaling in Redshift

- Elastic Resize

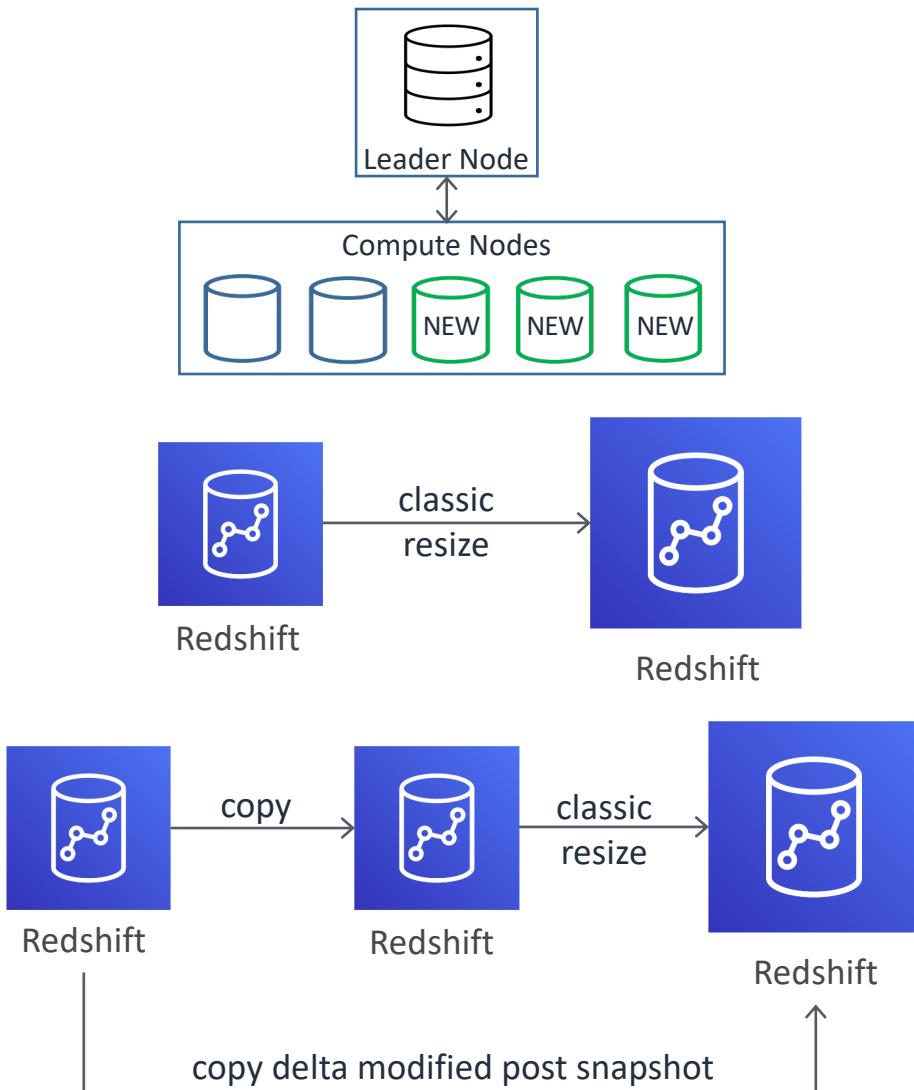
- Add / remove nodes to / from an existing cluster in minutes (4 to 8 mins)
- Is a manual process, cluster is unavailable during resize op

- Classic resize

- Change node type, the number of nodes, or both
- Copies data to a new cluster
- Source cluster will be in read-only mode during resize op

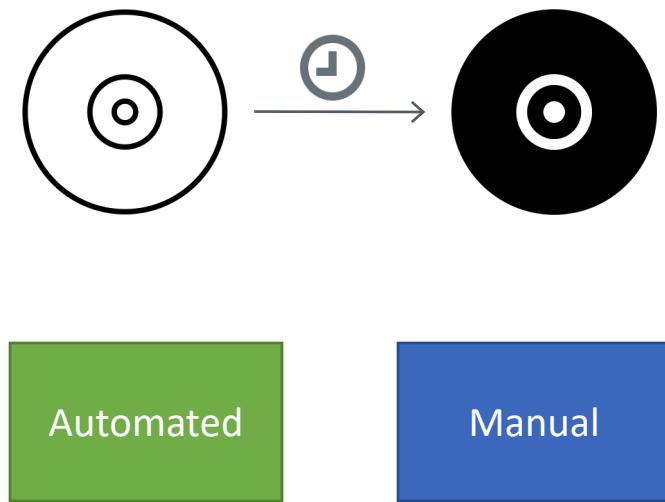
- Snapshot, restore, and classic resize

- Copy the cluster and resize the copied cluster
- Source cluster remains available throughout
- Manually copy the delta to the new cluster.



Redshift Backup and Restore

- Redshift maintains at least three copies of your data – the original, replica on the compute nodes, and a backup in S3
- Snapshots are point-in-time backups of a cluster; stored internally in S3
- Snapshots are incremental (only what has changed is saved)
- You can restore a snapshot into a **new cluster**
- Automated – every 8 hours / every 5 GB / or on a schedule. Set retention
- Manual – snapshot is retained until you delete it
- Can configure Redshift to automatically copy snapshots (automated or manual) to another AWS Region



Copying Redshift snapshots to another Region

- For unencrypted snapshots
 - configure cross-region snapshots
- For encrypted snapshots
 - Configure cross-region snapshots and additionally specify a snapshot copy grant
 - Snapshot copy grant requires a name and a KMS key and gets created in the destination region

Configure cross-region snapshot

Configure Amazon Redshift to automatically copy your automated and manual snapshots to another AWS Region. You incur data transfer costs when snapshots are copied to the destination AWS Region.

Copy snapshots

Yes
 No

Destination AWS Region

us-east-1

Automated snapshot retention period (days)

1

Manual snapshot retention period

How long do you want to retain your snapshot?

Indefinitely

The value must be from 1 to 3653 days.

Cancel Save

Configure cross-region snapshot

Configure Amazon Redshift to automatically copy your automated and manual snapshots to another AWS Region. You incur data transfer costs when snapshots are copied to the destination AWS Region.

Copy snapshots

Yes
 No

Destination AWS Region

us-east-1

Automated snapshot retention period (days)

1

Manual snapshot retention period

How long do you want to retain your snapshot?

Indefinitely

The value must be from 1 to 3653 days.

Choose a snapshot copy grant

Create new grant

Snapshot copy grant name

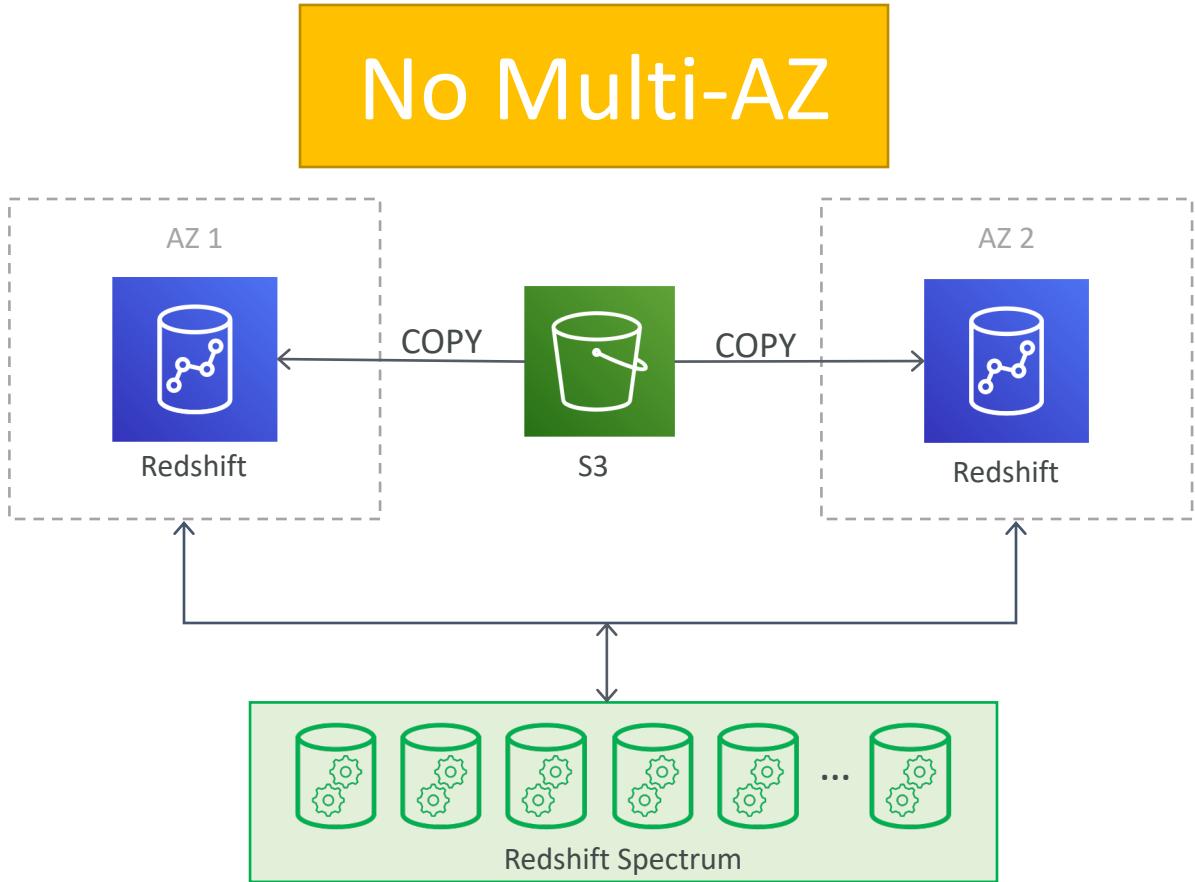
my-grant

Use AWS owned key
 Use key from current account
 Use key from different account

Cancel Save

Redshift Multi-AZ Deployments

- Multi-AZ deployments are not supported
- Alternative way:
 - Run Redshift clusters in multiple AZs by loading same data into different AZs
 - Restore a cluster snapshot to a different AZ
 - Use Spectrum to read S3 data from Redshift clusters spread across AZs



Redshift Availability and Durability

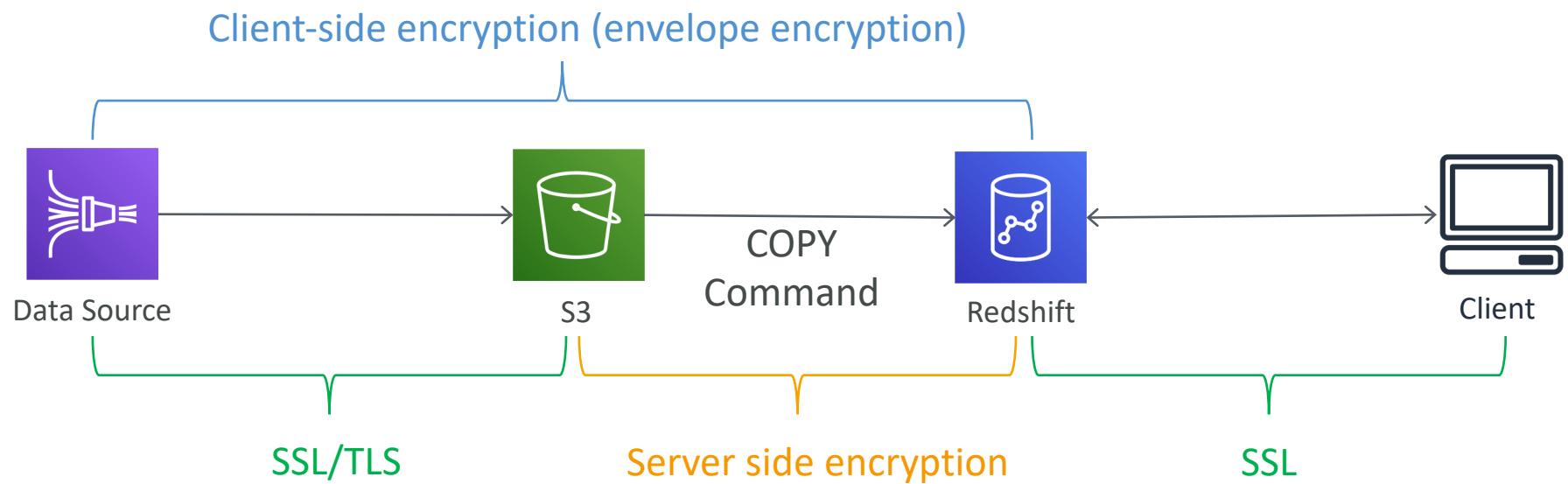
- Drive failure
 - Redshift cluster remains available (with decline in query performance)
 - Transparently uses a replica stored on other drives within the node
 - Single node clusters do not support data replication (must restore from snapshot on S3)
- Node failure
 - Automatically detects and replaces a failed node
 - Cluster remains unavailable until node is replaced
- AZ outage
 - Cluster remains unavailable until AZ outage resolves, data is preserved
 - Can restore from snapshot to another AZ within region (frequently accessed data is restored first)



Redshift

Redshift Security – Encryption

- Encryption at rest – uses hardware-accelerated AES-256 encryption
- Can use your own KMS keys or HSMs (hardware security module)
- Encryption in transit – uses SSL connections
- API requests must be signed using SigV4 process
- Spectrum supports S3's Server-Side Encryption (SSE)
- Can use client-side encryption using CMK before sending data to S3



Redshift Security – IAM & Network

- Access to Redshift resources is controlled at four levels
 - Cluster management – using IAM policies
 - Cluster connectivity – using cluster security groups / VPC
 - Database access – using DB user accounts/groups and permissions
 - CREATE USER / GROUP statements
 - GRANT / REVOKE statements
 - IAM Authentication (temporary credentials and SSO)
- Compute nodes can only be accessed via leader node (not directly)



IAM



VPC



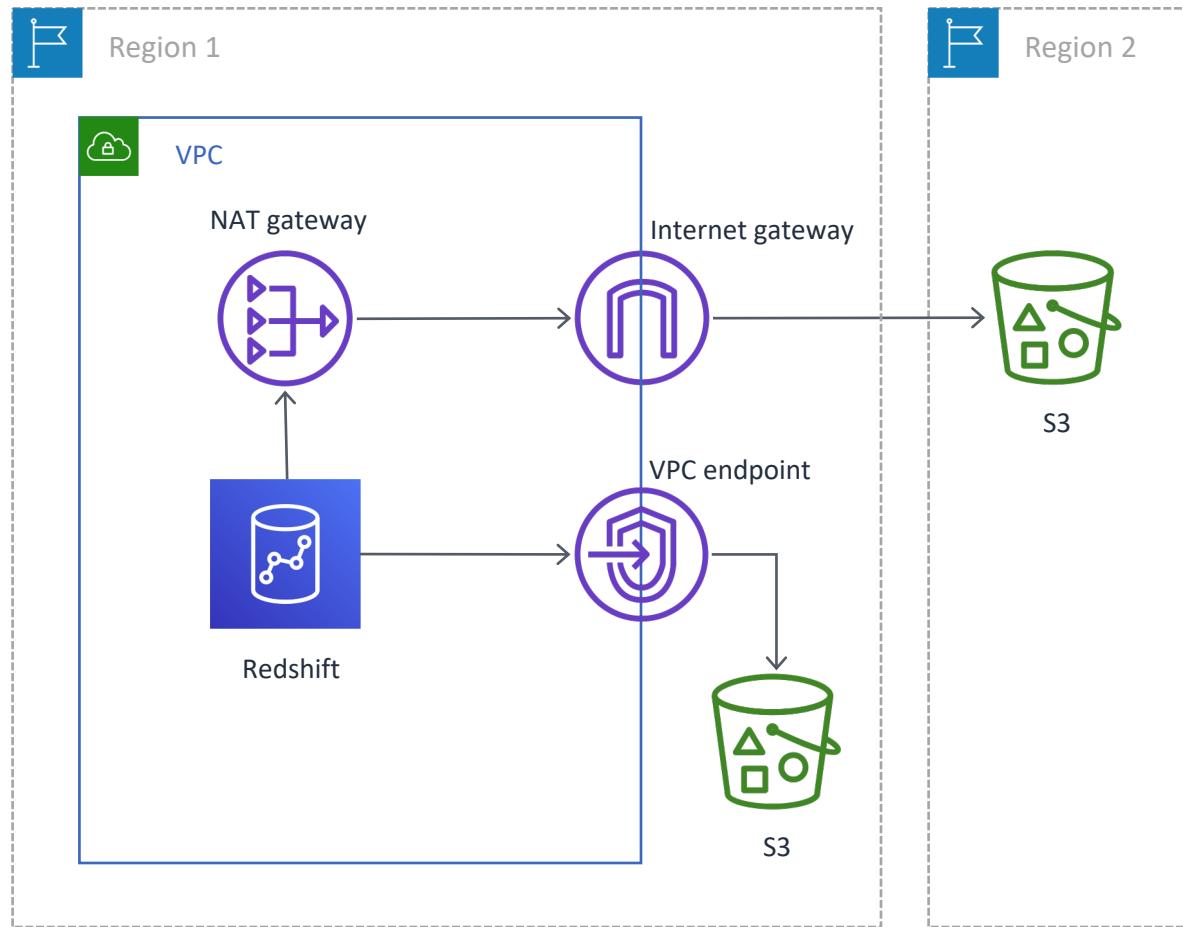
Database user accounts / groups



IAM Authentication

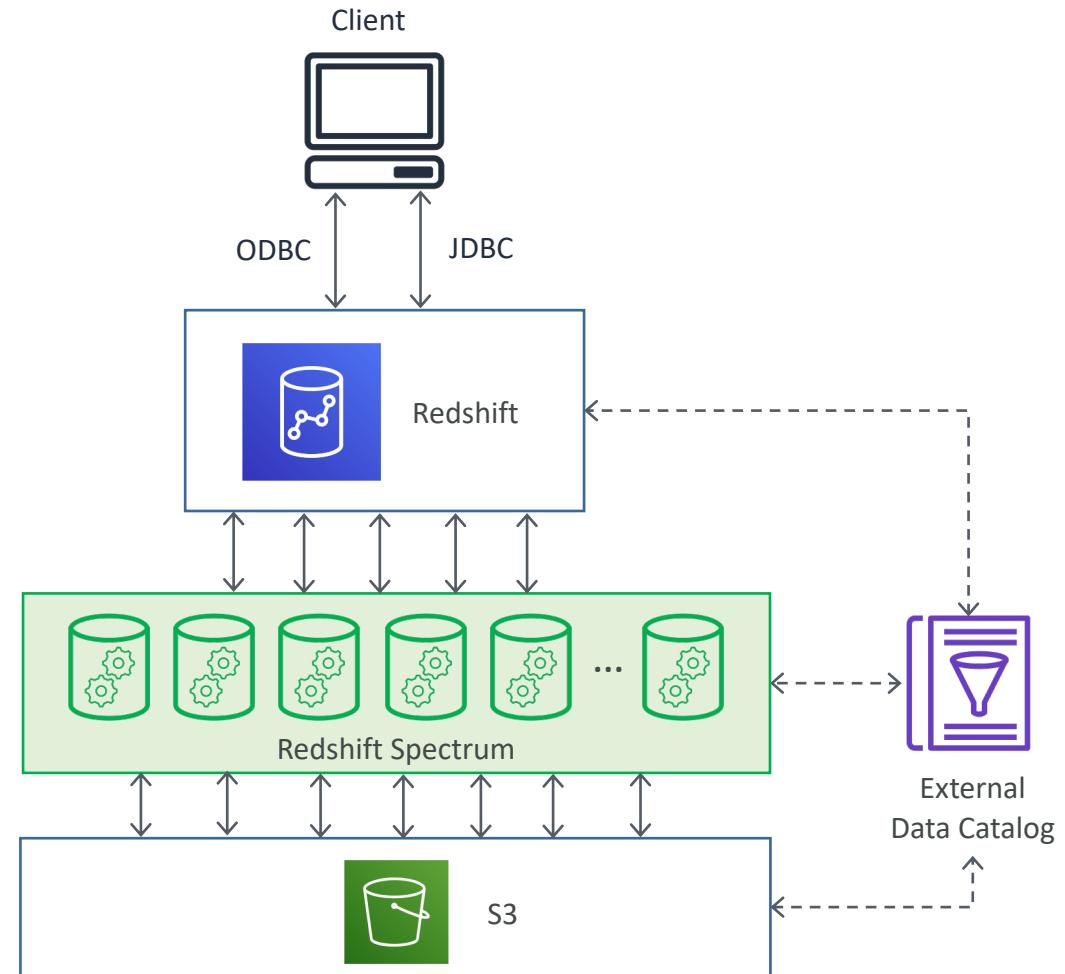
Enhanced VPC Routing in Redshift

- To manage data traffic between your Redshift cluster and other resources
- All COPY / UNLOAD traffic can go through VPC
- If disabled, all traffic is routed through internet (including traffic within AWS)
- S3 bucket within the same region – use VPC endpoints (S3 endpoints)
- S3 bucket in different region (or another service within AWS n/w) – use NAT GW
- To connect to AWS services outside your VPC – Use IGW



Redshift Spectrum w/ EnhancedVPC Routing

- Redshift Spectrum doesn't use enhanced VPC routing by default
- Additional config is required
- Modify your cluster's IAM role to allow traffic to the S3 bucket from Redshift Spectrum
- Modify the policy attached to the S3 bucket
 - Spectrum can't access S3 buckets with policy that restricts access to only specific VPC endpoints
 - Use a bucket policy that restricts access to only specific principals (users / accounts)
- Configure your VPC to allow your cluster to access AWS Glue / Athena



Redshift Monitoring

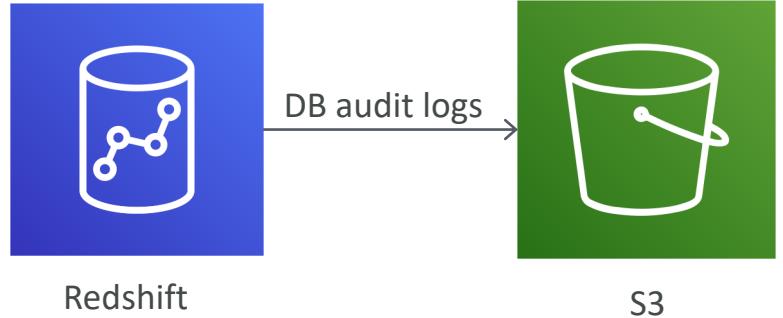
- Integrates with CloudWatch
- CloudWatch metrics
 - Accessible within Redshift console (or in CloudWatch)
 - Monitor CPU utilization, latency, and throughput etc.
- Query and Load performance
 - Not published as CloudWatch metrics
 - Displayed only in the Amazon Redshift console
 - Cluster performance, query history, DB performance, concurrency scaling data etc.
- Redshift Advisor
 - offers recommendations for cost optimization and performance tuning



CloudWatch

Database audit logging in Redshift

- Audit logs (stored in S3 buckets, must be enabled)
 - Connection log – connections / disconnections / auth attempts
 - User log – changes to user info
 - User activity log – logs each query
- STL / SVL tables –
 - Similar to audit logs
 - Available by default on every node in the cluster
 - e.g. STL_QUERY table logs all executed queries,
 - STL_CONNECTION_LOG logs connections / disconnections / auth attempts
- CloudTrail – captures all Redshift API calls as events



Redshift Pricing

- You pay only for what you use
 - Compute node hours
 - Backup storage – manual and automated
 - Data transfer (except data xfer b/w Redshift and S3 within region)
 - Spectrum data scanned – amount of S3 data scanned by your query
 - Managed Storage (per GB-month rate, only for RA3 node types)
 - Concurrency Scaling - per-second on-demand rate (in excess of free credits)
- No charges for Spectrum when you're not running queries



Redshift

Athena & Quicksight

- **Athena**

- Serverless SQL queries on top of your data in S3, pay per query, output to S3
- Supports CSV, JSON, Parquet, ORC
- Queries are logged in CloudTrail (which can be chained with CloudWatch logs)
- Great for sporadic queries



Athena

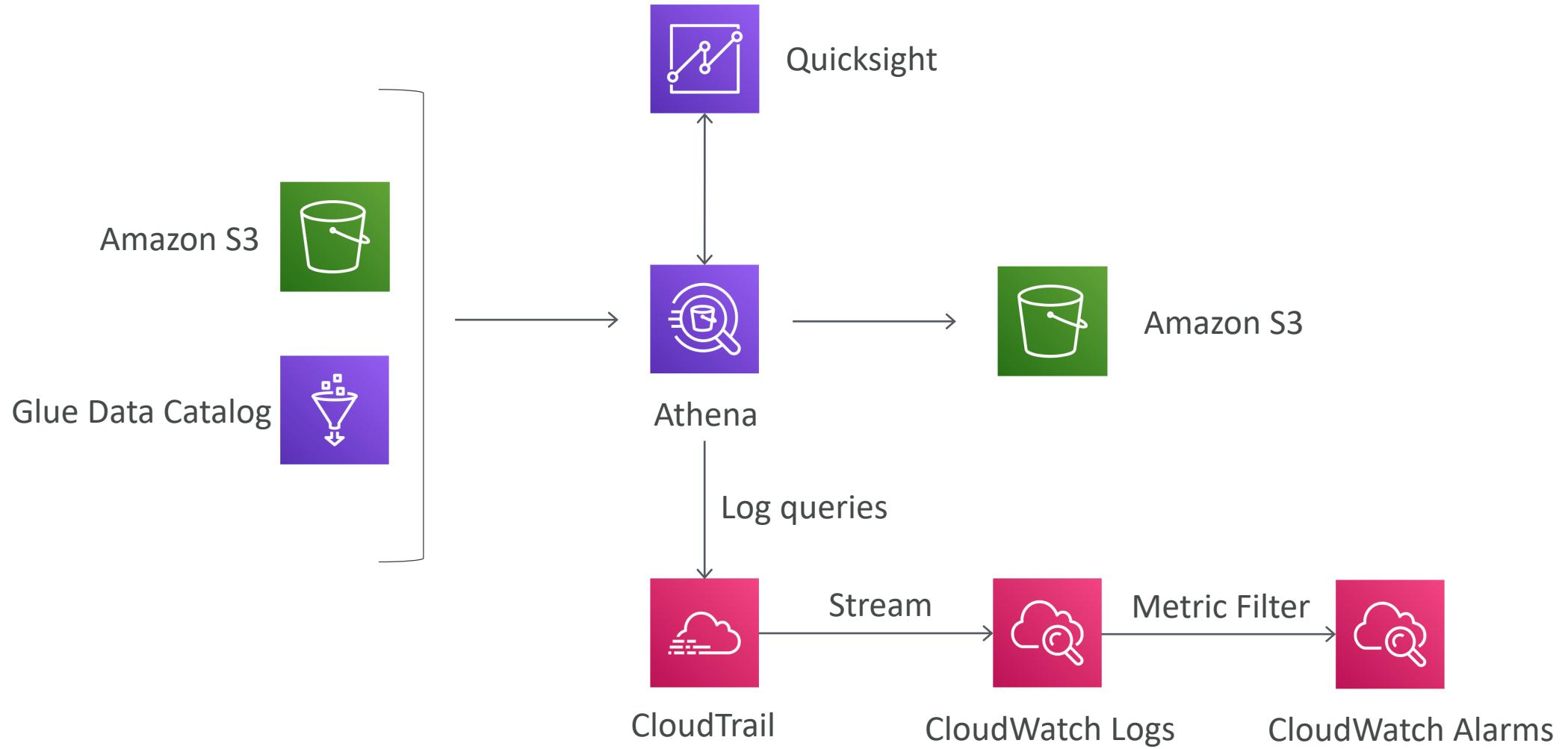
- **Quicksight**

- Business Intelligence tool for data visualization, creating dashboards
- Integrates with Athena, Redshift, EMR, RDS...



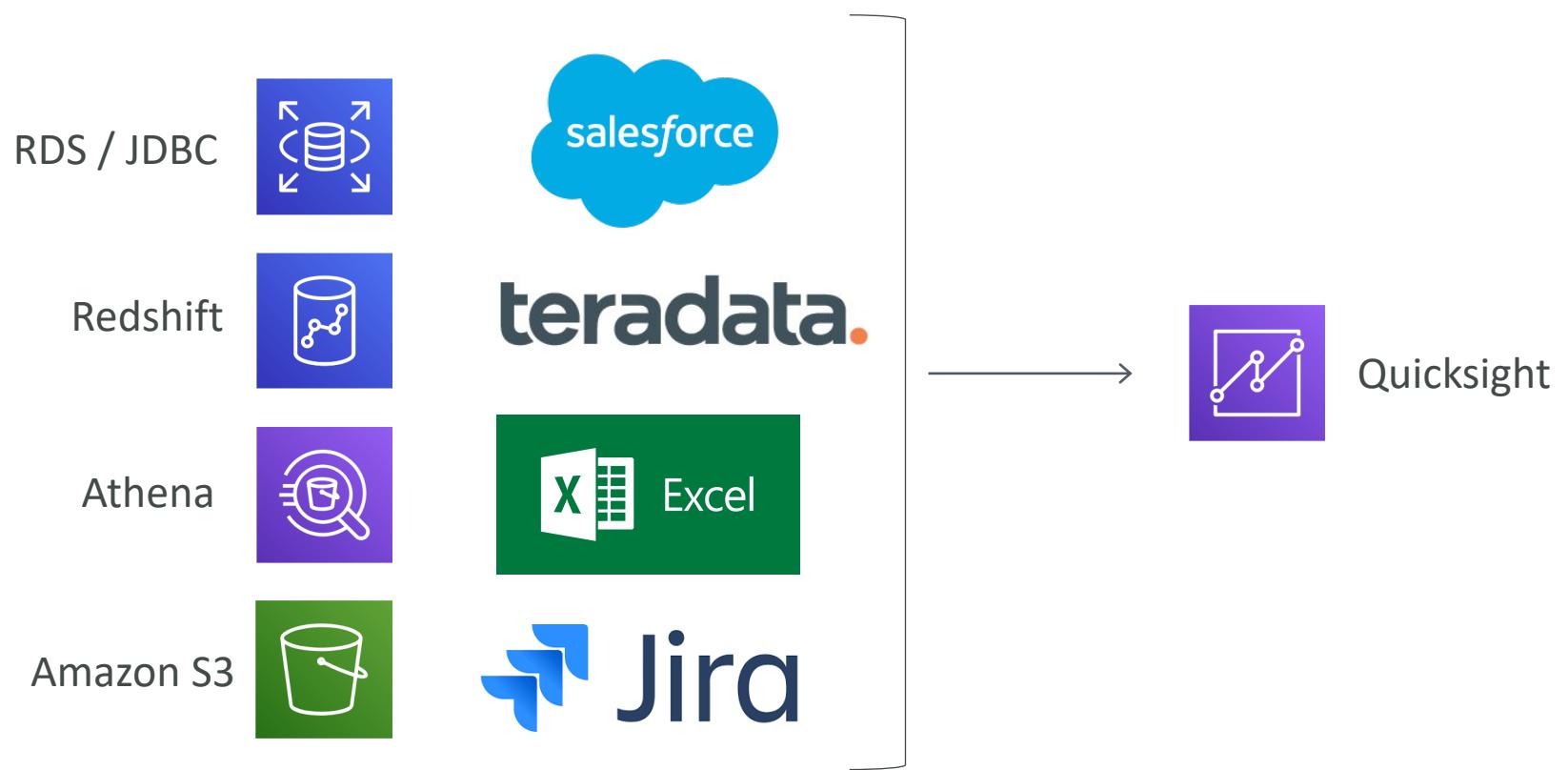
QuickSight

Athena



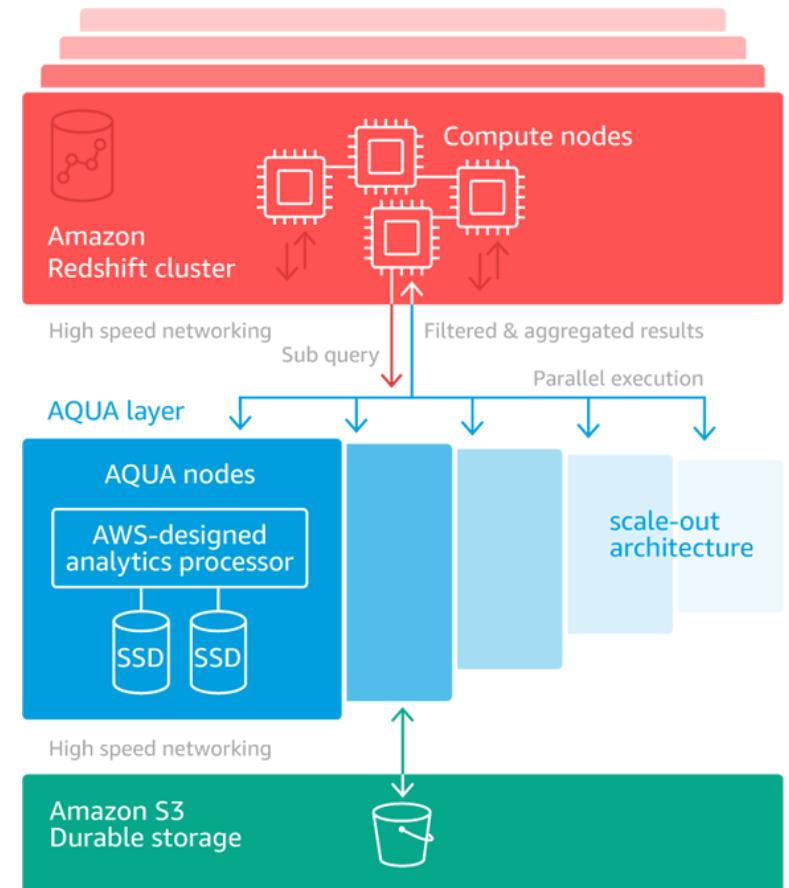
Quicksight

- Can be used to create analytical dashboards using Redshift data



AQUA for Redshift

- AQUA = Advanced Query Accelerator
- It's a new distributed and hardware-accelerated cache
- Runs data intensive tasks (filtering / aggregation) closer to the storage layer
 - Avoids networking bandwidth limitations
 - Other warehouses typically need to move data to compute nodes for processing
- Enables Redshift to run up to 10x faster than any other cloud DWs
- Can process large amounts of data in parallel across multiple nodes
- Automatically scales out to add storage capacity
- No code changes required



https://pages.awscloud.com/AQUA_Preview.html

ElastiCache

In-memory = microsecond latency!

Amazon ElastiCache – Overview

IN-MEMORY



ElastiCache



ElastiCache for
Redis

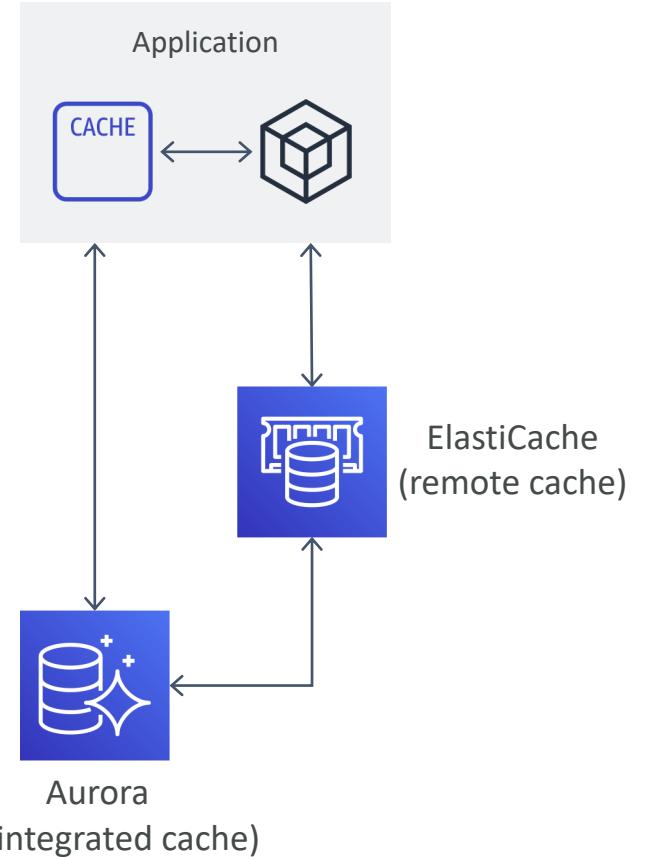


ElastiCache for
Memcached

- Fully-managed **in-memory** data store (caching service, to boost DB read performance)
- It is a **remote caching service**, or a side cache i.e. separate dedicated caching instance
- Provides rapid access to data across distributed nodes
- Two flavors (both are open-source key-value stores)
 - Amazon ElastiCache for Redis
 - Amazon ElastiCache for Memcached
- **Sub-millisecond latency** for real-time applications
- Redis supports complex data types, snapshots, replication, encryption, transactions, pub/sub messaging, transactional Lua scripting, and support for geospatial data
- Multithreaded architecture support with Memcached
- Redis suitable for complex applications including message queues, session caching, leaderboards etc.
- Memcached suitable for relatively simple applications like static website caching

Database caches

- Store frequently accessed data (read operations)
- Improve DB performance by taking the most read load off the DB
- Three types – integrated / local / remote caches
- Database integrated cache (stores data within DB)
 - Typically limited by available memory and resources
 - Example – Integrated cache in Aurora
 - integrated and managed cache with built-in write-through capabilities
 - enabled by default and no code changes needed
- Local cache (stores data within application)
- Remote cache (stores data on dedicated servers)
 - Typically built upon key/value NoSQL stores
 - Example – Redis and Memcached
 - Support up-to a million requests per second per cache node
 - Offer sub-millisecond latency
 - Caching of data and managing its validity is managed by your application



ElastiCache use cases



apps requiring
sub-millisecond latency



Real-time analytics



Gaming leaderboards



Session stores



Chat / messaging apps



Machine learning



Streaming media apps



Geospatial apps

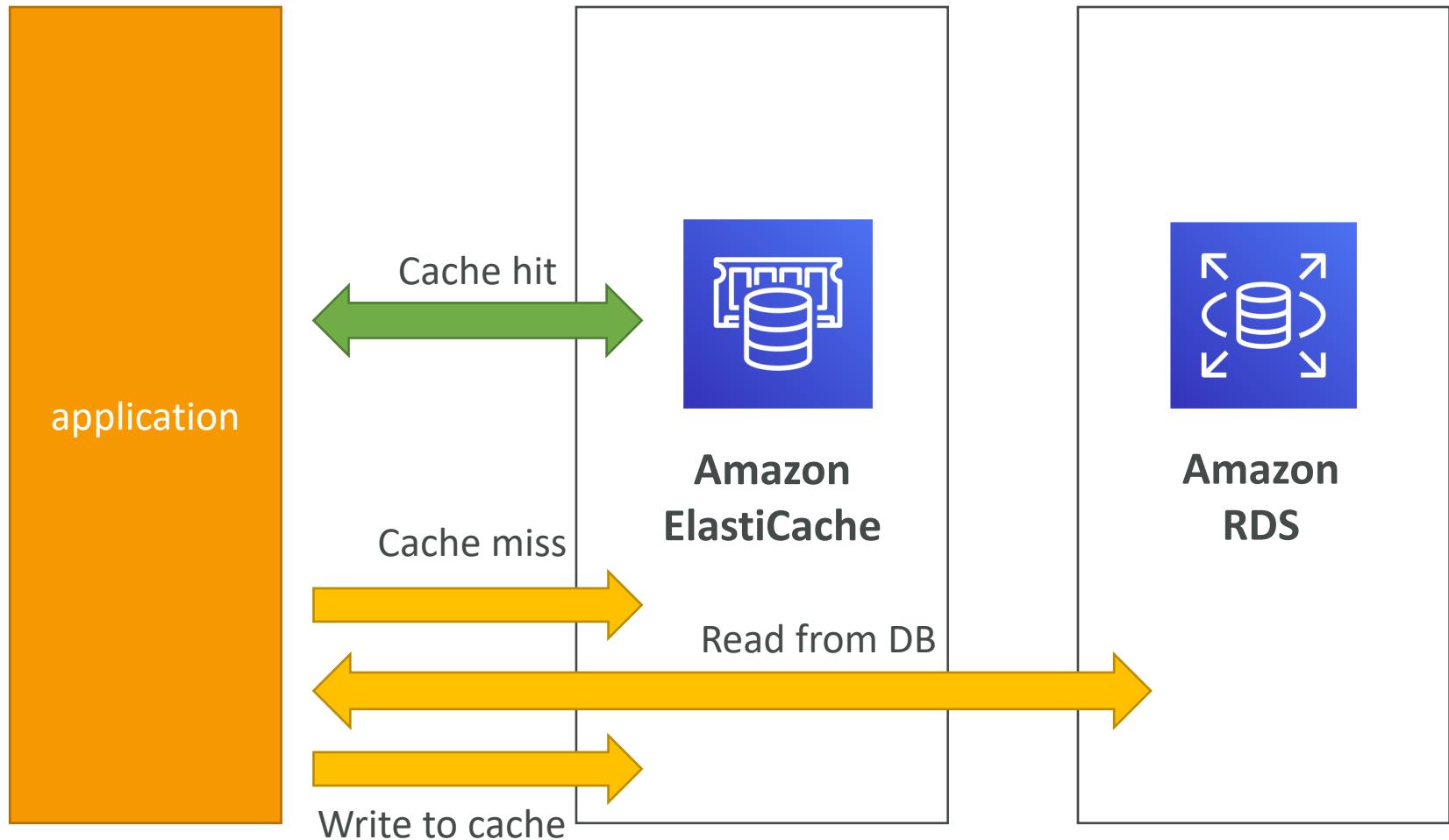


Message queues

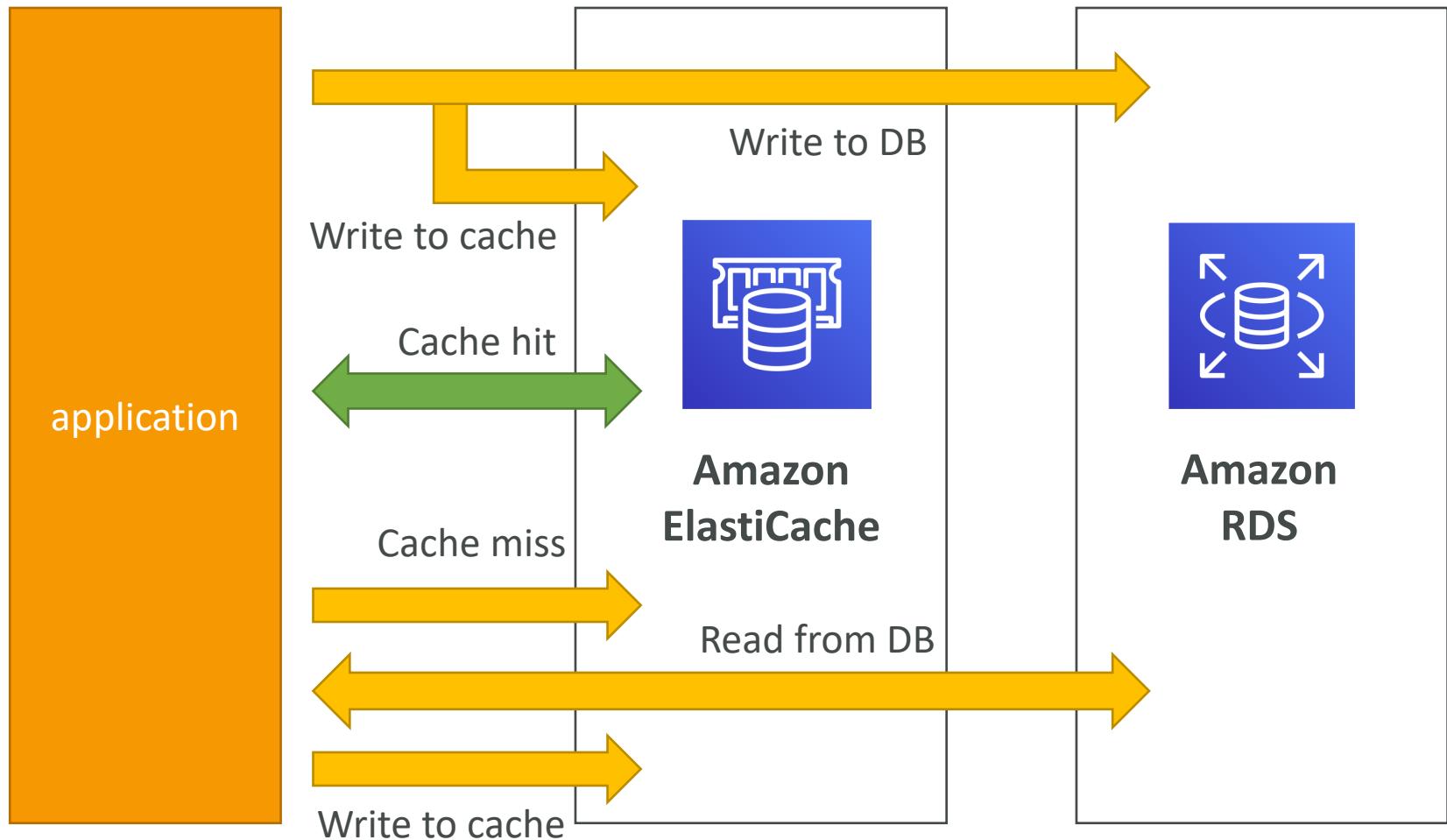
Caching Strategies

- Lazy loading – loads data into the cache only when necessary
 - Reactive approach
 - Only the queried data is cached (small size)
 - There is a cache miss penalty
 - Can contain stale data (use appropriate TTL)
- Write through – loads data into the cache as it gets written to the DB
 - Proactive approach
 - Data is always current (never stale)
 - Results in cache churn (most data is never read, use TTL to save space)
- Lazy loading with write through
 - Get the benefits of both strategies
 - Always use appropriate TTL

Lazy loading illustrated

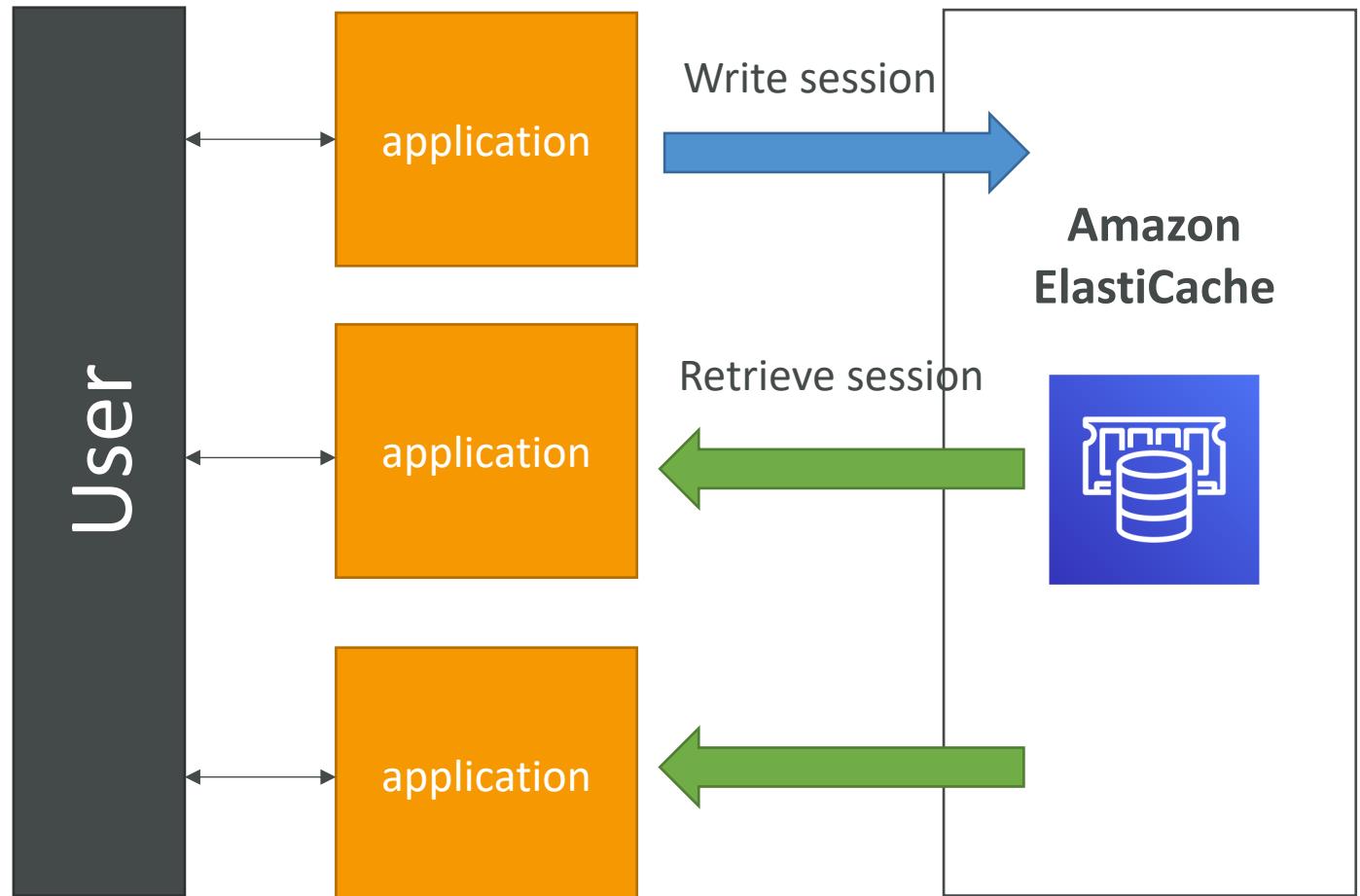


Write through illustrated



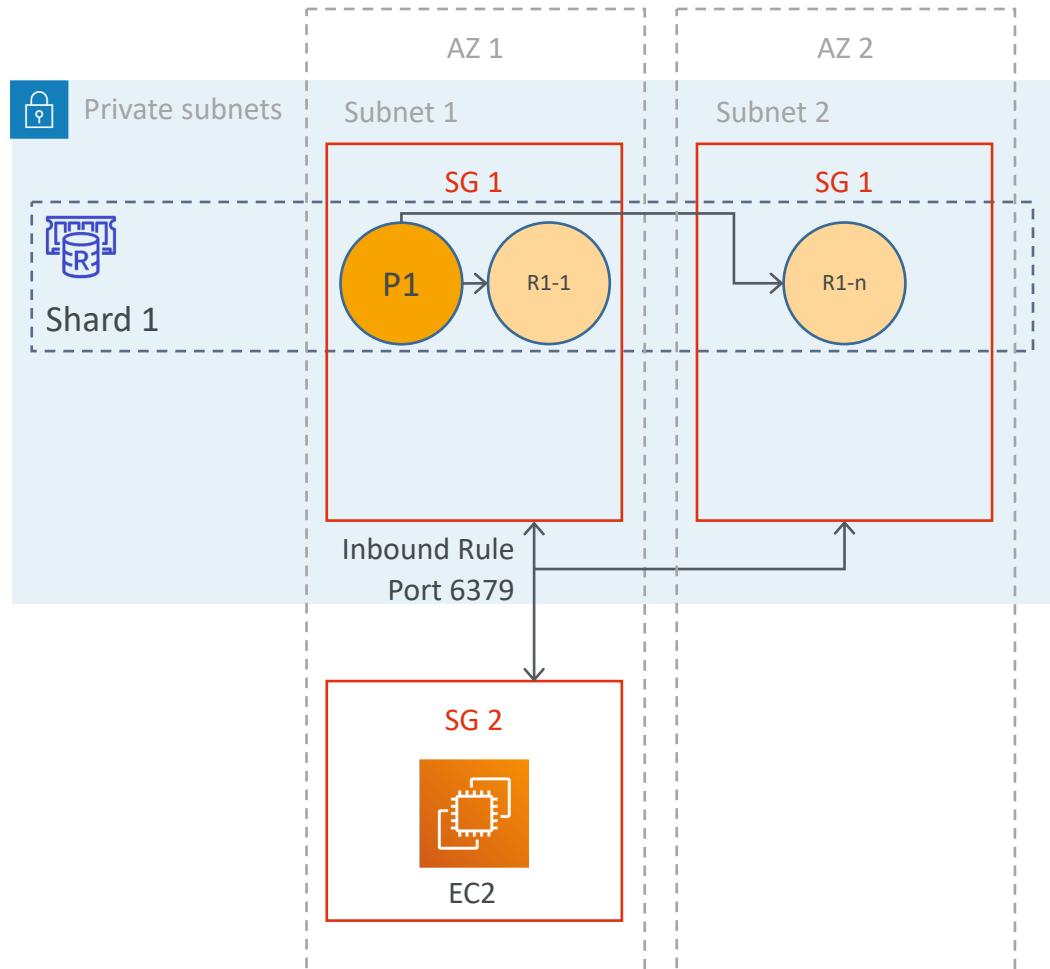
User session store illustrated

- User logs into any of the application
- The application writes the session data into ElastiCache
- The user hits another instance of our application
- The instance retrieves the data and the user is already logged in



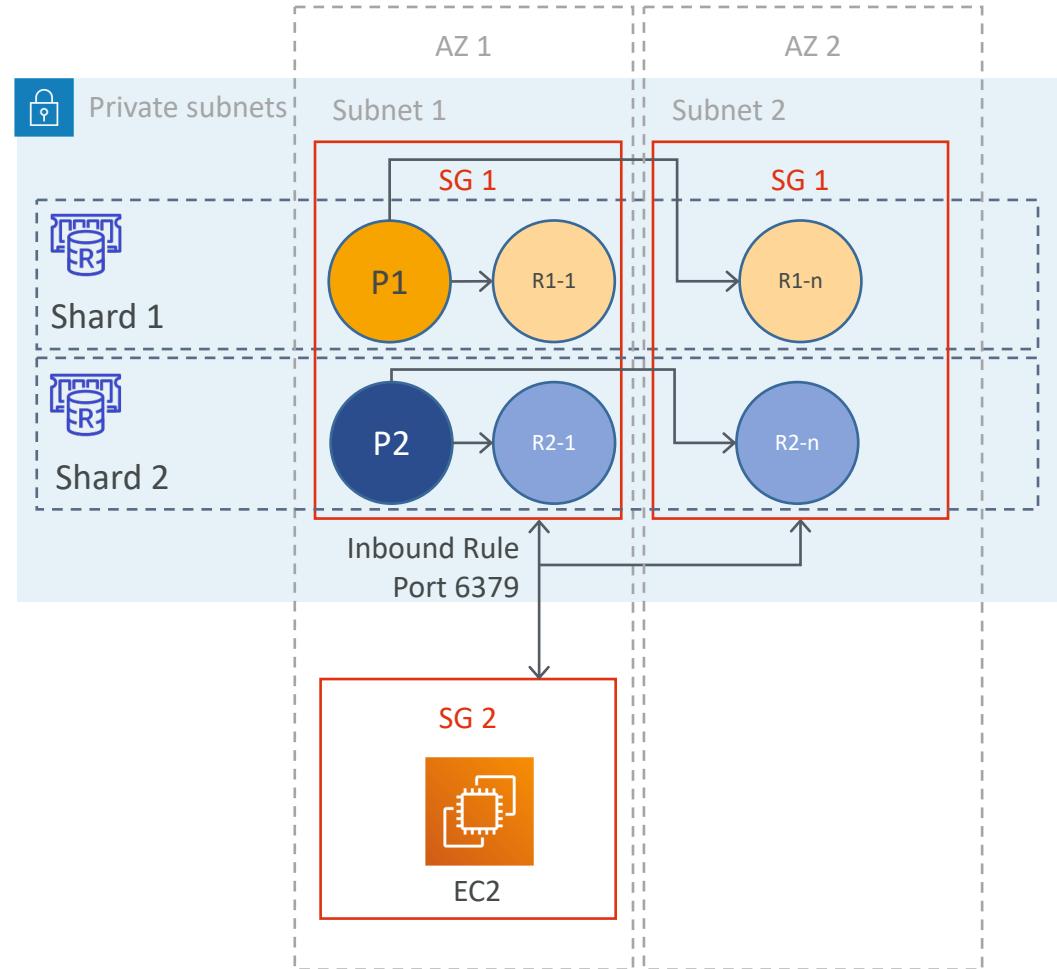
Redis Architecture – Cluster mode disabled

- Redis clusters are generally placed in private subnets
- Accessed from EC2 instance placed in a public subnet in a VPC
- Cluster mode disabled – single shard
- A shard has a primary node and 0-5 replicas
- A shard with replicas is also called as a replication group
- Replicas can be deployed as Multi-AZ
- Multi-AZ replicas support Auto-Failover capability
- Single reader endpoint (auto updates replica endpoint changes)



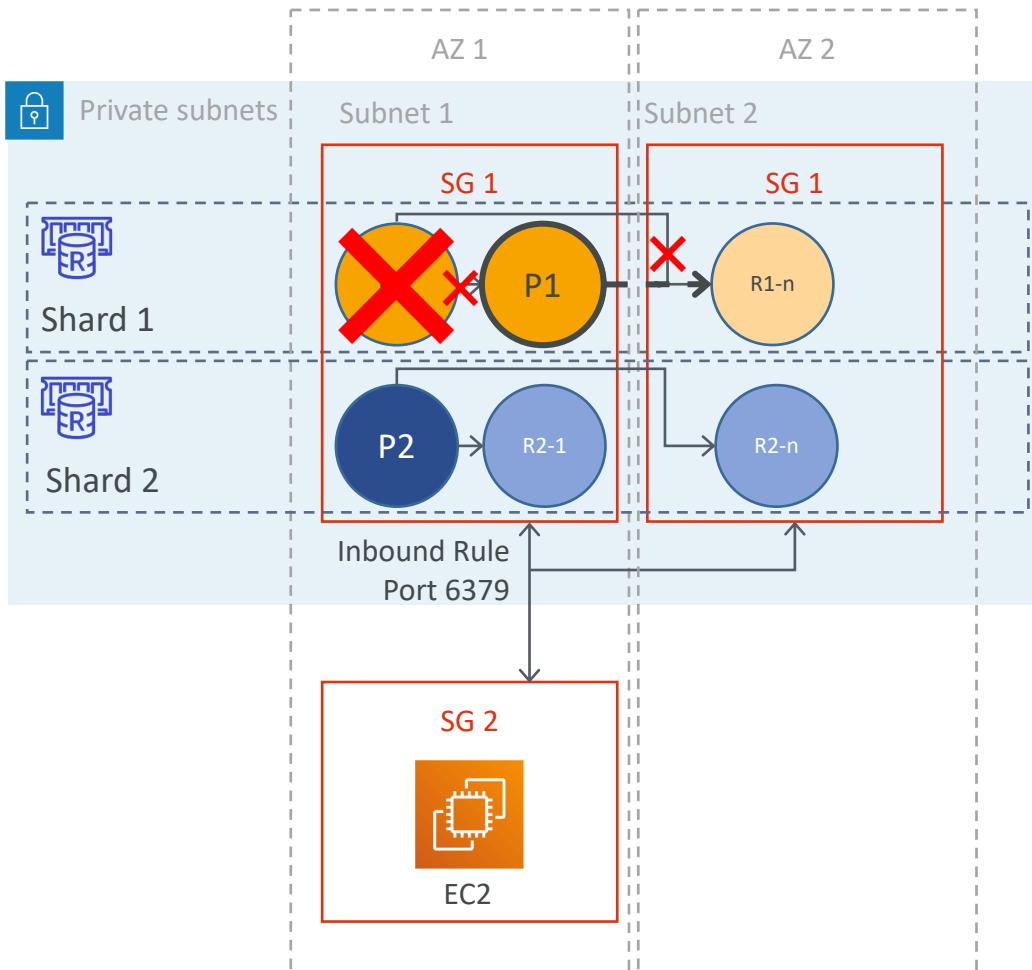
Redis Architecture – Cluster mode enabled

- Cluster mode enabled – multiple shards
- Data is distributed across the available shards
- A shard has a primary node and 0-5 replicas
- Multi-AZ replicas support Auto-Failover capability
- Max 90 nodes per cluster (90 shards w/ no replicas to 15 shards w/ 5 replicas each)
- Minimum 3 shards recommended for HA
- Use nitro system-based node types for higher performance (e.g. M5 / R5 etc)



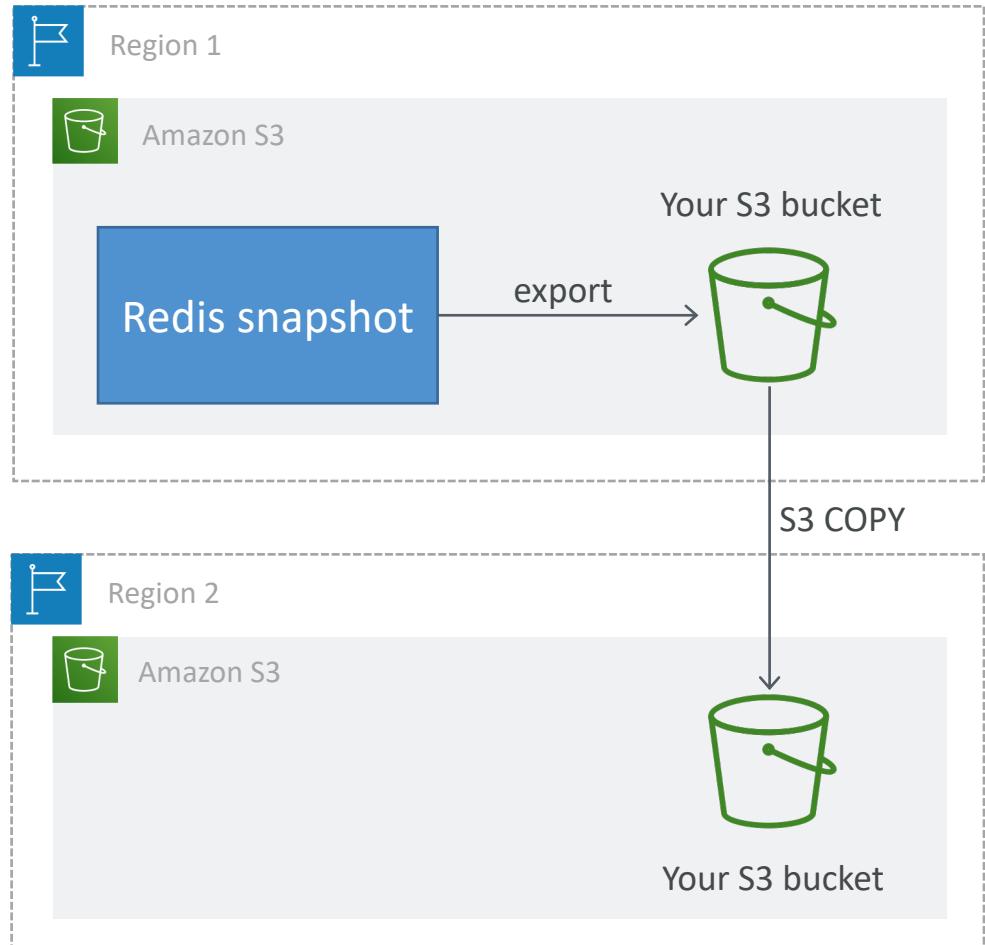
Redis Multi-AZ with Auto-Failover

- Fails over to a replica node on outage
- Minimal downtime (typically 3-6 minutes)
- ASYNC replication (= can have some data loss due to replication lag)
- Manual reboot does not trigger auto-failover (other reboots/failures do)
- You can simulate/test a failover using AWS console / CLI / API
- During planned maintenance for auto-failover enabled clusters
 - If cluster mode enabled – no write interruption
 - If cluster mode disabled – brief write interruption (few seconds)



Redis Backup and Restore

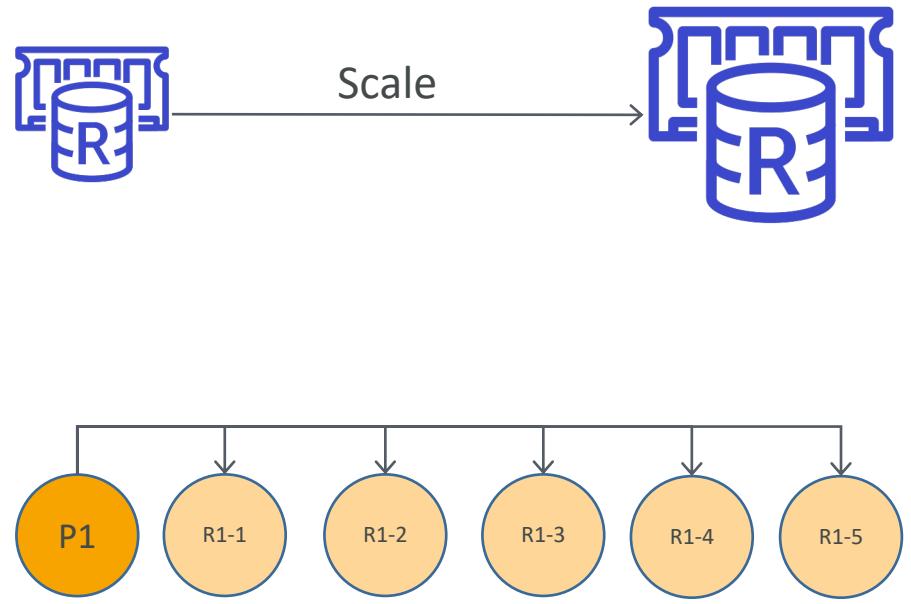
- Supports manual and automatic backups
- Backups are point-in-time copy of the entire Redis cluster, can't backup individual nodes
- Can be used to warm start a new cluster (=preloaded data)
- Can backup from primary node or from replica
- Recommended to backup from a replica (ensures primary node performance)
- Backups (also called snapshots) are stored in S3
- Can export snapshots to your S3 buckets in the same region
- Can then copy the exported snapshot to other region / account using S3 API



Redis Scaling

Cluster Mode Disabled

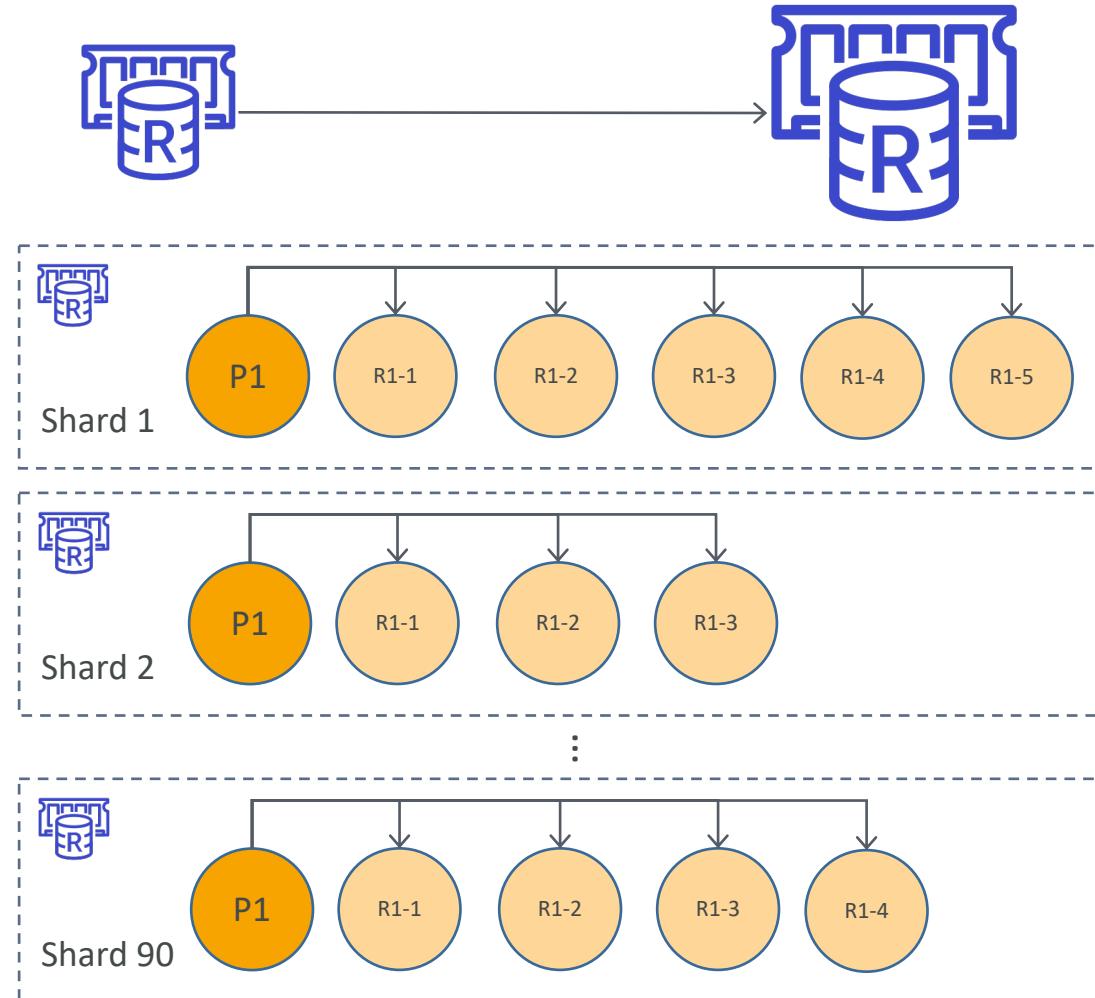
- Vertical Scaling
 - scale up / scale down node type
 - minimal downtime
- Horizontal scaling
 - add / remove replica nodes
 - if Multi-AZ with automatic failover is enabled, you cannot remove the last replica



Redis Scaling

Cluster Mode Enabled

- Vertical Scaling (Online)
 - scale up / scale down node type
 - no downtime
- Horizontal scaling (=resharding and shard rebalancing)
 - allows partitioning across shards
 - add / remove / rebalance shards
 - resharding = change the number of shards as needed
 - shard rebalancing = ensure that data is equally distributed across shards
 - two modes – offline (with downtime) and online (no downtime)



Horizontal scaling – resharding / rebalancing

	Online Mode (=no downtime)	Offline Mode (=downtime)
Cluster availability during scaling op	YES ✓	NO ✗
Can scale out / scale in / rebalance	YES ✓	YES ✓
Can scale up / down (change node type)	NO ✗	YES ✓
Can upgrade engine version	NO ✗	YES ✓
Can specify the number of replica nodes in each shard independently	NO ✗	YES ✓
Can specify the keyspace for shards independently	NO ✗	YES ✓

Redis Replication

Cluster Mode Disabled

- 1 shard
- 0-5 replicas
- If 0 replicas, primary failure = total data loss
- Multi-AZ supported
- Supports scaling
- If primary load is read-heavy, you can scale the cluster (though up to 5 replicas max)

Cluster Mode Enabled

- Up to 90 shards
- 0-5 replicas per shard
- If 0 replicas, primary failure = total data loss in that shard
- Multi-AZ required
- Supports partitioning
- Good for write-heavy nodes (you get additional write endpoints, one per shard)

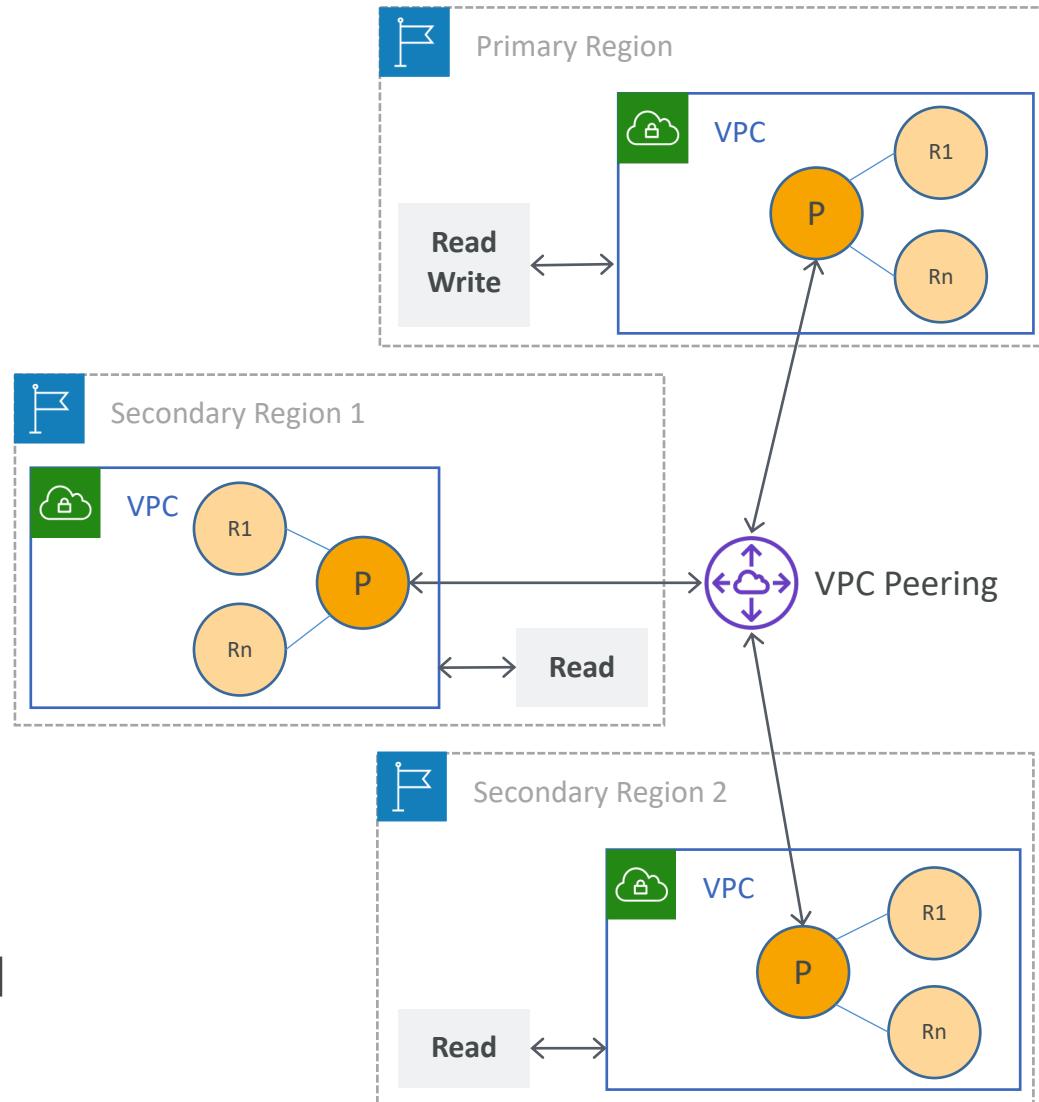
Creating a Redis Cluster



Demo

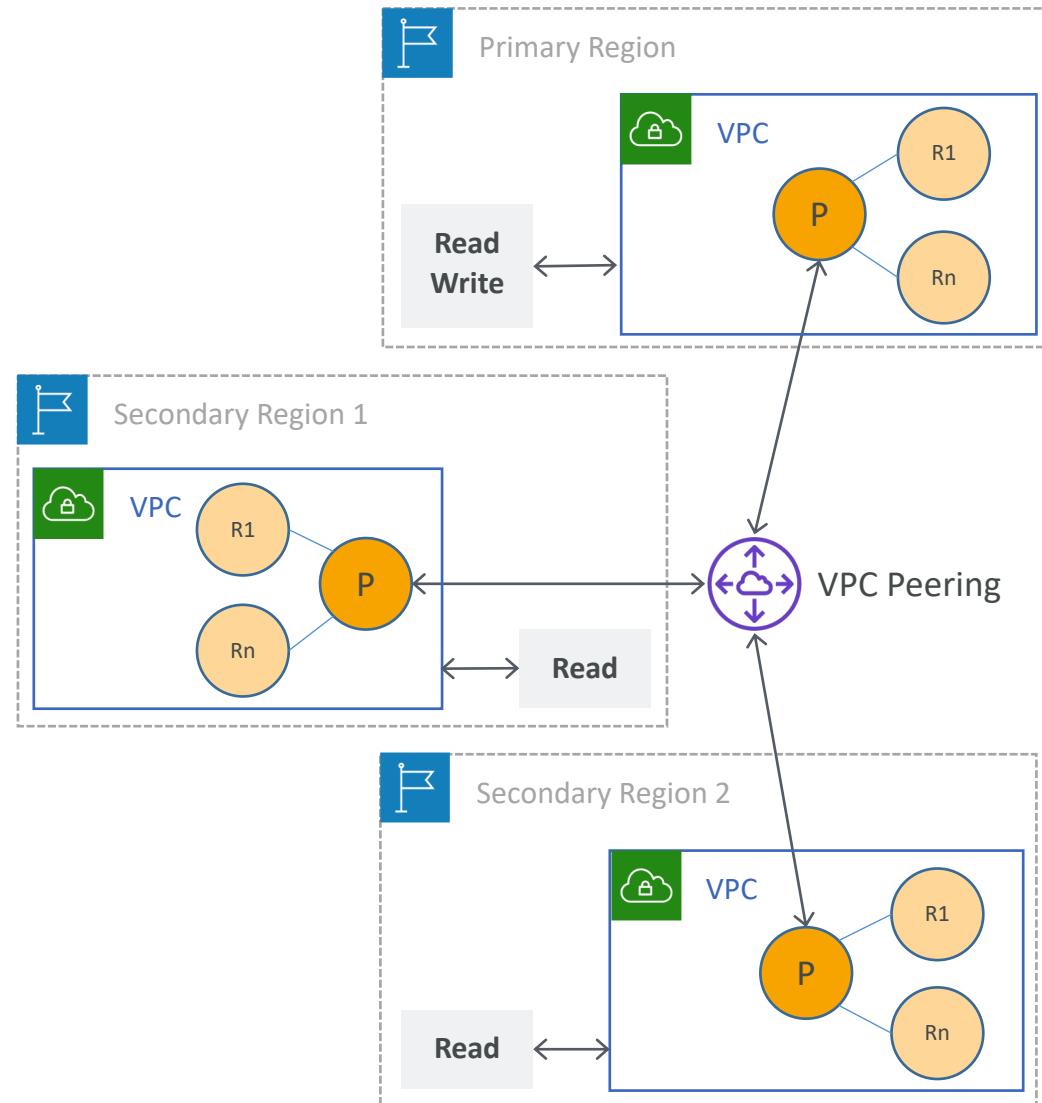
Redis – Global Datastore

- Allows you to create cross region replicas for Redis
- Single writer cluster (primary cluster), multiple reader clusters (secondary clusters)
- Can replicate to up to two other regions
- Improves local latency (bring data closer to your users)
- Provides for DR (you can manually promote a secondary cluster to be primary, not automatic)
- Not available for single node clusters (must convert it to a replication group first)
- Security for cross-region communication is provided through VPC peering



Redis – Global Datastore

- Clusters cannot be modified / resized as usual
 - you scale the clusters by modifying the global datastore
 - all member clusters will get scaled
- To modify a global datastore's parameters
 - modify the parameter group of any member cluster
 - Change gets applied to all member clusters automatically
- Data is replicated cross-region in < 1 sec (typically, not an SLA)
- RPO (typical) < 1sec (amt of data loss due to disaster)
- RTO (typical) < 1min (time taken for DR)



Redis – Good things to know

- Replica lag may grow and shrink over time. If a replica is too far behind the primary, reboot it
- In case of latency/throughput issues, scaling out the cluster helps
- In case of memory pressure, scaling out the cluster helps
- If the cluster is over-scaled, you can scale in to reduce costs
- In case of online scaling
 - cluster remains available, but with some performance degradation
 - level of degradation would depend on CPU utilization and amount of data
- You cannot change Redis cluster mode after creating it (can create a new cluster and warm start it with existing data)
- All nodes within a cluster are of the same instance type



ElastiCache for
Redis

Redis best practices

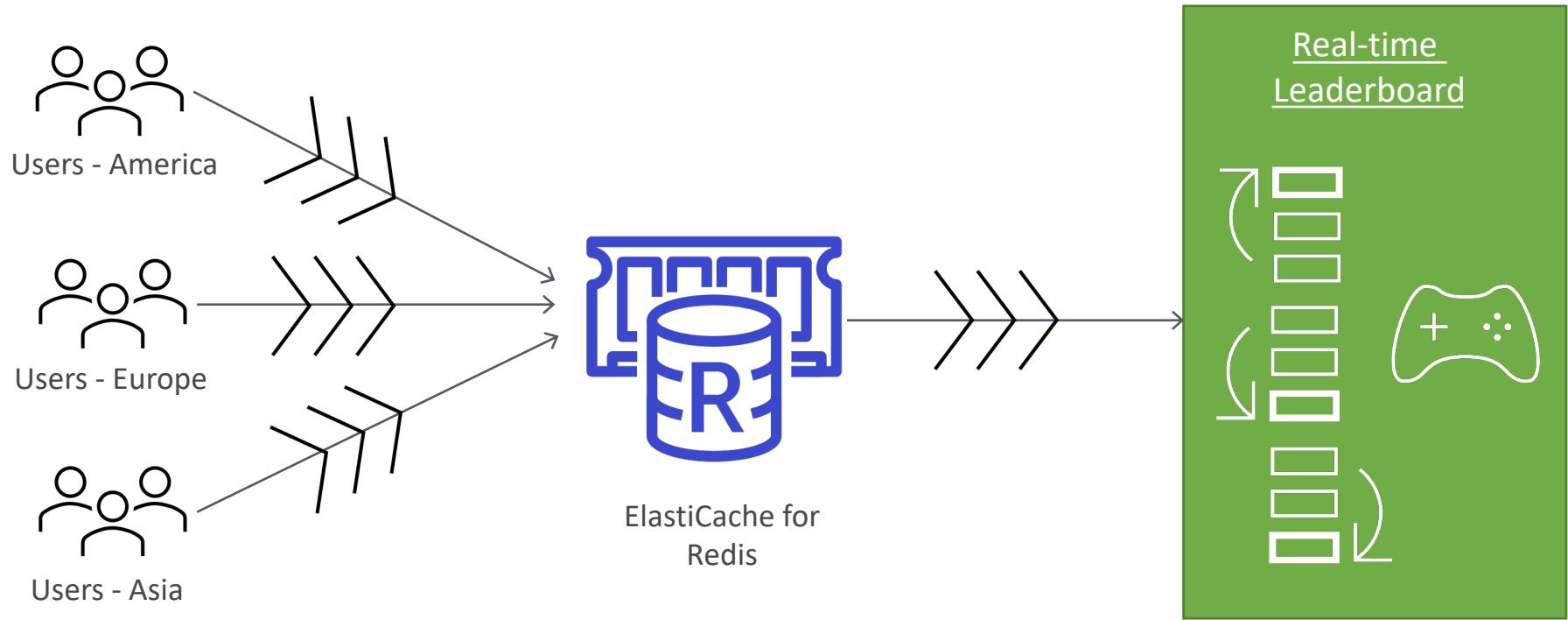
- Cluster mode – connect using the configuration endpoint (allows for auto-discovery of shard and keyspace (slot) mappings)
- Cluster mode disabled – use primary endpoint for writes and reader endpoint for reads (always kept up to date with any cluster changes)
- Set the parameter reserved-memory-percent=25% (for background processes, non-data)
- Keep socket timeout = 1 second (at least)
 - Too low => numerous timeouts on high load
 - Too high => application might take longer to detect connection issues
- Keep DNS caching timeout low (TTL = 5–10 seconds recommended)
- Do not use the “cache forever” option for DNS caching



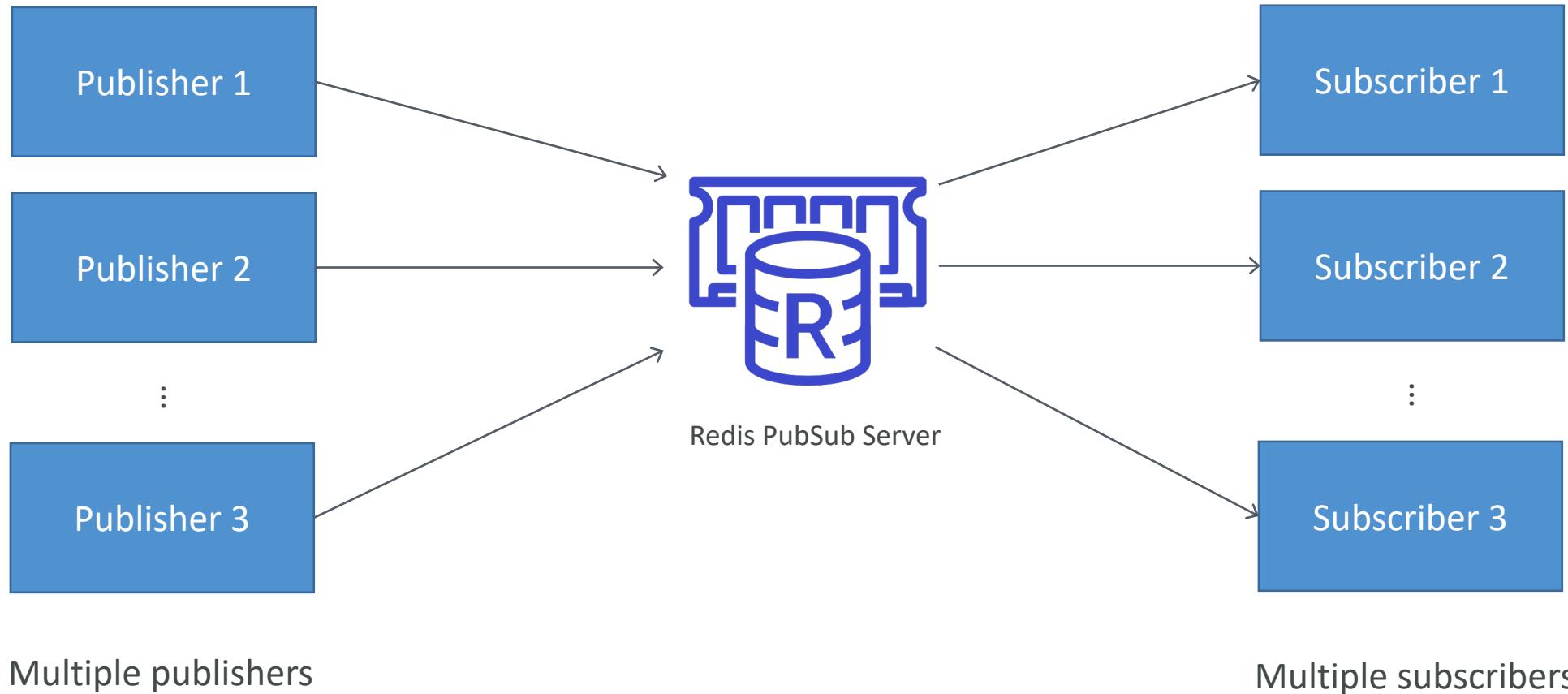
ElastiCache for
Redis

Redis use cases – Gaming Leaderboards

- Uses Redis sorted sets – automatically stores data sorted
- Example – top 10 scores for a game

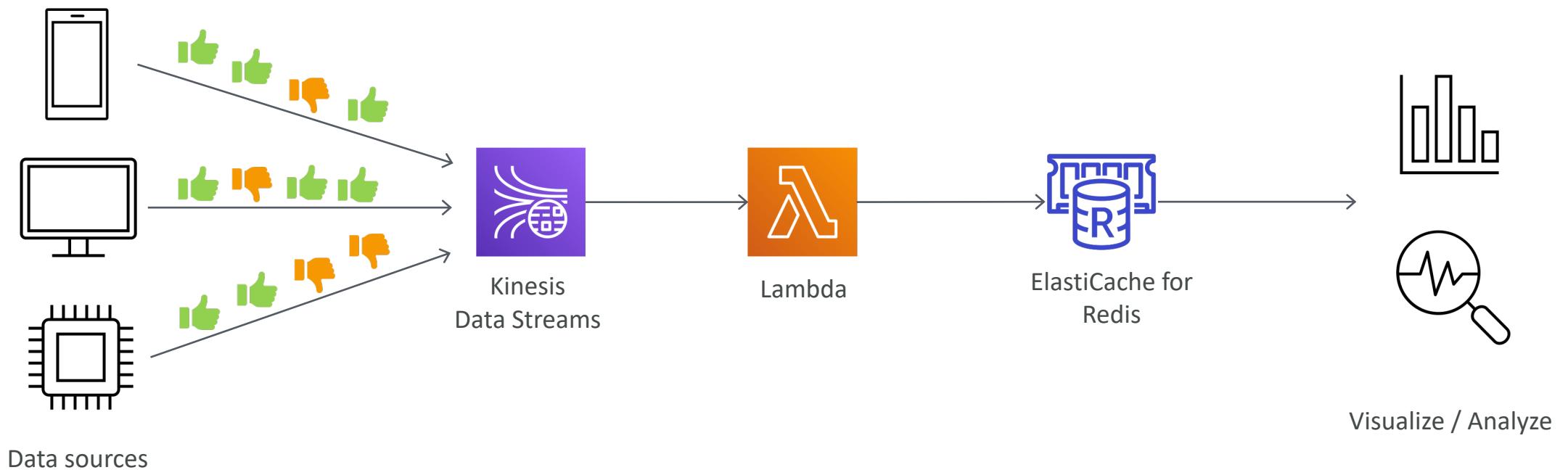


Redis use cases – Pub/sub messaging or queues



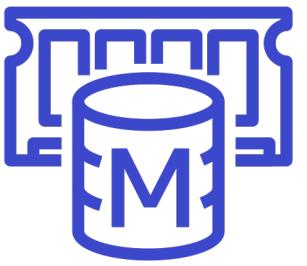
Redis use cases – Recommendation Data

- Uses INCR or DECR in Redis
- Using Redis hashes, you can maintain a list of who liked / disliked a product



Memcached Overview

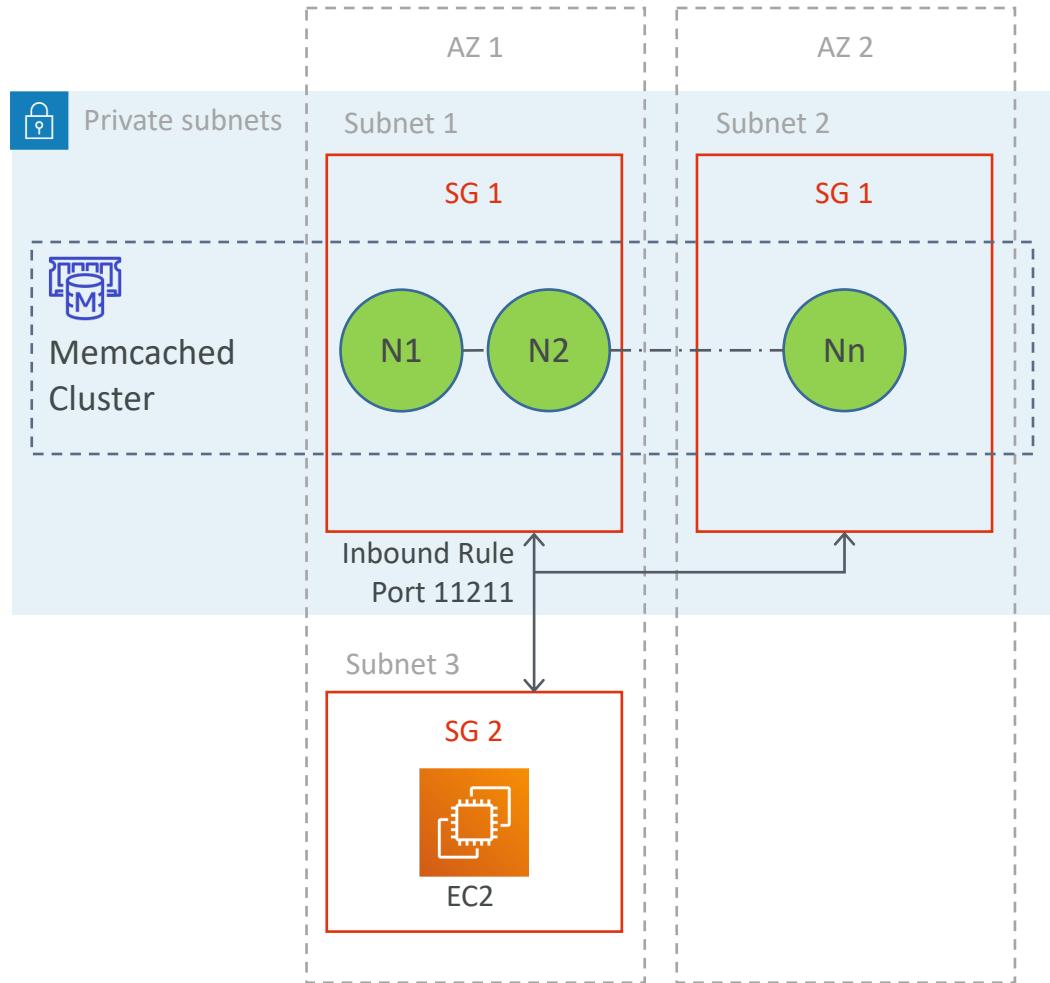
- Simple in-memory key-value store with sub-millisecond latency
- Automatic detection and recovery from cache node failures
- Typical applications
 - Session store (persistent as well as transient session data store)
 - DB query results caching (relational or NoSQL DBs – RDS / DynamoDB etc.)
 - Webpage caching
 - API caching
 - Object caching (images / files / metadata)
- Well suited for web / mobile apps, gaming, IoT, ad-tech, and e-commerce



ElastiCache for
Memcached

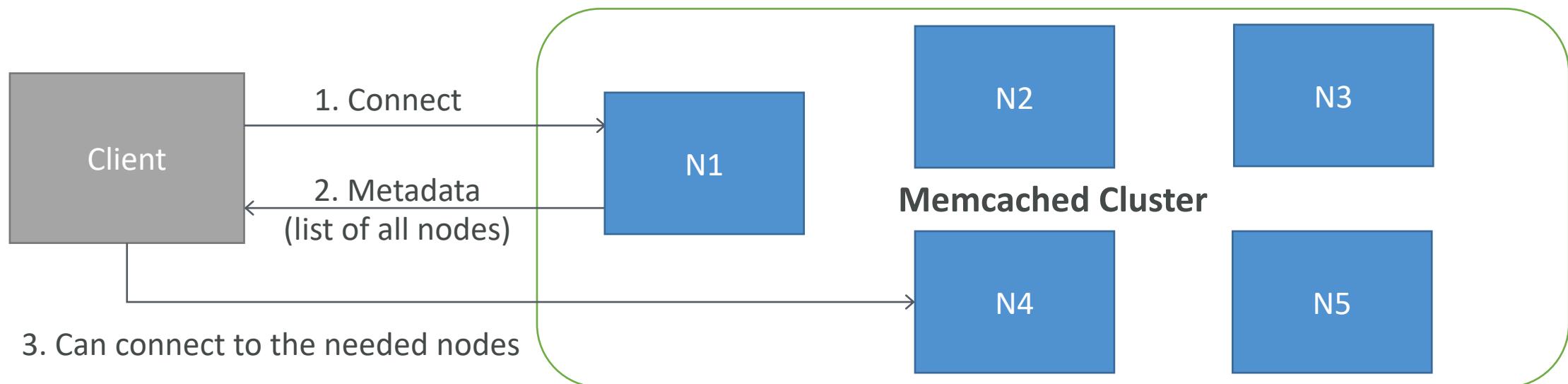
Memcached Architecture

- Memcached cluster is generally placed in private subnet
- Accessed from EC2 instance placed in a public subnet in a VPC
- Allows access only from EC2 network (apps should be hosted on whitelisted EC2 instances)
- Whitelist using security groups
- Up to 20 nodes per cluster
- Data is distributed across the available nodes
- **Replicas are not supported**
- Node failure = data loss
- Nodes can be deployed as Multi-AZ (to reduce data loss)



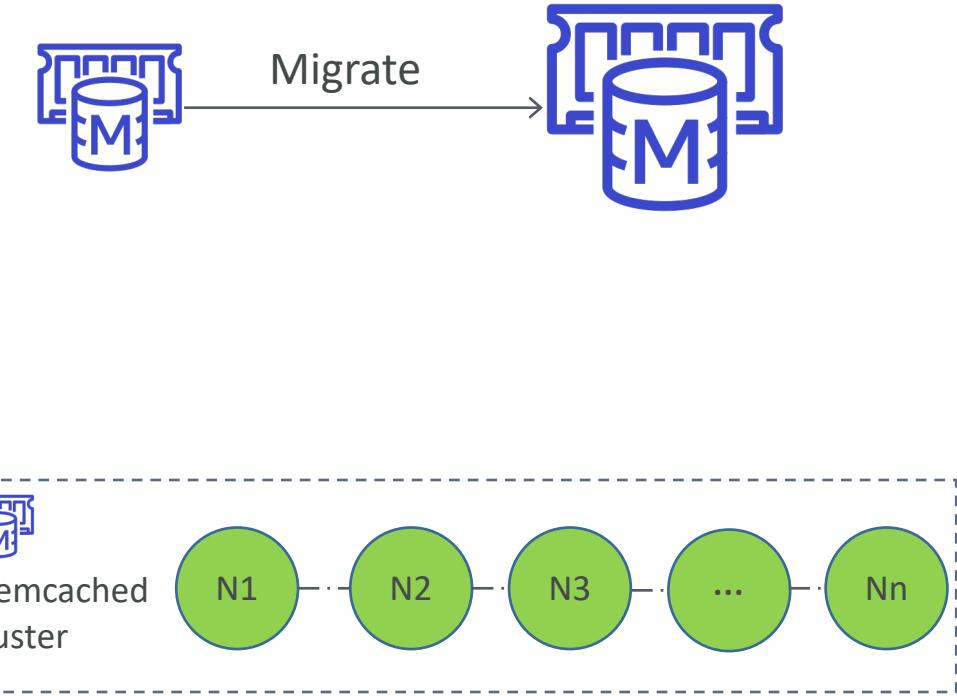
Memcached Auto Discovery

- Allows client to automatically identify nodes in your Memcached cluster
- No need to manually connect to individual nodes
- Simply connect to any one node (using configuration endpoint) and retrieve a list of all other nodes
- The metadata (list of all nodes) gets updated dynamically as you add / remove nodes
- Node failures are automatically detected, and nodes get replaced
- Enabled by default (you must use Auto Discovery capable client)



Memcached Scaling

- Vertical scaling not supported
 - can resize by creating a new cluster and migrating your application
- Horizontal scaling
 - allows you to partition your data across multiple nodes
 - up to 20 nodes per cluster and 100 nodes per region (soft limit)
 - no need to change endpoints post scaling (if you use auto-discovery)
 - must re-map at least some of your keyspace post scaling (evenly spread cache keys across all nodes)



Creating a Memcached Cluster



Demo

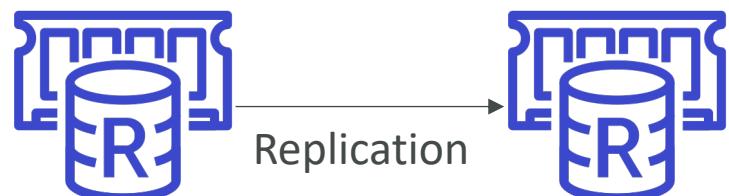


DataCumulus | RIZMAXed

Choosing between Redis and Memcached

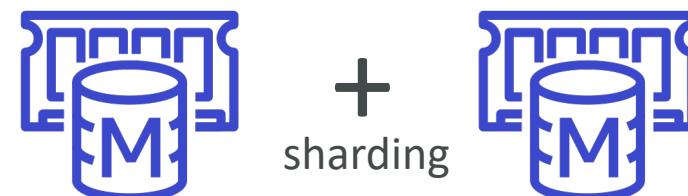
Redis

- Sub-millisecond latency
- Supports complex data types (sorted sets, hashes, bitmaps, hyperloglog, geospatial index)
- Multi AZ with Auto-Failover, supports sharding
- Read Replicas for scalability and HA
- Data Durability using AOF persistence
- Backup and restore features



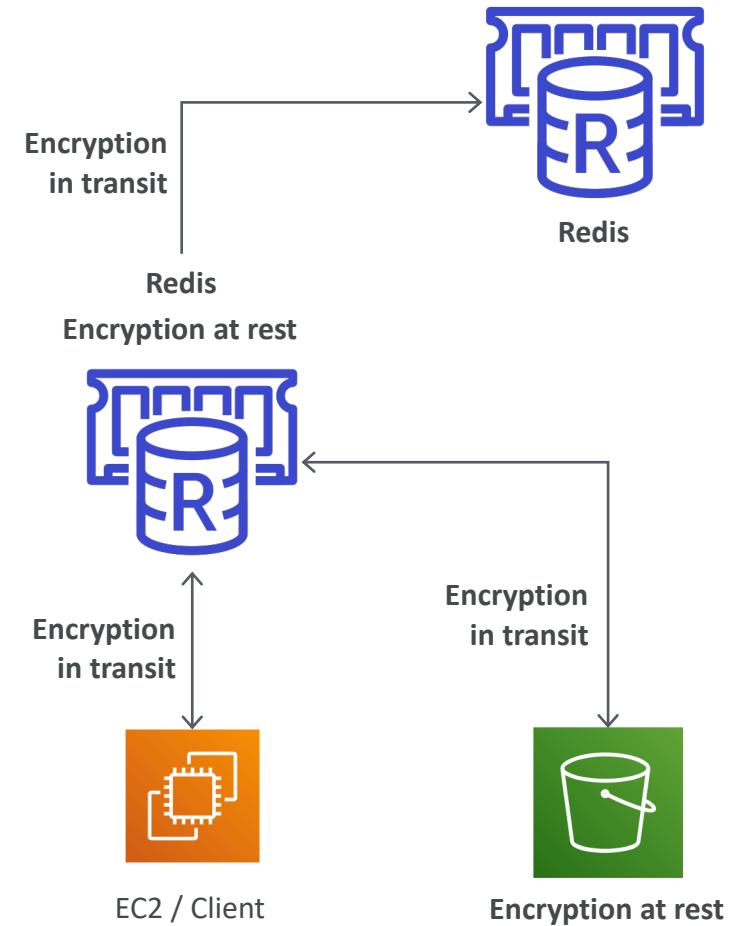
Memcached

- Sub-millisecond latency
- Support only simple data types (string, objects)
- Multi-node for sharding
- Non persistent
- No backup and restore
- Multi-threaded architecture



ElastiCache Security – Encryption

- Memcached does not support encryption
- Encryption at rest for Redis (using KMS)
- Encryption in-transit for Redis (using TLS / SSL)
 - Between server and client
 - Is an optional feature
 - Can have some performance impact
 - Supports encrypted replication
- Redis snapshots in S3 use S3's encryption capabilities



ElastiCache Security – Auth and Access Control

- Authentication into the cache
 - Redis AUTH – server can authenticate the clients (requires SSL/TLS enabled)
 - Server Authentication – clients can authenticate that they are connecting to the right server
- IAM
 - IAM policies can be used for AWS API-level security (create cache, update cache etc.)
 - ElastiCache doesn't support IAM permissions for actions within ElastiCache (which clients can access what)

Enabling Redis AUTH

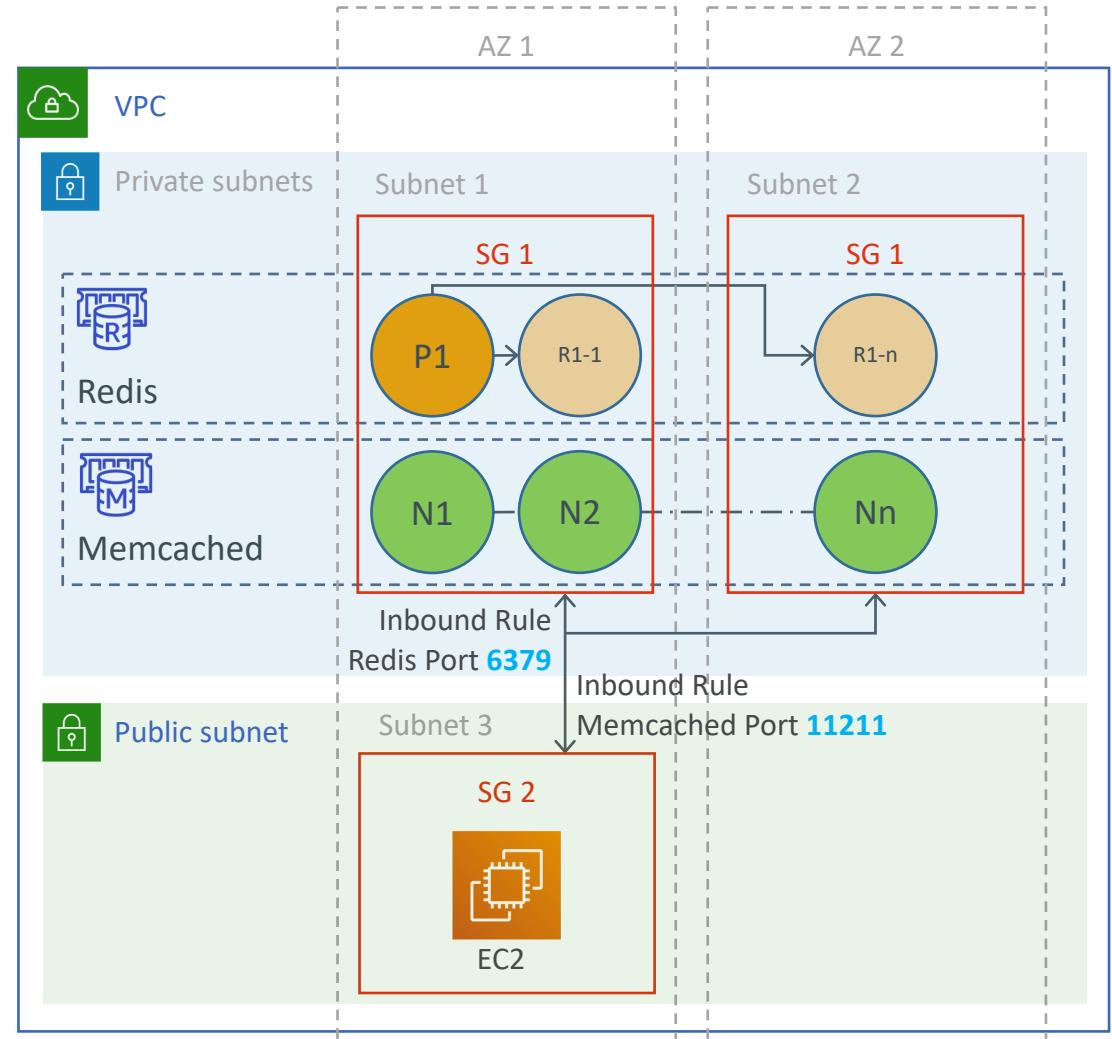
```
aws elasticache
modify-replication-group
--replication-group-id redidgroup
--auth-token This-is-the-password
--auth-token-update-strategy ROTATE
--apply-immediately n
```

Connecting with Redis AUTH

```
src/redis-cli -h <endpoint> -p 6379
-a This-is-the-password
```

ElastiCache Security – Network

- Recommended to use private subnets
- Control network access to ElastiCache through VPC security groups
- ElastiCache Security Groups - allows to control access to ElastiCache clusters running outside Amazon VPC
- For clusters within Amazon VPC, simply use VPC security groups



ElastiCache Logging and Monitoring

- Integrated with CloudWatch
 - Host level metrics – CPU / memory / network
 - Redis metrics – replication lag / engine CPU utilization / metrics from Redis INFO command
 - 60-second granularity
- ElastiCache Events
 - Integrated with SNS
 - Log of events related to cluster instances / SGs / PGs
 - Available within ElastiCache console
- API calls logged with CloudTrail



CloudWatch



SNS



CloudTrail

ElastiCache Pricing

- Priced per node-hour consumed for each node type
- Partial node-hours consumed are billed as full hours
- Can use reserved nodes for upfront discounts (1-3 year terms)
- Data transfer
 - No charge for data transfer between EC2 and ElastiCache within AZ
 - All other data transfer chargeable
- Backup storage
 - For automated and manual snapshots (per GB per month)
 - Space for one snapshot is complimentary for each active Redis cluster



ElastiCache

DocumentDB

MongoDB in the Cloud. Well, almost!

Amazon DocumentDB – Overview

DOCUMENT



DocumentDB

- Fully-managed (non-relational) document database for MongoDB workloads
- JSON documents (nested key-value pairs) stored in collections (\approx tables)
- Compatible w/ majority of MongoDB applications, drivers, and tools
- High performance, scalability, and availability
- Support for flexible indexing, powerful ad-hoc queries, and analytics
- Storage and compute can scale independently
- Supports 15 low-latency read replicas (Multi-AZ)
- Auto scaling of storage from 10 GB to 64 TB
- Fault-tolerant and self-healing storage
- Automatic, continuous, incremental backups and PITR

Document Database

- Stores JSON documents (semi-structured data)
- Key-value pairs can be nested

Relational DB (SQL)	DocumentDB (MongoDB)
Table	Collection
Rows	Documents
Columns	Fields
Primary key	Object ID
Nested table / object	Embedded documents
Index / view / array	Index / view / array

```
{
  "trace_id": "daeae4ef-5495-4643-81ca-911d0615d5b8",
  "request_id": "daeae4ef-5495-4643-81ca-911d0615d5b8",
  "request_type": "NEW",
  "request_timestamp": 1581681322024,
  "request_status": "queued"
},
{
  "trace_id": "fcdse4ef-4565-4677-888a-876d0615d888",
  "request_id": "fcdse4ef-4565-4677-888a-876d0615d888",
  "request_type": "NEW",
  "requestor" : {
    "name": "Jimmy Brown",
    "contact" : {
      "email": "jb@xyz.com",
      "phone": "+1234567890"
    }
  },
  "request_timestamp": 1581681323182,
  "request_status": "processing",
  "message": "Your request is being processed.",
  "assigned_to": "Jason Root"
}
```

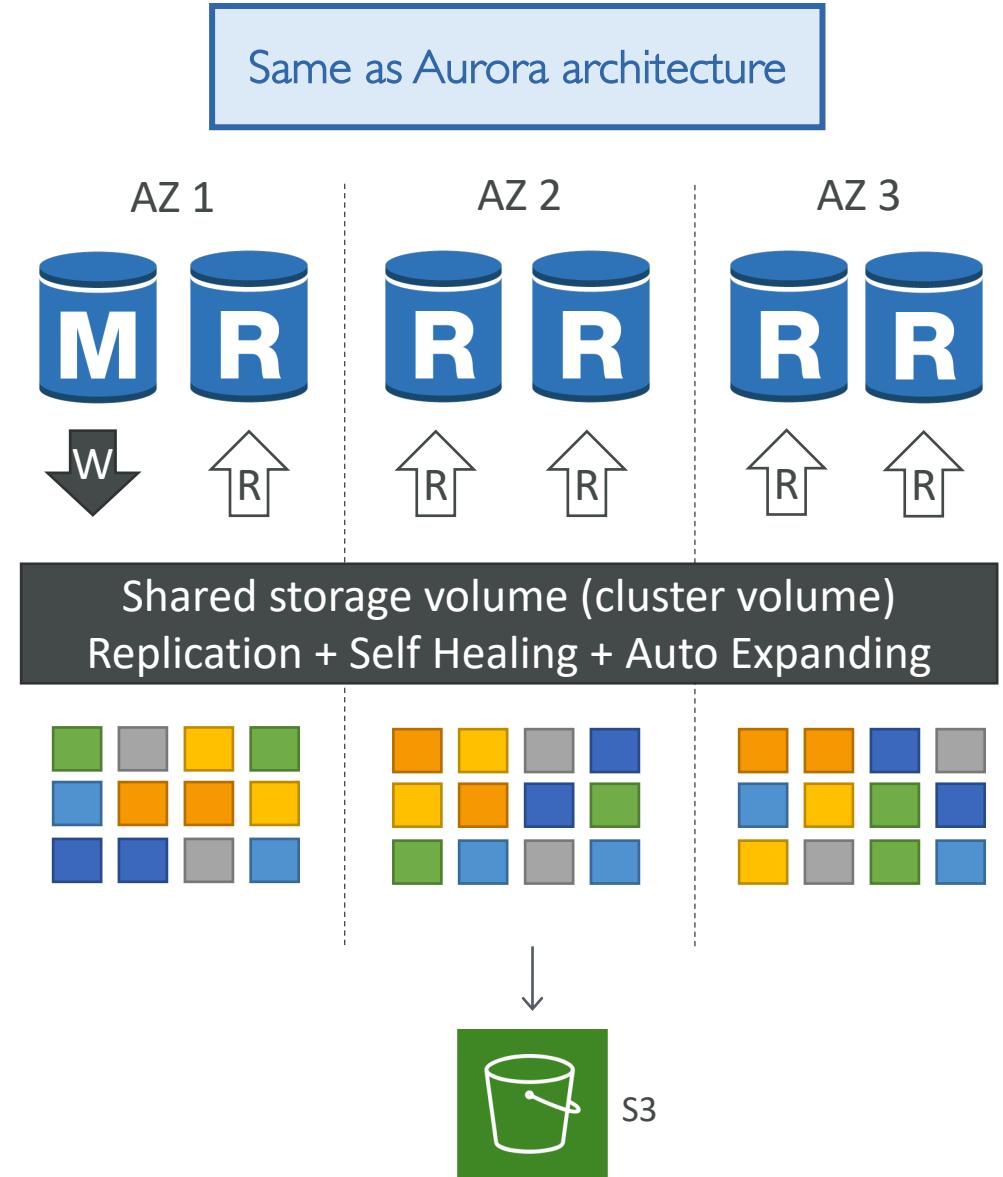
Why document database?

- JSON is the de-facto format for data exchange
- DocumentDB makes it easy to insert, query, index, and perform aggregations over JSON data
- Store JSON output from APIs straight into DB and start analysing it
- flexible document model, data types, and indexing
- Add / remove indexes easily
- Run ad hoc queries for operational and analytics workloads
- for known access patterns – use DynamoDB instead

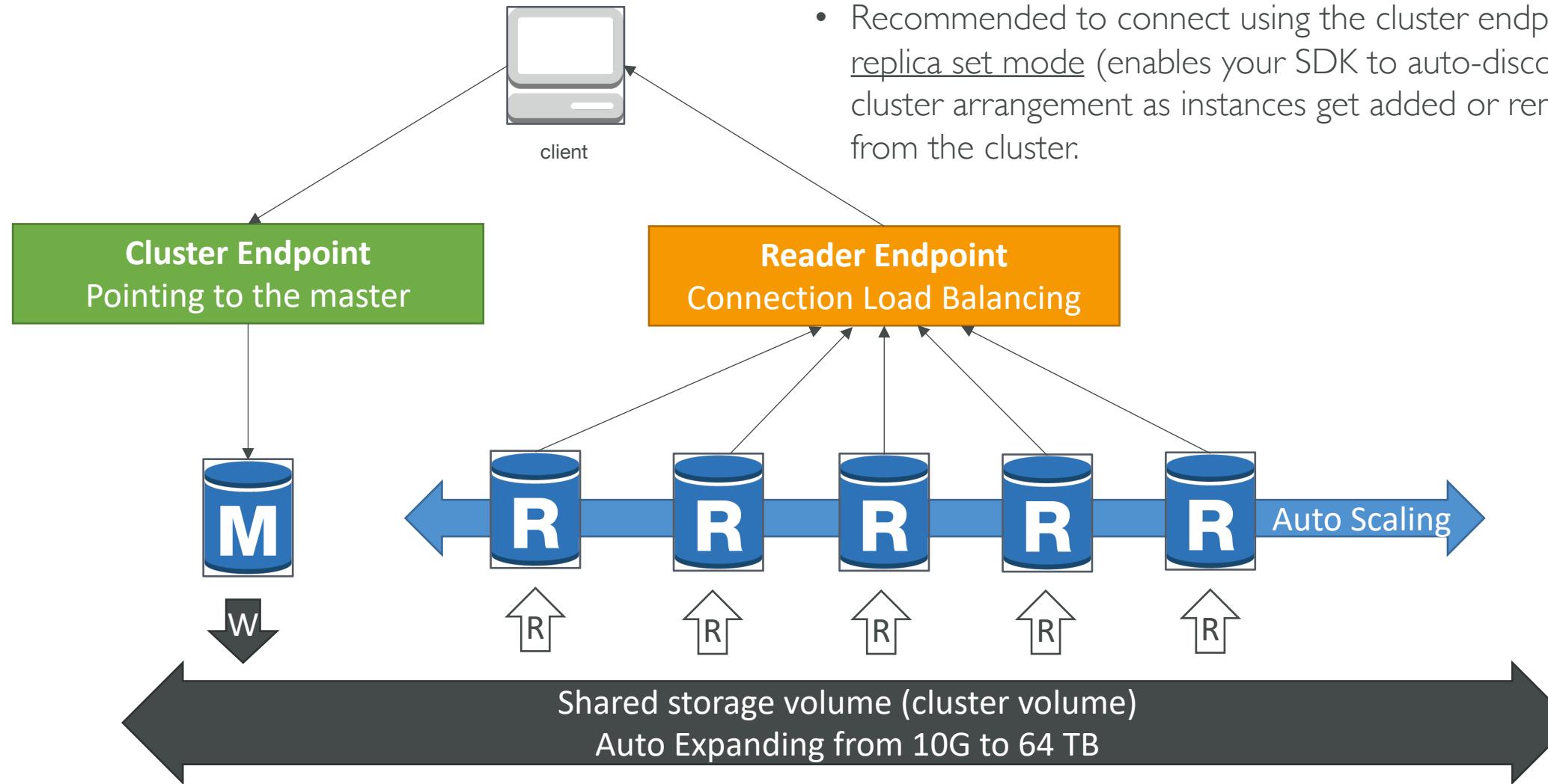
```
{  
  "trace_id": "daeae4ef-5495-4643-81ca-911d0615d5b8",  
  "request_id": "daeae4ef-5495-4643-81ca-911d0615d5b8",  
  "request_type": "NEW",  
  "request_timestamp": 1581681322024,  
  "request_status": "queued"  
},  
{  
  "trace_id": "fcdse4ef-4565-4677-888a-876d0615d888",  
  "request_id": "fcdse4ef-4565-4677-888a-876d0615d888",  
  "request_type": "NEW",  
  "requestor" : {  
    "name": "Jimmy Brown",  
    "contact" : {  
      "email": "jb@xyz.com",  
      "phone": "+1234567890"  
    }  
  },  
  "request_timestamp": 1581681323182,  
  "request_status": "processing",  
  "message": "Your request is being processed.",  
  "assigned_to": "Jason Root"  
}
```

DocumentDB Architecture

- 6 copies of your data across 3 AZ (distributed design)
 - Lock-free optimistic algorithm (quorum model)
 - 4 copies out of 6 needed for writes (4/6 write quorum - data considered durable when at least 4/6 copies acknowledge the write)
 - 3 copies out of 6 needed for reads (3/6 read quorum)
 - Self healing with peer-to-peer replication, Storage is striped across 100s of volumes
- One DocumentDB Instance takes writes (master)
- Compute nodes on replicas do not need to write/replicate (=improved read performance)
- Log-structured distributed storage layer – passes incremental log records from compute to storage layer (=faster)
- Master + up to 15 Read Replicas serve reads
- Data is continuously backed up to S3 in real time, using storage nodes (compute node performance is unaffected)

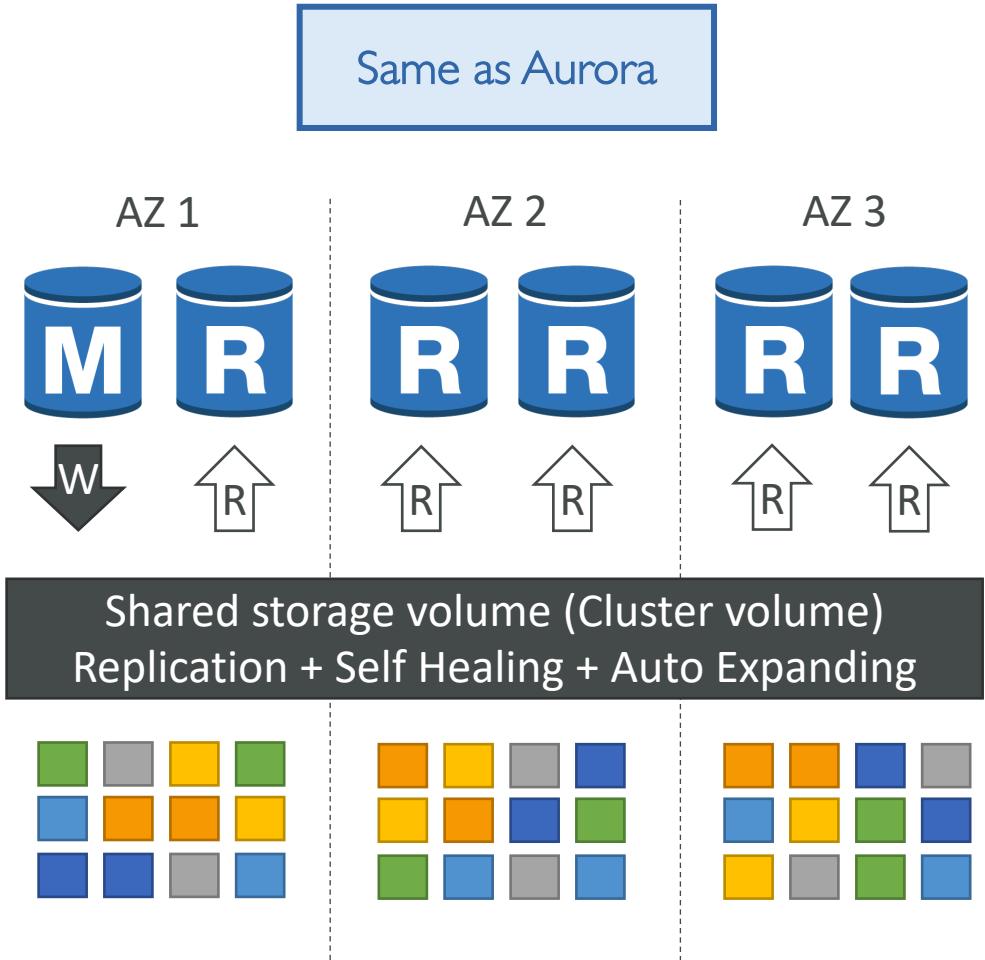


DocumentDB Cluster



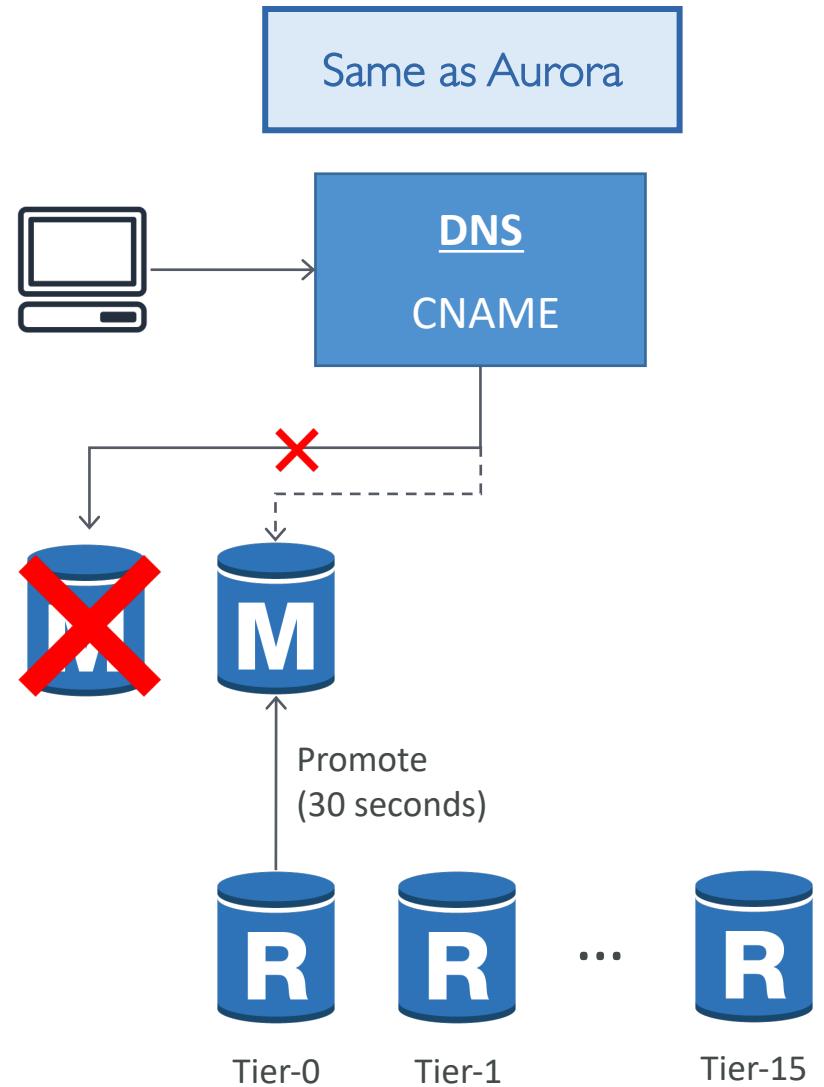
DocumentDB Replication

- Up to 15 read replicas
- ASYNC replication
- Replicas share the same underlying storage layer
- Typically take 10s of milliseconds (replication lag)
- Minimal performance impact on the primary due to replication process
- Replicas double up as failover targets (standby instance is not needed)



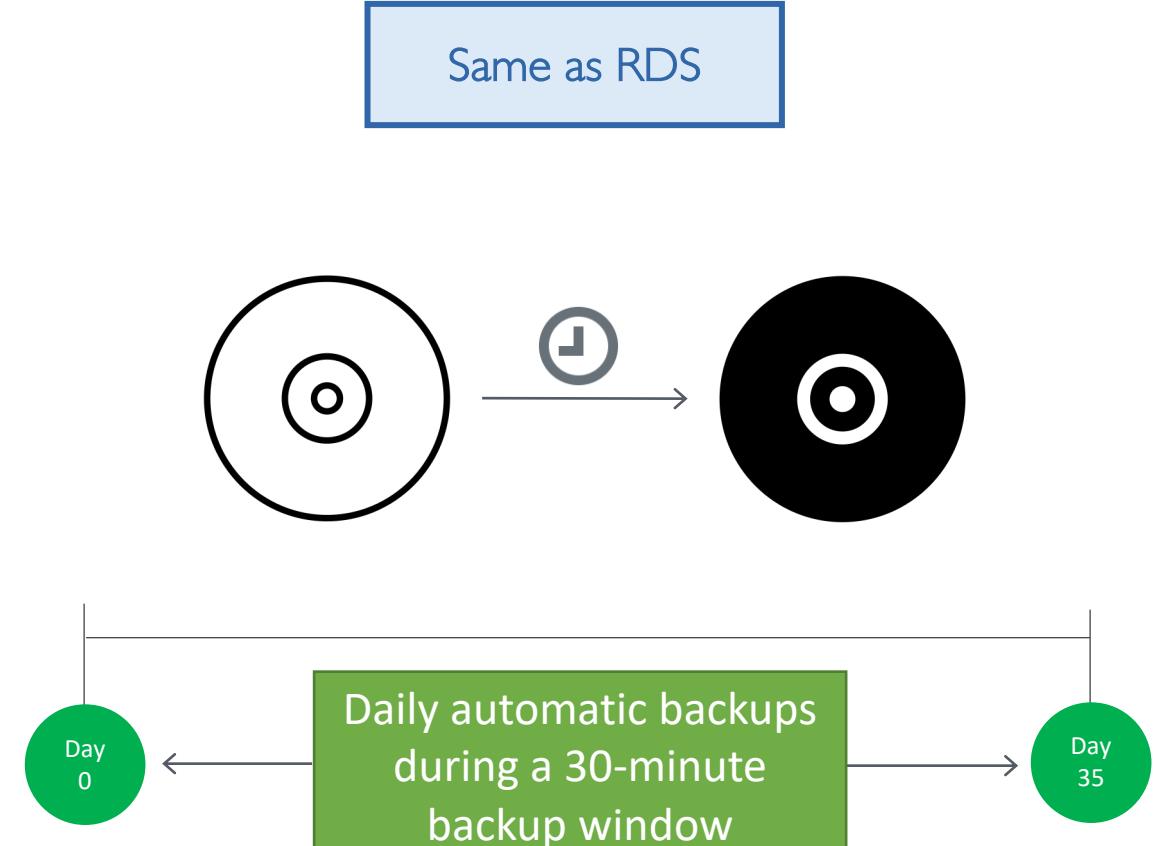
DocumentDB HA failovers

- Failovers occur automatically
- A replica is automatically promoted to be the new primary during DR
- DocumentDB flips the CNAME of the DB instance to point to the replica and promotes it
- Failover to a replica typically takes 30 seconds (minimal downtime)
- Creating a new instance takes about 8-10 minutes (post failover)
- Failover to a new instance happens on a best-effort basis and can take longer



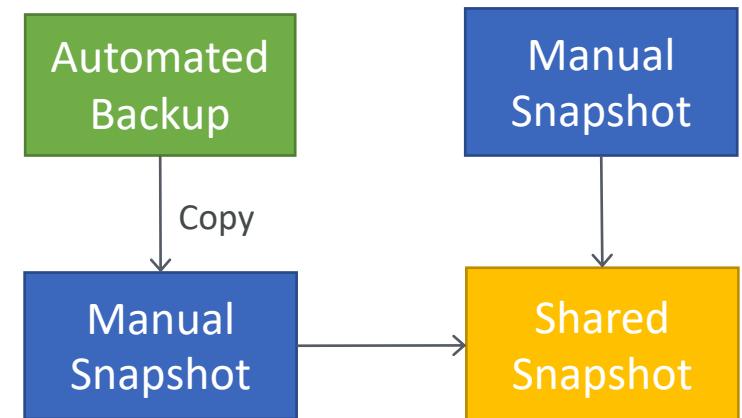
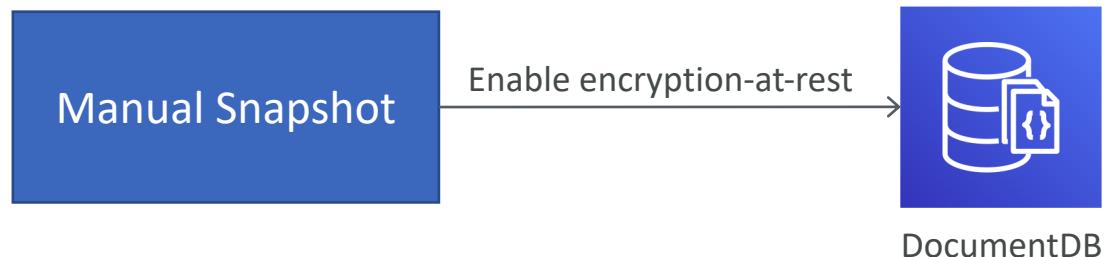
DocumentDB Backup and Restore

- Supports automatic backups
- Continuously backs up your data to S3 for PITR (max retention period of 35 days)
- latest restorable time for a PITR can be up to 5 mins in the past
- The first backup is a full backup. Subsequent backups are incremental
- Take manual snapshots to retain beyond 35 days
- Backup process does not impact cluster performance



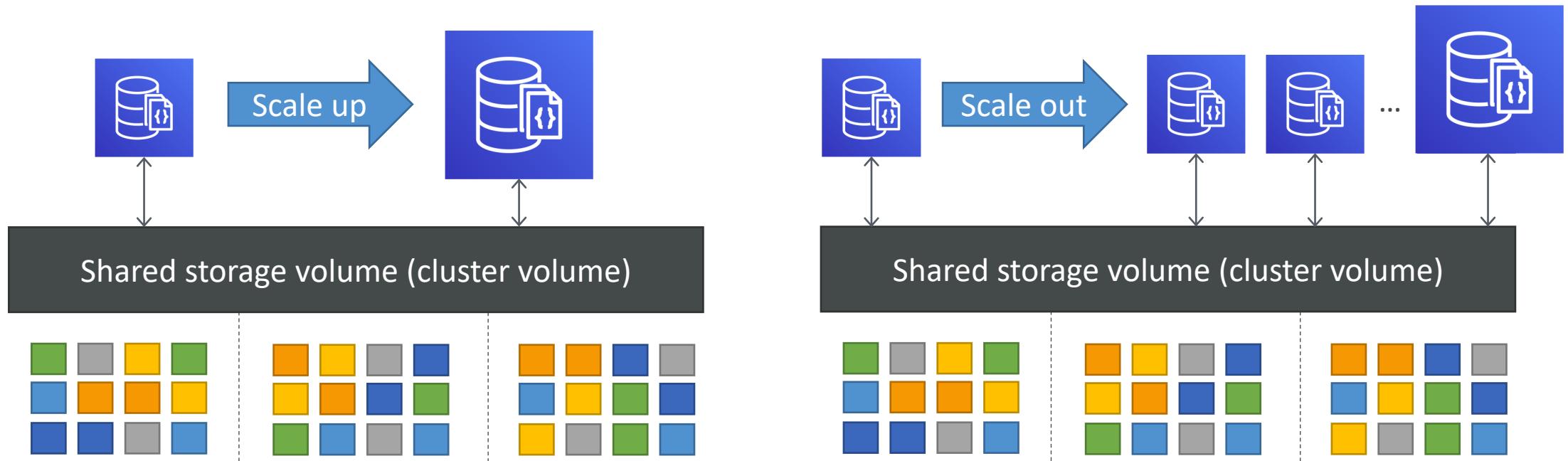
DocumentDB Backup and Restore

- Can only restore to a new cluster
- Can restore an unencrypted snapshot to an encrypted cluster (but not the other way round)
- To restore a cluster from an encrypted snapshot, you must have access to the KMS key
- Can only share manual snapshots (can copy and share automated ones)
- Can't share a snapshot encrypted using the default KMS key of the a/c
- Snapshots can be shared across accounts, but within the same region



DocumentDB Scaling

- MongoDB sharding not supported (instead offers read replicas / vertical scaling / storage scaling)
- Vertical scaling (scale up / down) – by resizing instances
- Horizontal scaling (scale out / in) – by adding / removing up to 15 read replicas
- Can scale up a replica independently from other replicas (typically for analytical workloads)
- Automatic scaling storage – 10 GB to 64TB (no manual intervention needed)

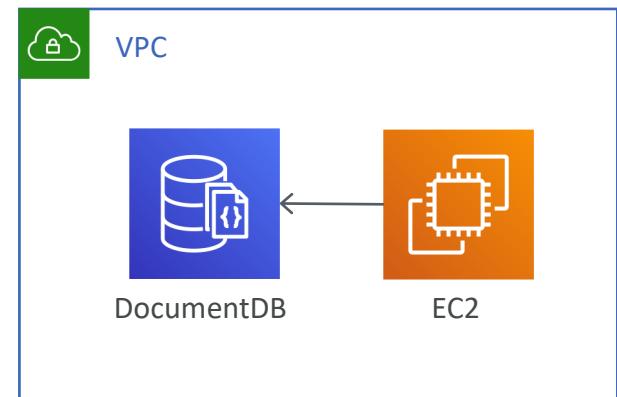


DocumentDB Security – IAM & Network

- You use IAM to manage DocumentDB resources
- Supports MongoDB default auth SCRAM (Salted Challenge Response Authentication Mechanism) for DB authentication
- Supports built-in roles for DB users with RBAC (role-based access control)
- DocumentDB clusters are VPC-only (use private subnets)
- Clients (MongoDB shell) can run on EC2 in public subnets within VPC
- Can connect to your on-premises IT infra via VPN



IAM



DocumentDB Security – Encryption

- Encryption at rest – with AES-256 using KMS
 - Applied to cluster data / replicas / indexes / logs / backups / snapshots
- Encryption in transit – using TLS
 - To enable TLS, set `tls` parameter in cluster parameter group
- To connect over TLS:
 - Download the certificate (public key) from AWS
 - Pass the certificate key while connecting to the cluster



KMS

DocumentDB > Cluster parameter groups > documentdbpg

Cluster parameters

Cluster parameter name	Values	Allowed values	Modifiable	Apply type	Data type	Description
audit_logs	disabled	enabled,disabled	true	dynamic	string	Enables auditing on cluster.
change_stream_log_retention_duration	10800	3600-86400	true	dynamic	integer	Duration of time in seconds that the change stream log is retained and can be consumed.
profiler	disabled	enabled,disabled	true	dynamic	string	Enables profiling for slow operations
profiler_sampling_rate	1.0	0.0-1.0	true	dynamic	float	Sampling rate for logged operations
profiler_threshold_ms	100	50-2147483646	true	dynamic	integer	Operations longer than profiler_threshold_ms will be logged
tls	enabled	disabled,enabled	true	static	string	Config to enable/disable TLS
ttl_monitor	enabled	disabled,enabled	true	dynamic	string	Enables TTL Monitoring

DocumentDB Pricing

- On-demand instances – pricing per second with a 10-minute minimum
- IOPS – per million IO requests
- Each DB page reads operation from the storage volume counts as one IO (one page = 8KB)
- Write IOs are counted in 4KB units.
- DB Storage – per GB per month
- Backups – per GB per month (backups up to 100% of your cluster's data storage is free)
- Data transfer – per GB
- Can temporarily stop compute instances for up to 7 days



DocumentDB

DocumentDB Monitoring

- API calls logged with CloudTrail
- Common CloudWatch metrics
 - CPU or RAM utilization – CPUUtilization / FreeableMemory
 - IOPS metrics – VolumeReadIOPS / VolumeWriteIOPS / WriteIOPS / ReadIOPS
 - Database connections – DatabaseConnections
 - Network traffic – NetworkThroughput
 - Storage volume consumption – VolumeBytesUsed
- Two types of logs can be published/exported to CloudWatch Logs
 - Profiler logs
 - Audit logs



CloudTrail



CloudWatch

DocumentDB Profiler (profiler logs)

- Logs (into CloudWatch Logs) the details of ops performed on your cluster
- Helps identify slow operations and improve query performance
- Accessible from CloudWatch Logs
- To enable profiler:
 - Set the parameters – profiler, profiler_threshold_ms, and profiler_sampling_rate
 - Enable Logs Exports for Audit logs by modifying the instance
 - Both the steps above are mandatory



CloudWatch

DocumentDB audit logs

- Records DDL statements, authentication, authorization, and user management events to CloudWatch Logs
- Exports your cluster's auditing records (JSON documents) to CloudWatch Logs
- Accessible from CloudWatch Logs
- To enable auditing:
 - Set parameter audit_logs=enabled
 - Enable Logs Exports for Audit logs by modifying the instance
 - Both the steps above are mandatory



CloudWatch

DocumentDB Performance Management

- Use explain command to identify slow queries

```
db.runCommand({explain: {<query document>} })
```

- Can use **db.adminCommand** to find and terminate queries
- Example – to terminate long running / blocked queries

```
db.adminCommand({killOp: 1, op: <opid of the query>});
```

Creating a DocumentDB Cluster



Demo



DataCumulus | RIZMAXed

Neptune

It's all about discovering relationships!



DataCumulus | RIZMAXed

Amazon Neptune – Overview

GRAPH

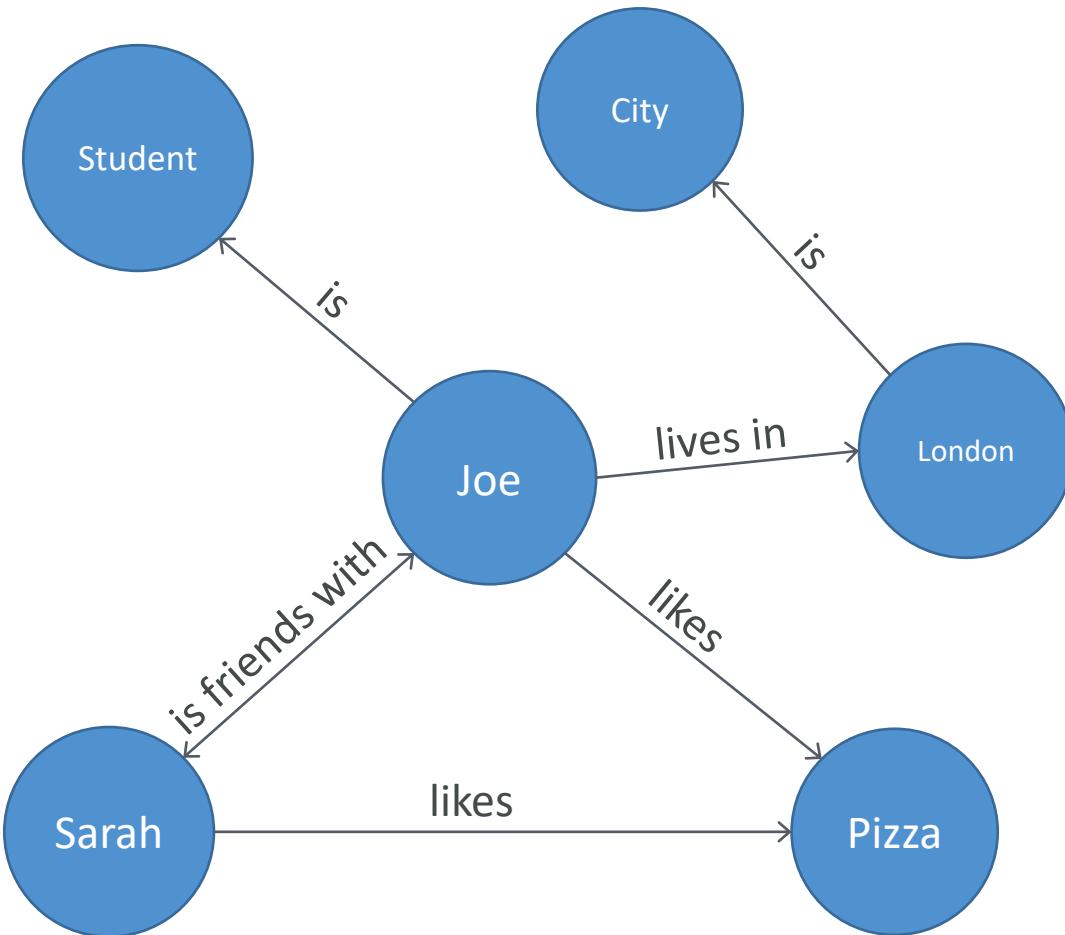


Neptune

- Fully managed graph database service (non-relational)
- Relationships are first-class citizens
- Can quickly navigate relationships and retrieve complex relations between highly connected datasets
- Can query billions of relationships with millisecond latency
- ACID compliant with immediate consistency
- Supports transaction semantics for highly concurrent OLTP workloads (ACID transactions)
- Supported graph query languages – Apache TinkerPop Gremlin and RDF/SPARQL
- Supports 15 low-latency read replicas (Multi-AZ)
- Use cases:
 - Social graph / Knowledge graph
 - Fraud detection
 - Real-time big data mining
 - Customer interests and recommendations (Recommendation engines)

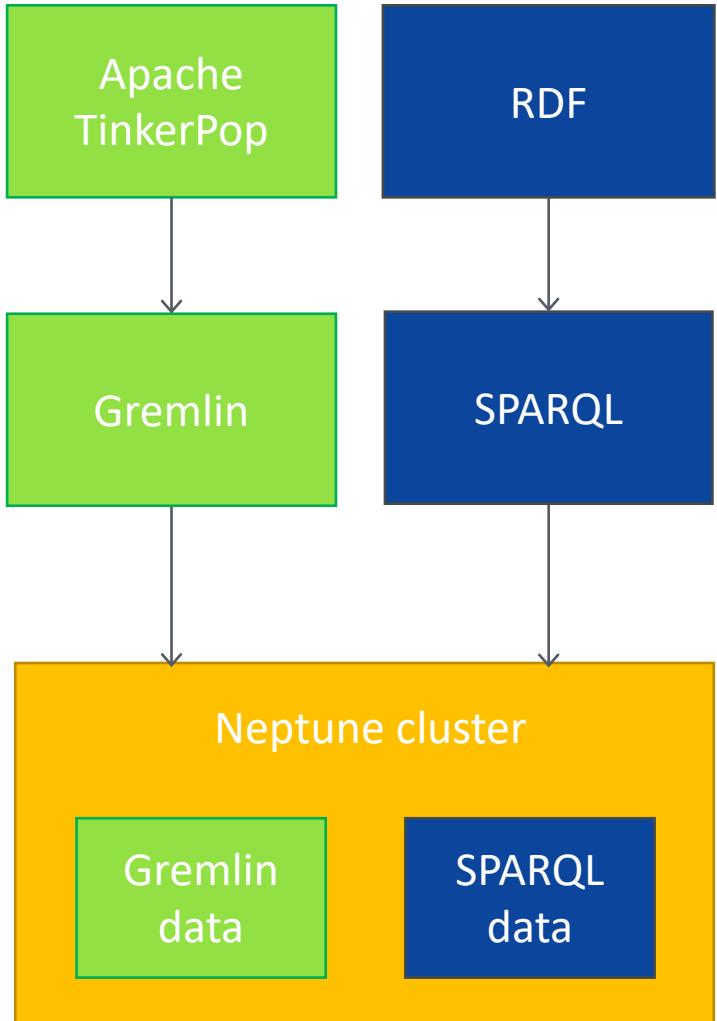
Graph Database

- Models relationships between data
 - e.g. Subject / predicate / object / graph (quad)
 - Joe likes pizza
 - Sarah is friends with Joe
 - Sarah likes pizza too
 - Joe is a student and lives in London
 - Let's you ask questions like "identify Londoners who like pizza" or "identify friends of Londoners who like pizza"
- Uses nodes (vertices) and edges (actions) to describe the data and relationships between them
- DB stores – person / action / object (and a graph ID or edge ID)
- Can filter or discover data based on strength, weight, or quality of relationships



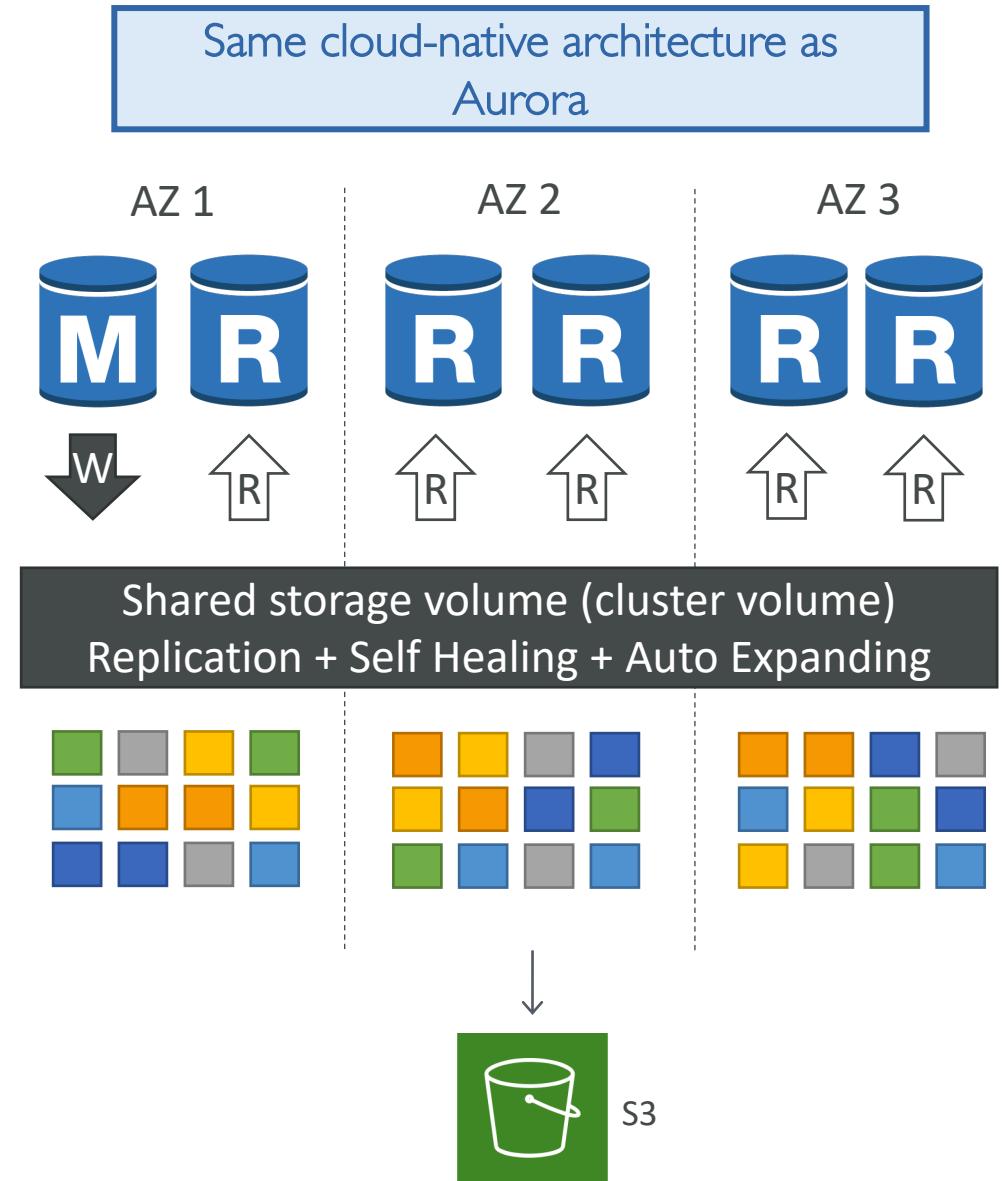
Graph query languages

- Neptune supports two popular modeling frameworks – Apache TinkerPop and RDF/SPARQL
- TinkerPop uses **Gremlin** traversal language
- RDF (W3C standard) uses **SPARQL**
- SPARQL is great for multiple data sources, has large variety of datasets available
- We can use Gremlin or SPARQL to load data into Neptune and then to query it
- You can store both Gremlin and SPARQL graph data on the same Neptune cluster
- It gets stored separately on the cluster
- Graph data inserted using one query language can only be queried with that query language (and not with the other)

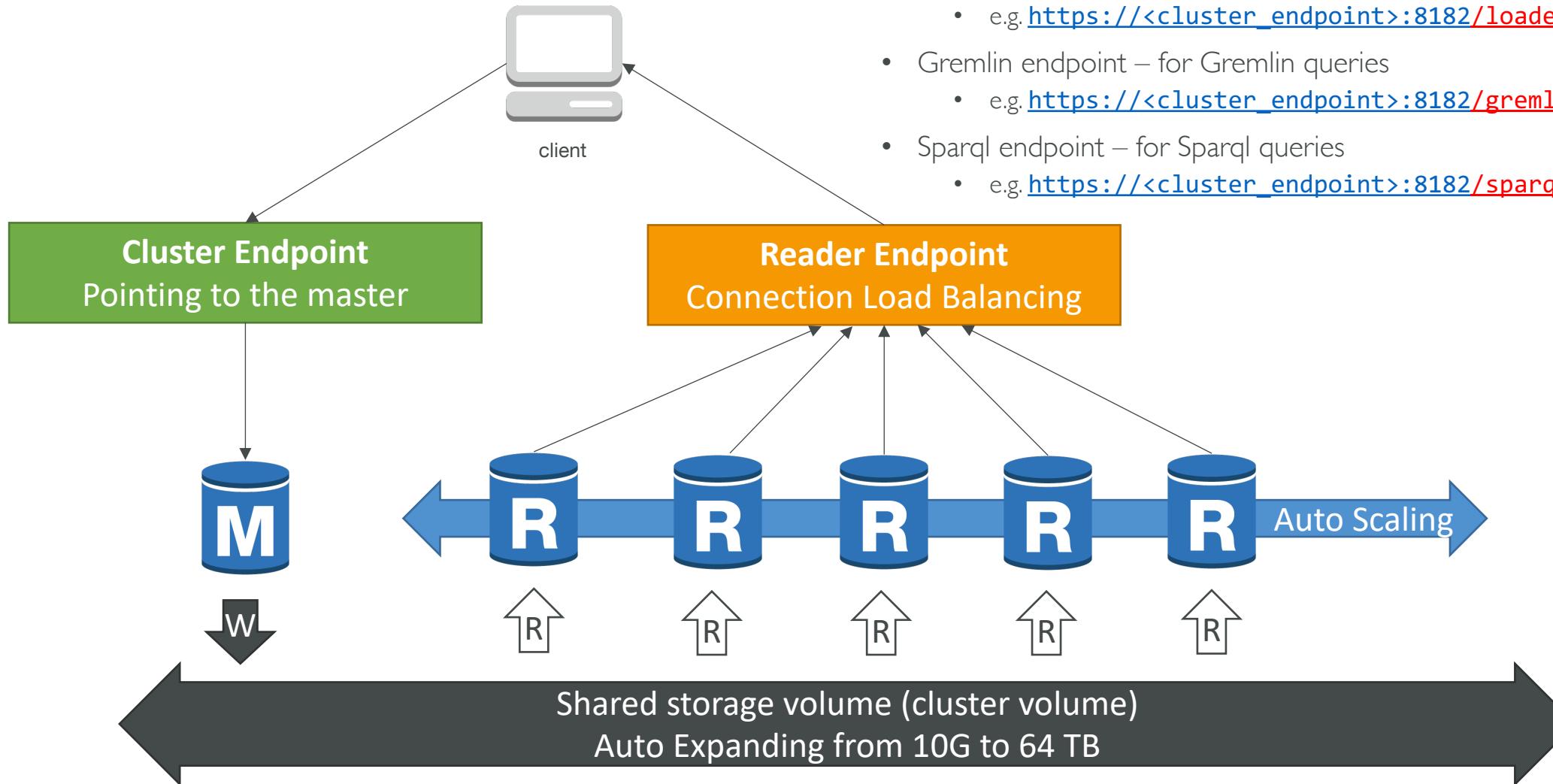


Neptune Architecture

- 6 copies of your data across 3 AZ (distributed design)
 - Lock-free optimistic algorithm (quorum model)
 - 4 copies out of 6 needed for writes (4/6 write quorum - data considered durable when at least 4/6 copies acknowledge the write)
 - 3 copies out of 6 needed for reads (3/6 read quorum)
 - Self healing with peer-to-peer replication, Storage is striped across 100s of volumes
- One Neptune Instance takes writes (master)
- Compute nodes on replicas do not need to write/replicate (=improved read performance)
- Log-structured distributed storage layer – passes incremental log records from compute to storage layer (=faster)
- Master + up to 15 Read Replicas serve reads
- Data is continuously backed up to S3 in real time, using storage nodes (compute node performance is unaffected)



Neptune Cluster



Creating a Neptune Cluster

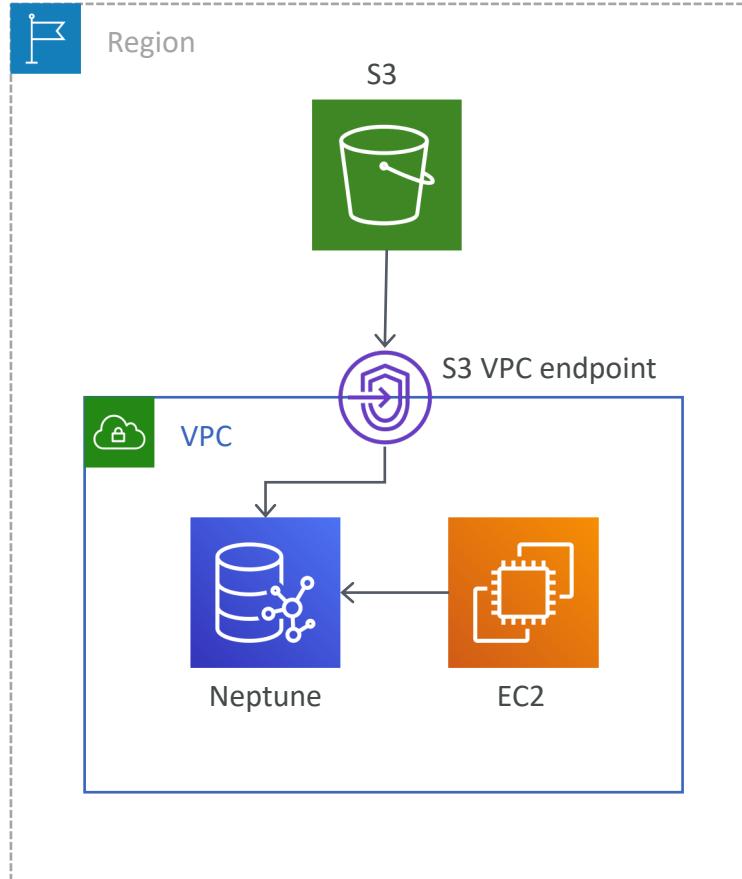


Demo

Bulk loading data into Neptune

- Use the loader endpoint (HTTP POST to the loader endpoint)
 - e.g.

```
curl -X POST -H 'Content-Type: application/json'  
https://<cluster_endpoint>:8182/loader -d  
'{  
    "source": "s3://bucket_name/key_name,  
    ...  
'}
```
- S3 data can be accessed using an S3 VPC endpoint (allows access to S3 resources from your VPC)
- Neptune cluster must assume an IAM role with **S3 read access**
- S3 VPC endpoint can be created using the VPC management console
- S3 bucket must be in the same region as the Neptune cluster
- Load data formats
 - csv (for gremlin), ntriples / nquads / rdfxml / turtle (for sparql)
- All files must be UTF-8 encoded
- Multiple files can be loaded in a single job



Loading graph data into Neptune from S3



Demo

Neptune Workbench

- Lets you query your Neptune cluster using notebooks
- Notebooks are Jupyter notebooks hosted by Amazon SageMaker
- Available within AWS console
- Notebook runs behind the scenes on an EC2 host in the same VPC and has IAM authentication
- The security group that you attach in the VPC where Neptune is running must have an additional rule that allows inbound connections from itself

The screenshot shows a Jupyter Notebook interface titled "My-Neptune-Notebook". The top bar includes tabs for "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", "Help", and "Neptune". A Python 3 kernel is selected. The notebook displays two code cells:

```
In [1]: %status
Out[1]: {'status': 'healthy',
'startTime': 'Thu Jun 25 13:15:29 UTC 2020',
'dbEngineVersion': '1.0.2.2.R2',
'role': 'writer',
'gremlin': {'version': 'tinkerpop-3.4.3'},
'sparql': {'version': 'sparql-1.1'},
'labMode': {'ObjectIndex': 'disabled',
'ReadWriteConflictDetection': 'enabled'}}
```

```
In [2]: %load
Source: s3://
Format:
AWS Region: us-east-1
Load ARN: Type something
Fail on Fail...: TRUE
Parallelism : HIGH
Update Sin...: FALSE
Submit
```

The configuration panel for cell In [2] includes fields for Source (set to s3://), Format, AWS Region (set to us-east-1), Load ARN (placeholder "Type something"), Fail on Fail... (set to TRUE), Parallelism (set to HIGH), and Update Sin... (set to FALSE). A "Submit" button is at the bottom.

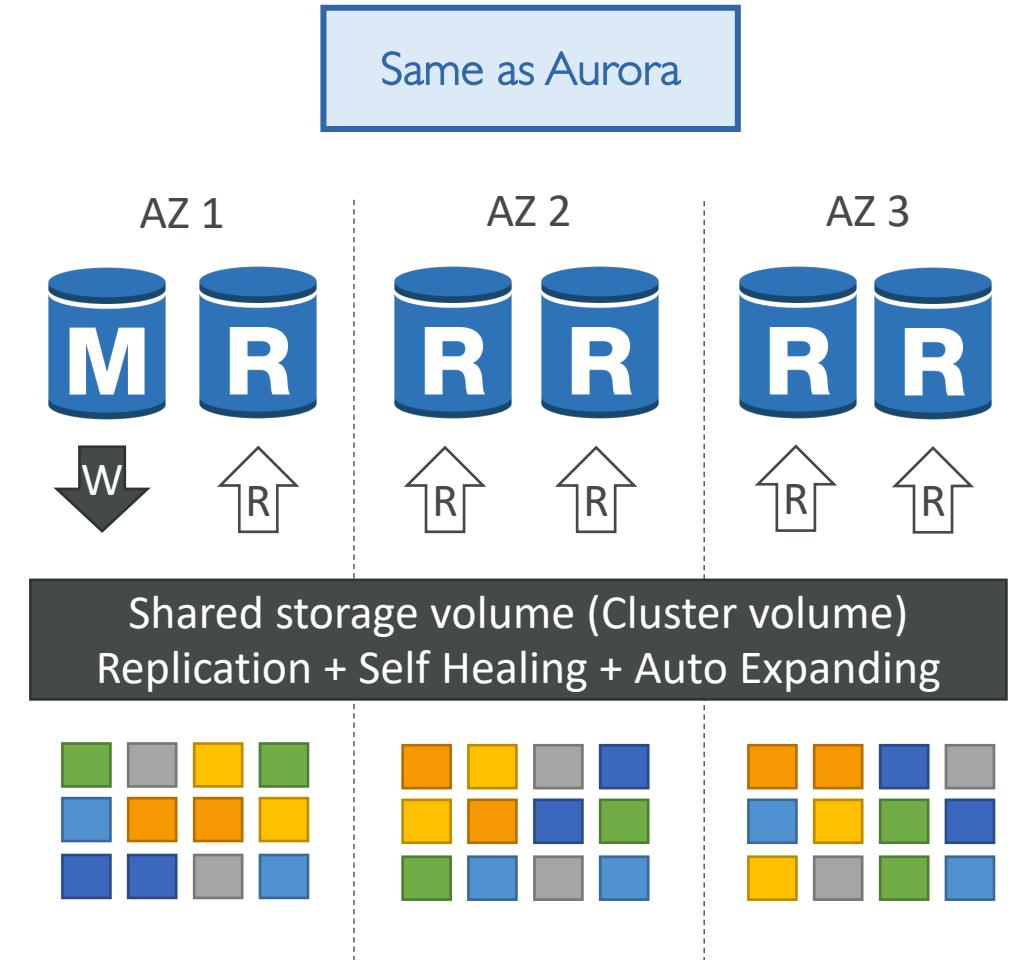
Querying Neptune via Notebooks (Jupyter)



Demo

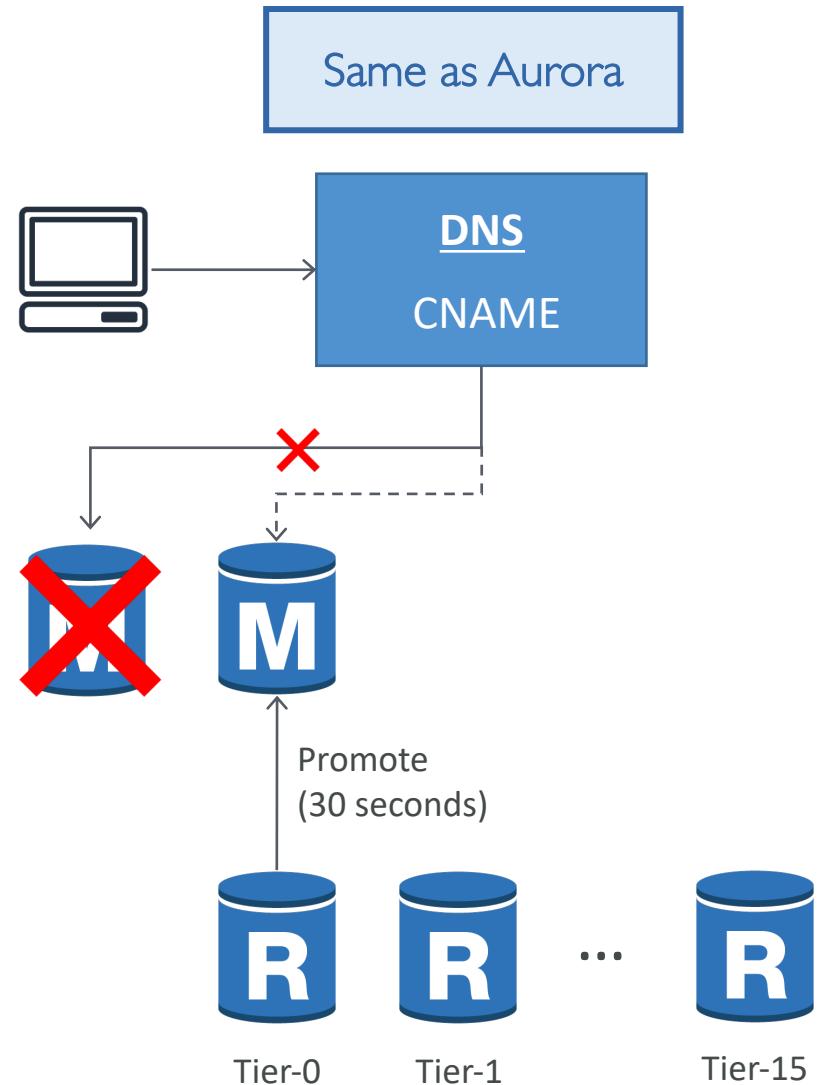
Neptune Replication

- Up to 15 read replicas
- ASYNC replication
- Replicas share the same underlying storage layer
- Typically take 10s of milliseconds (replication lag)
- Minimal performance impact on the primary due to replication process
- Replicas double up as failover targets (standby instance is not needed)



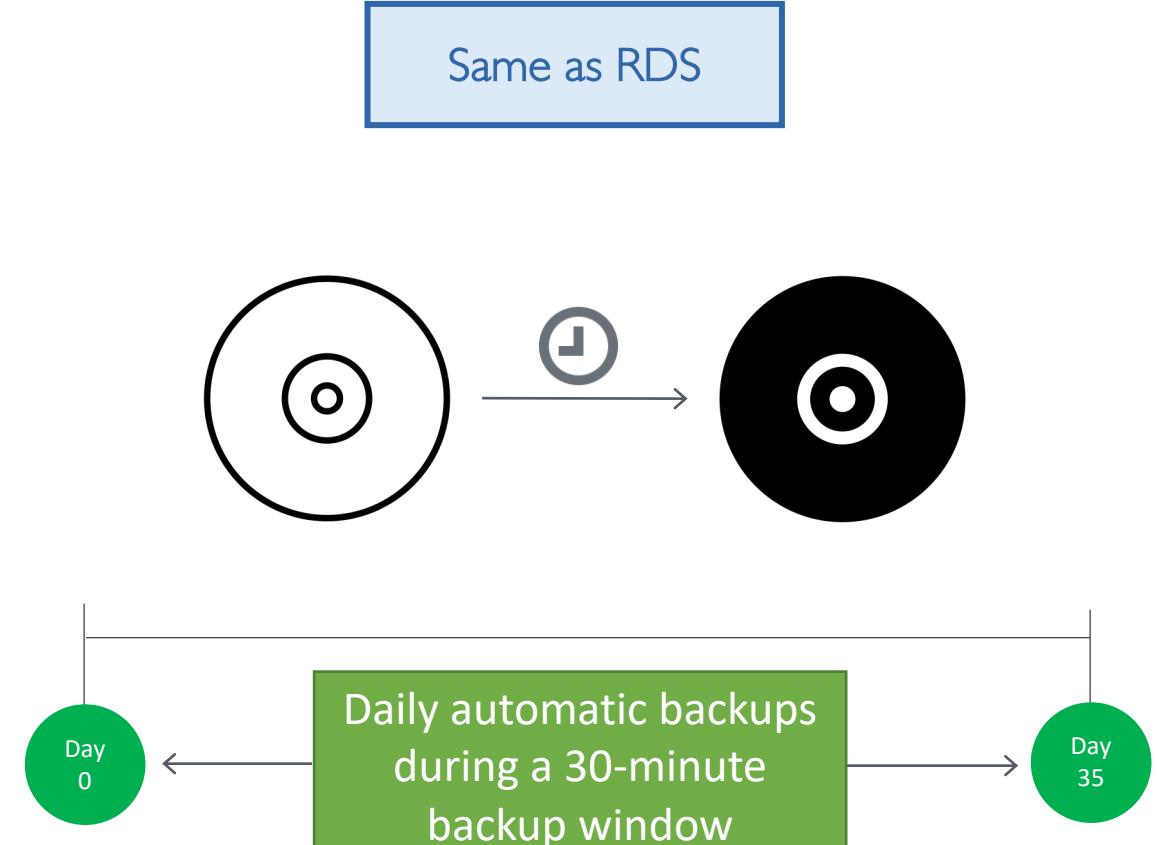
Neptune High Availability

- Failovers occur automatically
- A replica is automatically promoted to be the new primary during DR
- Neptune flips the CNAME of the DB instance to point to the replica and promotes it
- Failover to a replica typically takes under 30-120 seconds (minimal downtime)
- Creating a new instance takes about 15 minutes (post failover)
- Failover to a new instance happens on a best-effort basis and can take longer



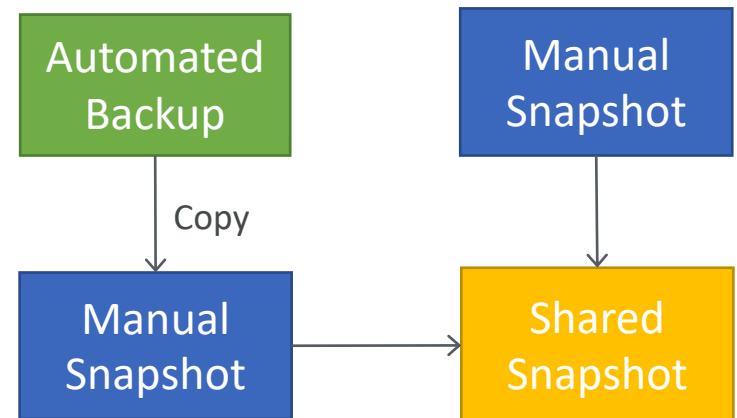
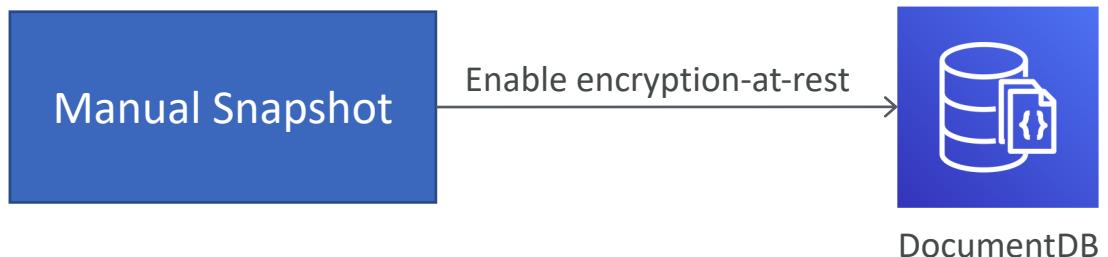
Neptune Backup and Restore

- Supports automatic backups
- Continuously backs up your data to S3 for PITR (max retention period of 35 days)
- latest restorable time for a PITR can be up to 5 mins in the past (RPO = 5 minutes)
- The first backup is a full backup.
Subsequent backups are incremental
- Take manual snapshots to retain beyond 35 days
- Backup process does not impact cluster performance



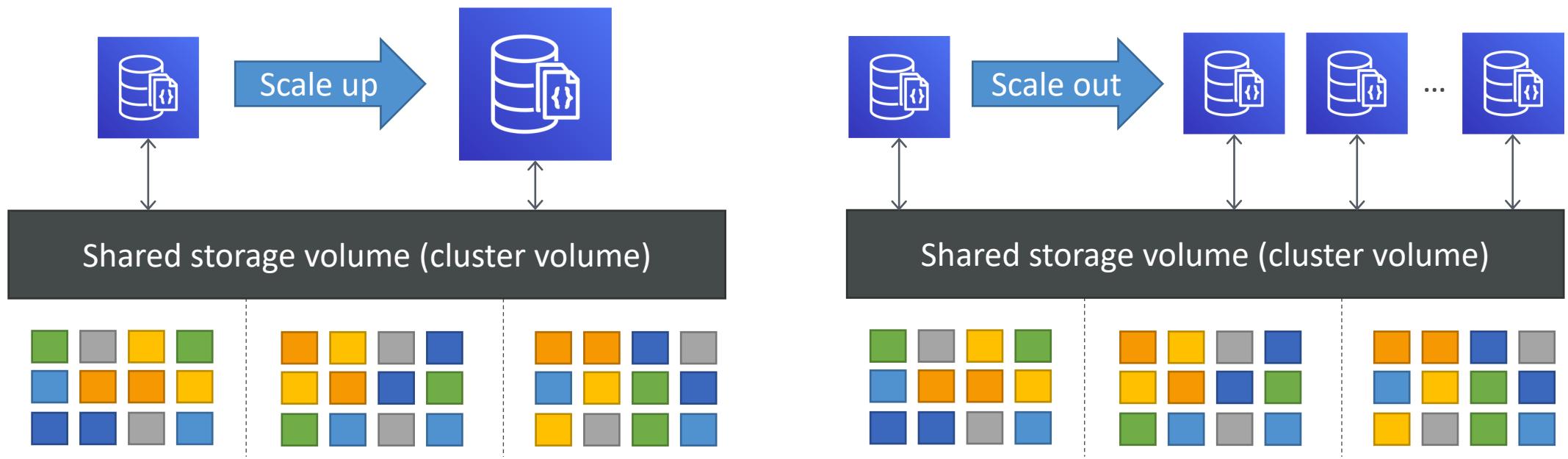
Neptune Backup and Restore

- Can only restore to a new cluster
- Can restore an unencrypted snapshot to an encrypted cluster (but not the other way round)
- To restore a cluster from an encrypted snapshot, you must have access to the KMS key
- Can only share manual snapshots (can copy and share automated ones)
- Can't share a snapshot encrypted using the default KMS key of the a/c
- Snapshots can be shared across accounts, but within the same region



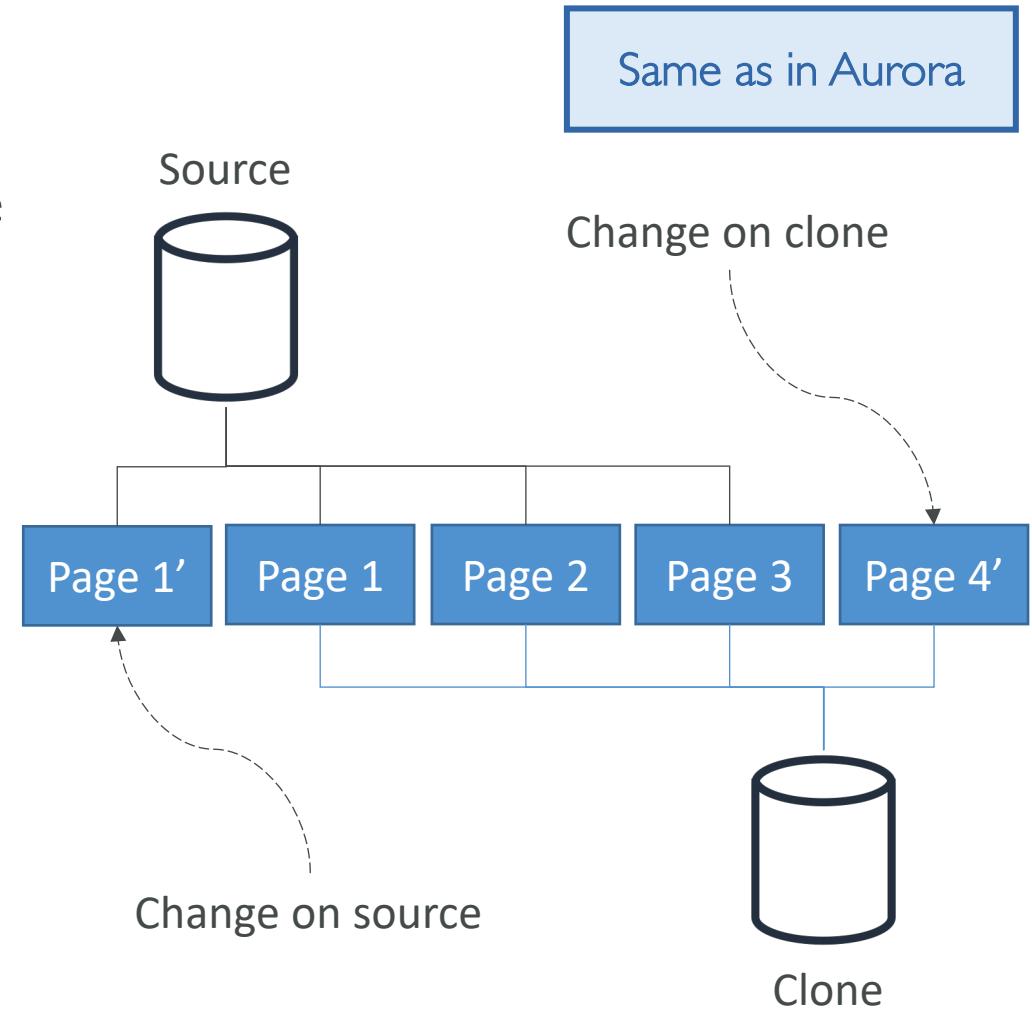
Neptune Scaling

- Vertical scaling (scale up / down) – by resizing instances
- Horizontal scaling (scale out / in) – by adding / removing up to 15 read replicas
- Automatic scaling storage – 10 GB to 64 TB (no manual intervention needed)



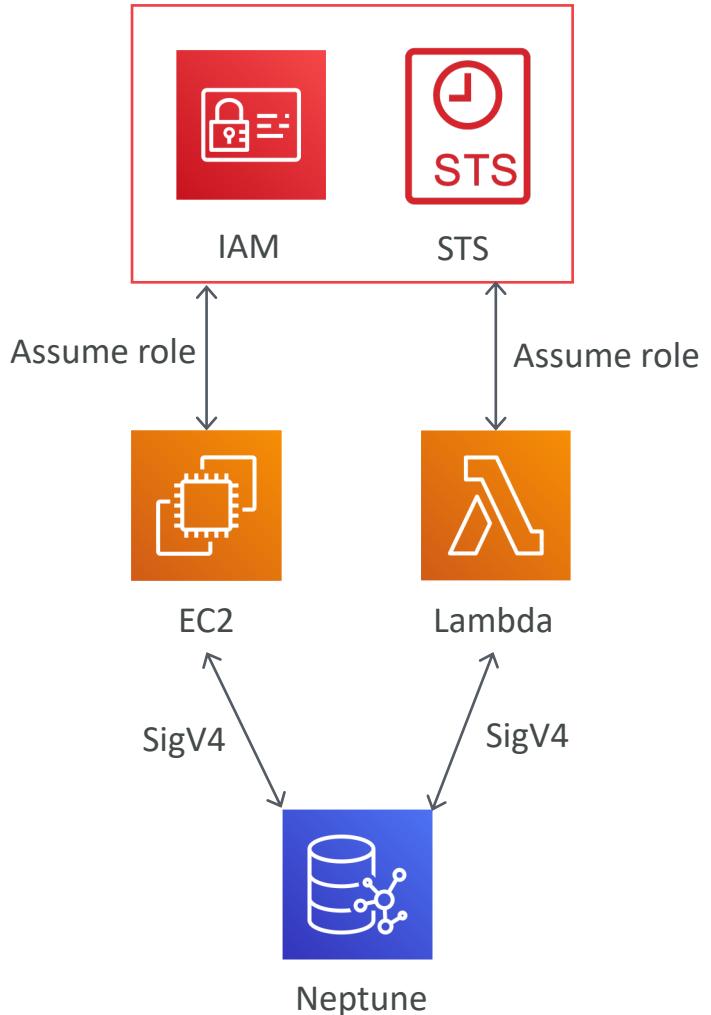
Database Cloning in Neptune

- Different from creating read replicas – clones support both reads and writes
- Different from replicating a cluster – clones use same storage layer as the source cluster
- Requires only minimal additional storage
- Quick and cost-effective
- Only within region (can be in different VPC)
- Can be created from existing clones
- Uses a copy-on-write protocol
 - both source and clone share the same data initially
 - data that changes, is then copied at the time it changes either on the source or on the clone (i.e. stored separately from the shared data)
 - delta of writes after cloning is not shared



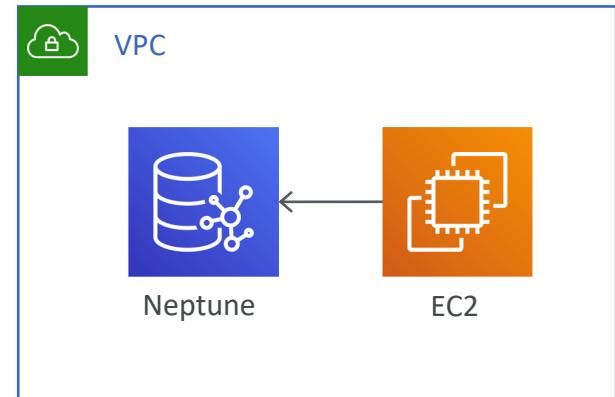
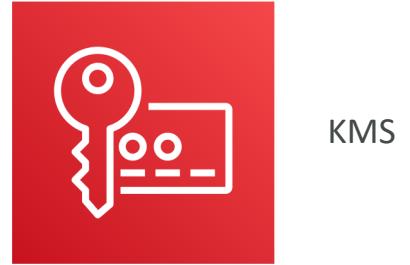
Neptune Security – IAM

- Uses IAM for authentication and authorization to manage Neptune resources
- Supports IAM Authentication (with AWS SigV4)
- You use temporary credentials using an assumed role
 - Create an IAM role
 - Setup trust relationship
 - Retrieve temp creds
 - Sign the requests using the creds



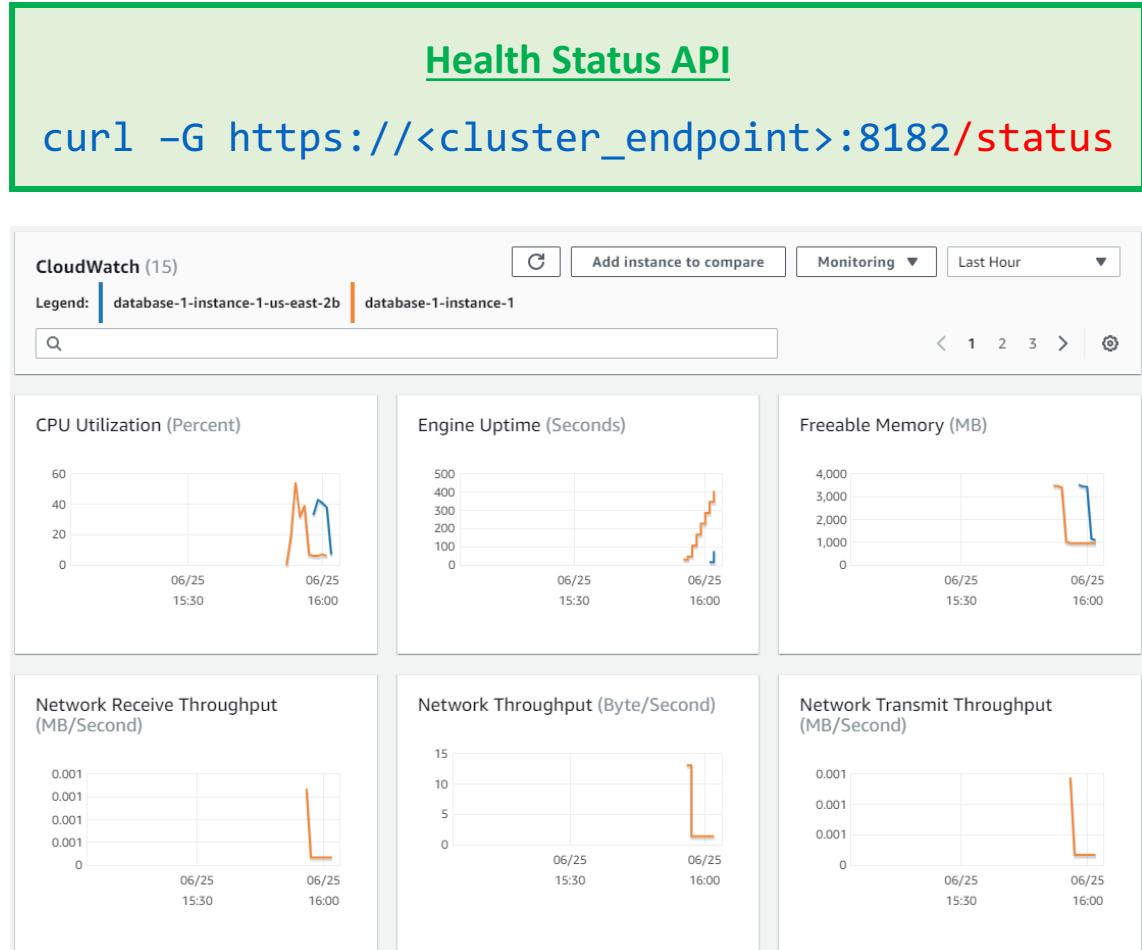
Neptune Security – Encryption & Network

- Encryption in transit – using SSL / TLS
 - Cluster parameter `neptune_enforce_ssl = 1` (is default)
- Encryption at rest – with AES-256 using KMS
 - encrypts data, automated backups, snapshots, and replicas in the same cluster
- Neptune clusters are VPC-only (use private subnets)
- Clients can run on EC2 in public subnets within VPC
- Can connect to your on-premises IT infra via VPN
- Use security groups to control access



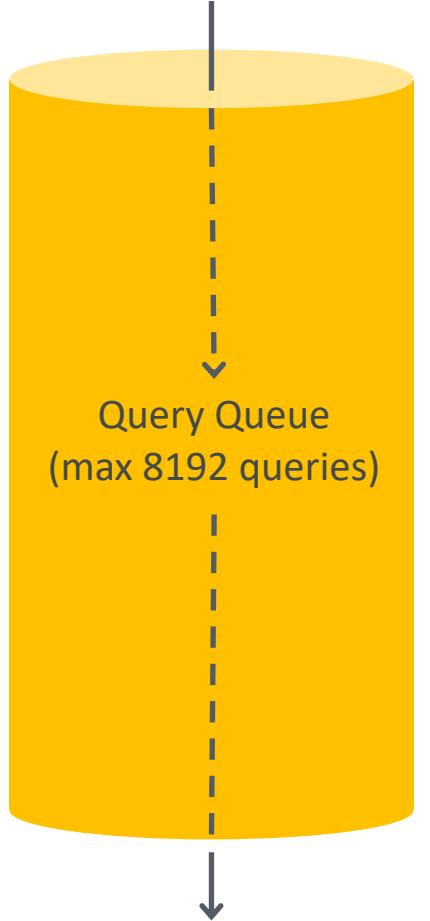
Neptune Monitoring

- Integrated with CloudWatch
- can use **Audit log** files by enabling DB cluster parameter neptune_enable_audit_log
- must restart DB cluster after enabling audit logs
- audit log files are rotated beyond 100MB (not configurable)
- audit logs are not stored in sequential order (can be ordered using the timestamp value of each record)
- audit log data can be published (exported) to a CloudWatch Logs log group by enabling **Log exports** for your cluster
- API calls logged with CloudTrail



Query Queuing in Neptune

- Max 8192 queries can be queued up per Neptune instance
- Queries beyond 8192 will result in ThrottlingException
- Use CloudWatch metric MainRequestQueuePendingRequests to get number of queries queued (5 min granularity)
- Get acceptedQueryCount value using Query Status API
 - For Gremlin, acceptedQueryCount = current count of queries queued
 - For SPARQL, acceptedQueryCount = all queries accepted since the server started



Neptune Service Errors

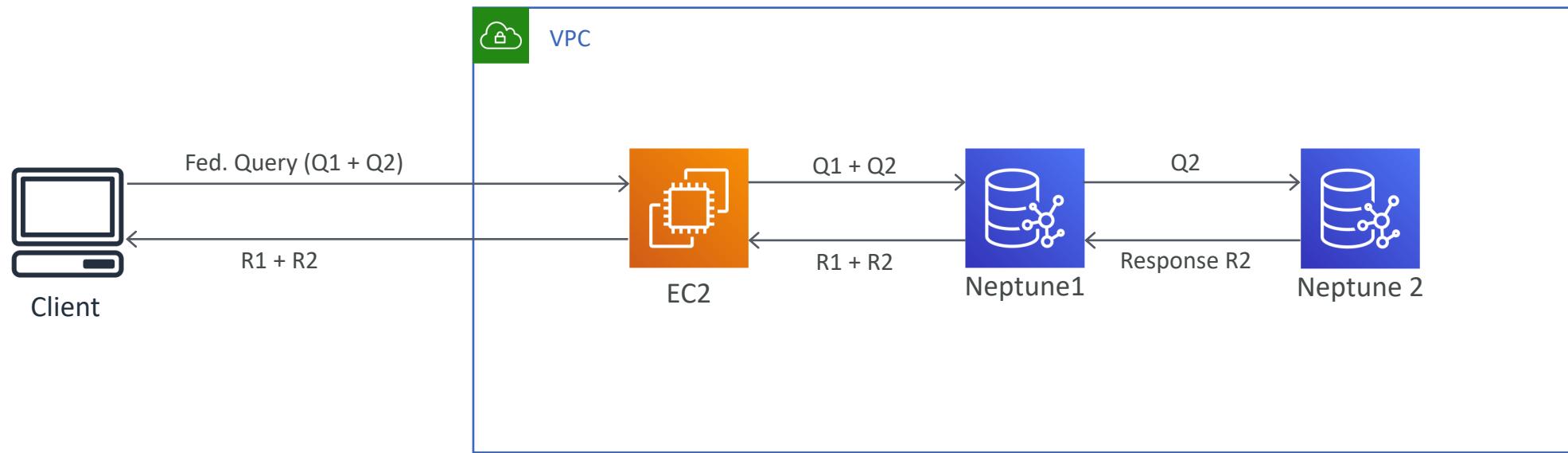
- Graph engine errors
 - Errors related to cluster endpoints, are HTTP error codes
 - Query errors – QueryLimitException / MemoryLimitExceededException / TooManyRequestsException etc.
 - IAM Auth errors – Missing Auth / Missing token / Invalid Signature / Missing headers / Incorrect Policy etc
- API errors
 - HTTP errors related to APIs (CLI / SDK)
 - InternalFailure / AccessDeniedException / MalformedQueryString / ServiceUnavailable etc
- Loader Error
 - LOAD_NOT_STARTED / LOAD_FAILED / LOAD_S3_READ_ERROR / LOAD_DATA_DEADLOCK etc



Neptune

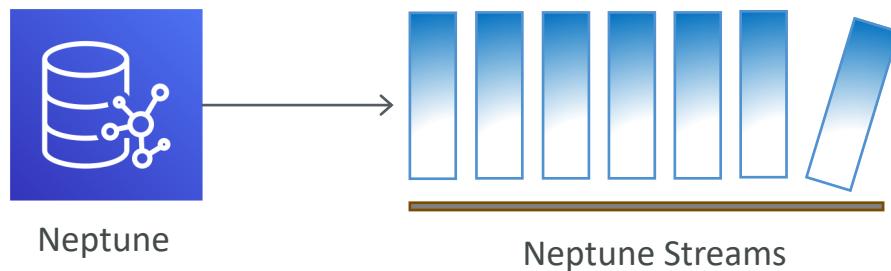
SPARQL federated query

- Query across multiple Neptune clusters or external data sources that support the protocol, and aggregate the results
- Supports only read operations



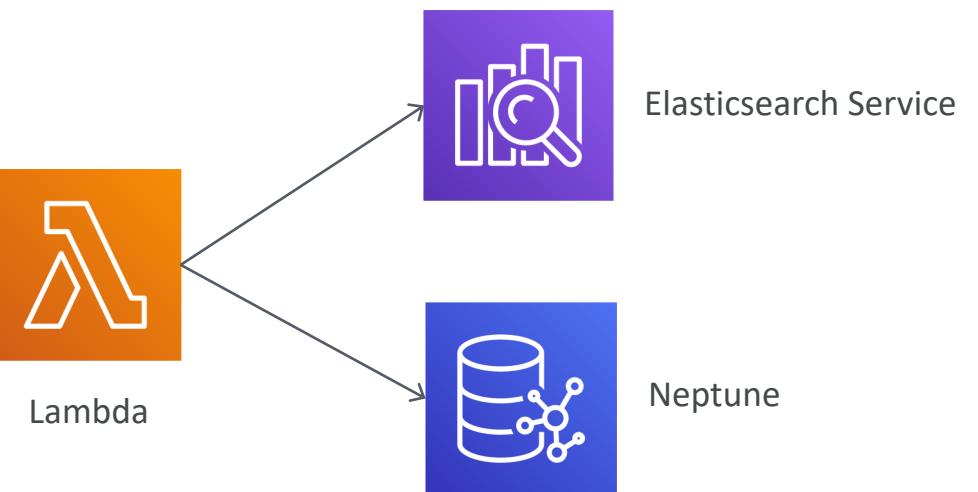
Neptune Streams

- Capture changes to your graph (change logs)
- Similar to DynamoDB streams
- Can be processed with Lambda (use Neptune Streams API)
- SPARQL
 - https://<cluster_endpoint>:8182/sparql/stream
- Gremlin
 - https://<cluster_endpoint>:8182/gremlin/stream
- Only GET method is allowed



Use cases

- Amazon ES Integration
 - To perform full-text search queries on Neptune data
 - Uses Streams + federated queries
 - Supported for both gremlin and SPARQL
- Neptune-to-Neptune Replication



Neptune Pricing

- You only pay for what you use
- On-demand instances – per hour pricing
- IOPS – per million IO requests
 - Every DB page read operation = one IO
 - Each page is 16 KB in Neptune
 - Write IOs are counted in 4KB units
- DB Storage – per GB per month
- Backups (automated and manual) – per GB per month
- Data transfer – per GB
- Neptune Workbench – per instance hour



Neptune

Amazon Elasticsearch Service

The ELK Stack on AWS Cloud!

Amazon Elasticsearch Service – Overview

SEARCH

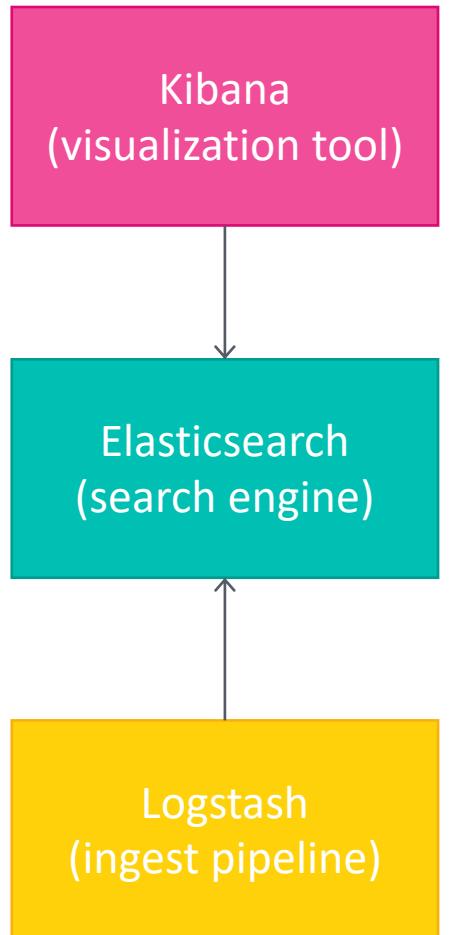


Elasticsearch
Service

- May be called Amazon ES at the exam
- Managed version of ElasticSearch (open source project)
- The ELK Stack on AWS Cloud (ElasticSearch + Logstash + Kibana)
 - ElasticSearch – provides search and indexing functionality
 - Logstash – provides log ingestion mechanism, alternative to CloudWatch Logs
 - Kibana – provides real-time dashboards for ES data, is a visualization tool
- Needs to provision servers (not a serverless offering)
- Use cases:
 - Log analytics
 - Real time application monitoring
 - Security analytics
 - Full text search
 - Clickstream analytics
 - Indexing

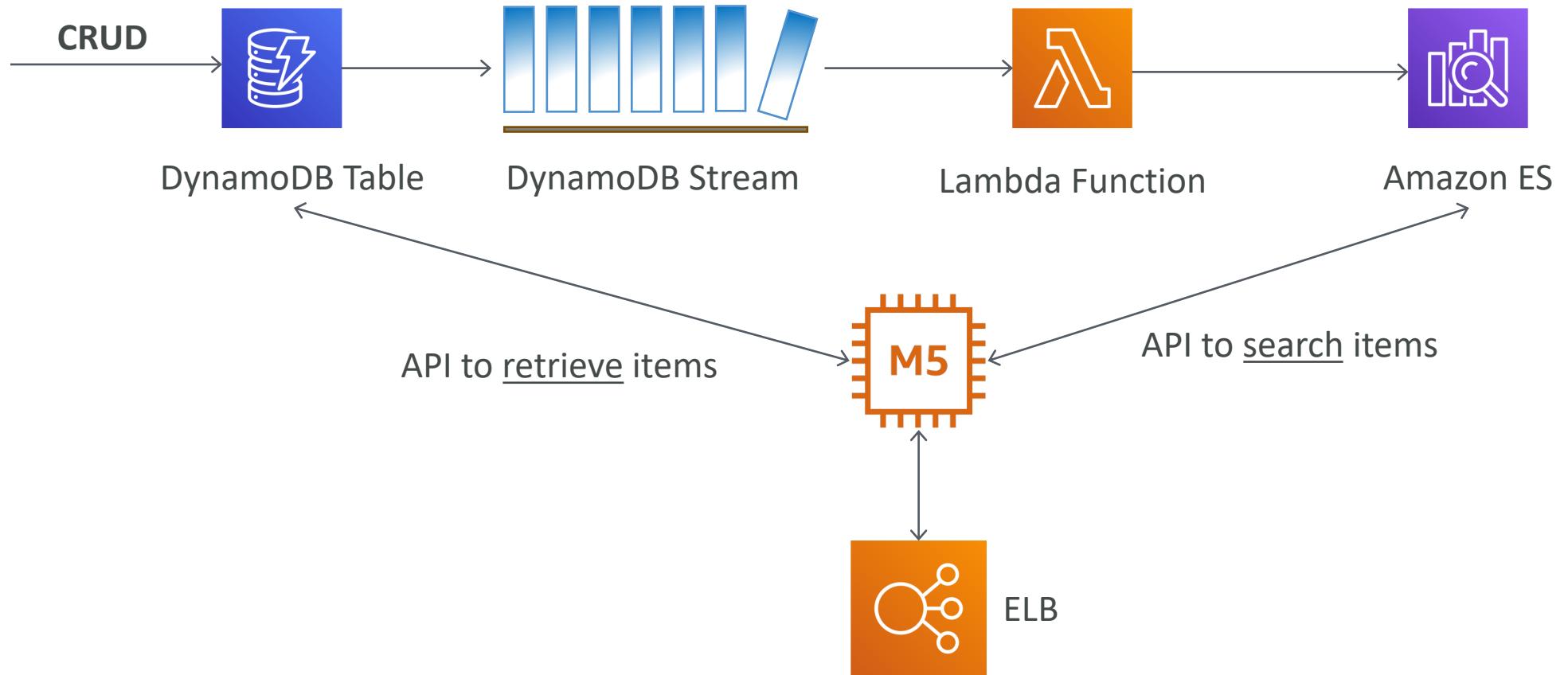
Elasticsearch + Logstash + Kibana (ELK)

- ElasticSearch – provides search and indexing capability
 - You send data in the form of JSON documents to Elasticsearch using the API / Logstash / Firehose
 - Elasticsearch automatically adds a searchable reference to these documents in the cluster's index.
- Logstash
 - Log ingestion mechanism, uses the “Logstash agent”
 - Alternative to CloudWatch Logs (you decide on retention and granularity)
- Kibana
 - Provides real-time dashboards on top of the data that sits in ES
 - Alternative to CloudWatch dashboards (more advanced capabilities)



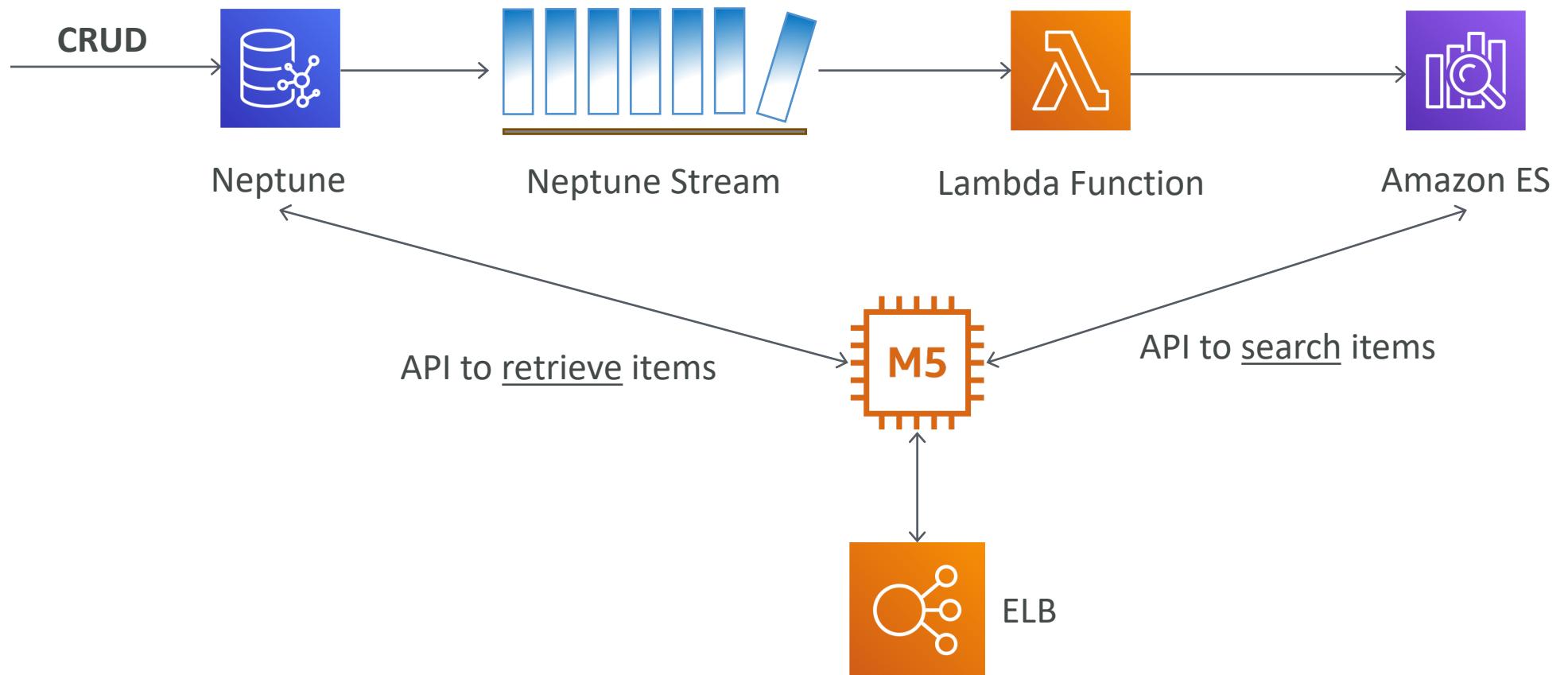
ElasticSearch Service patterns

DynamoDB



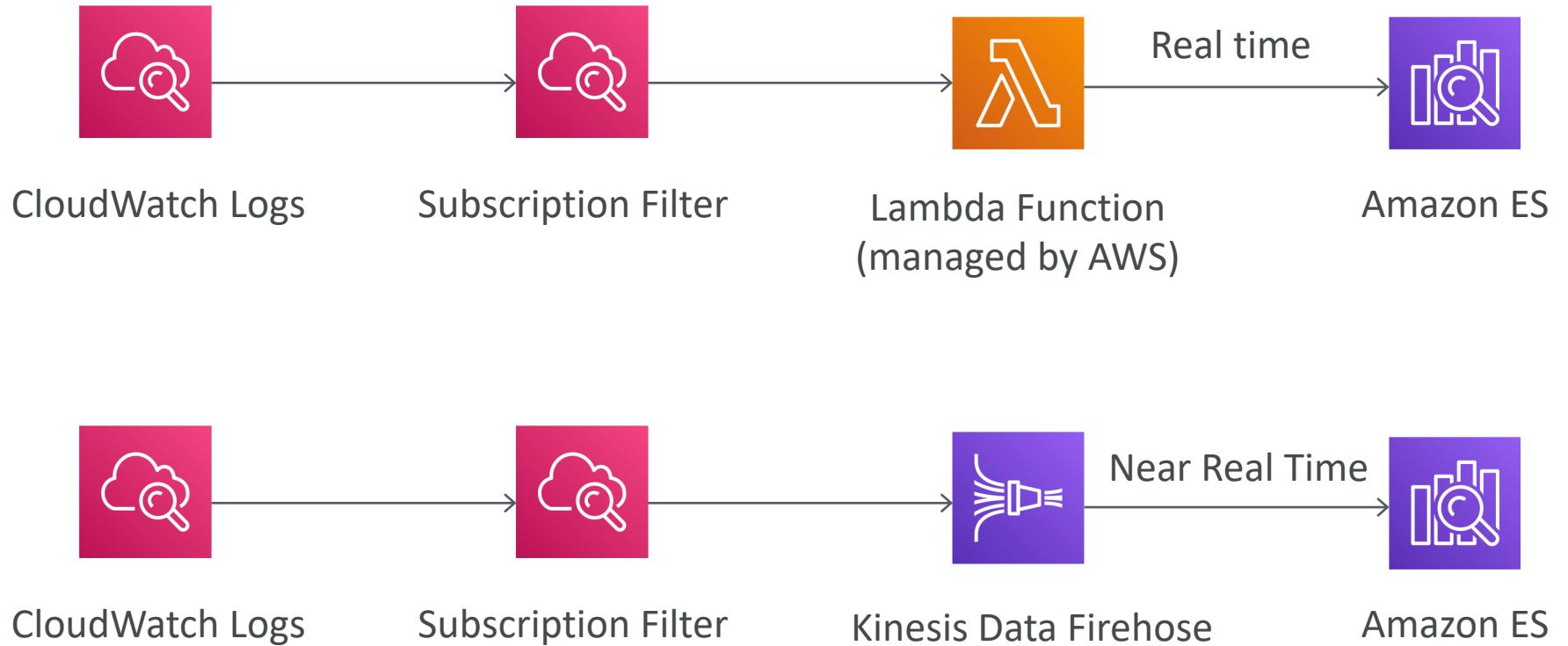
ElasticSearch Service patterns

Neptune



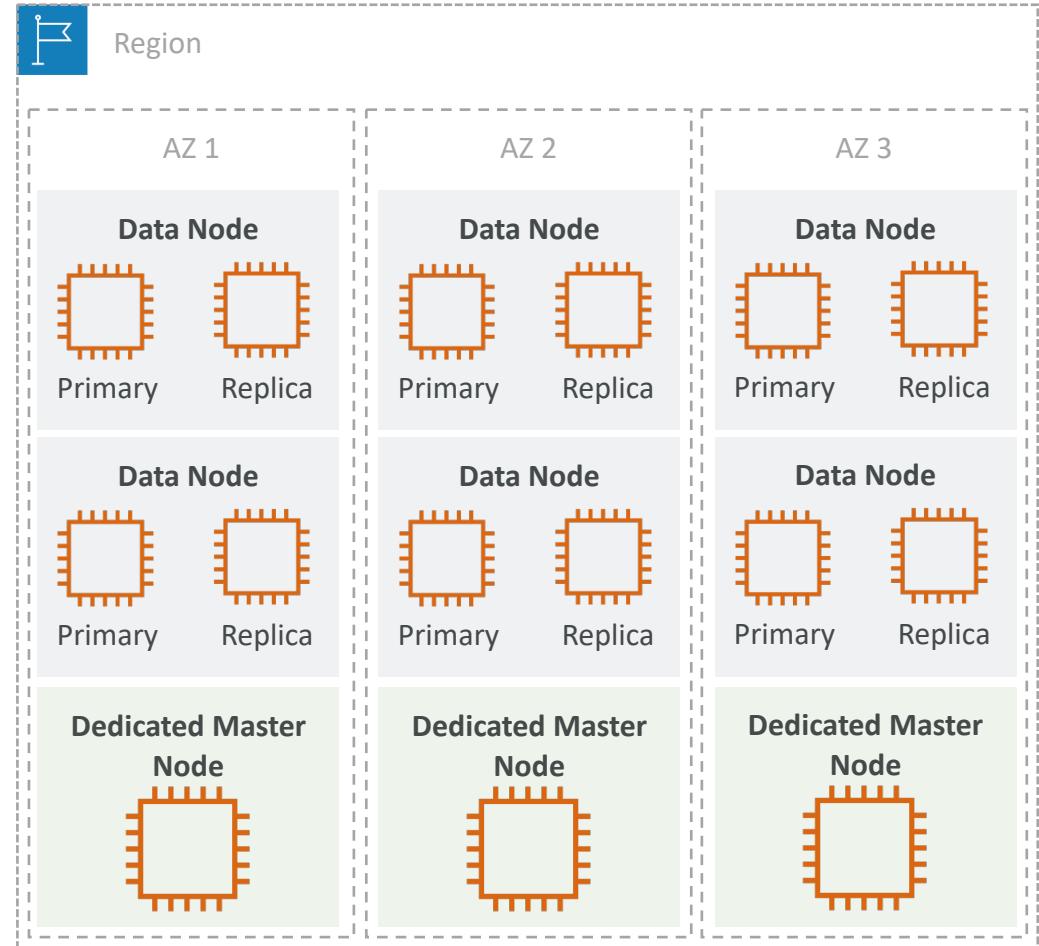
ElasticSearch Service patterns

CloudWatch Logs



Elasticsearch Service – Multi-AZ

- Supports multi-AZ – up to three-AZ deployments (1-, 2-, or 3-AZs)
- Can be deployed within VPC
- Uses dedicated master nodes to increase cluster stability
- Dedicated master nodes are distributed across 3-AZs, even if you select 2-AZ deployment
- Recommended to use 3-AZs for production
- For multi-AZ, create at least one replica for each index in the cluster.
- Without replicas, cross-AZ replication doesn't happen which largely defeats the purpose of Multi-AZ.



Logging in Elasticsearch Service

- Amazon ES provides three types of Elasticsearch logs
 - error logs
 - search slow logs
 - index slow logs
- Accessible through CloudWatch Logs
- Can be enabled from within ES console (disabled by default)



CloudWatch

ElasticSearch Service Pricing

- Instance pricing (priced per hour)
 - On-Demand instance
 - Reserved instance – discounted pricing over 1- or 3-year term
 - UltraWarm instance – On-Demand or Managed storage (new tier type, cost-effective way to store large amounts of read-only data)
- EBS volume pricing (magnetic, general purpose, and provisioned IOPS)
- Standard data transfer charges
- Automated snapshots – free with 14-day retention
- Manual snapshots – per GB



Elasticsearch Service

ElasticSearch Service



Demo

Timestream

When the order of time matters the most!

Amazon Timestream – Overview

TIME-SERIES

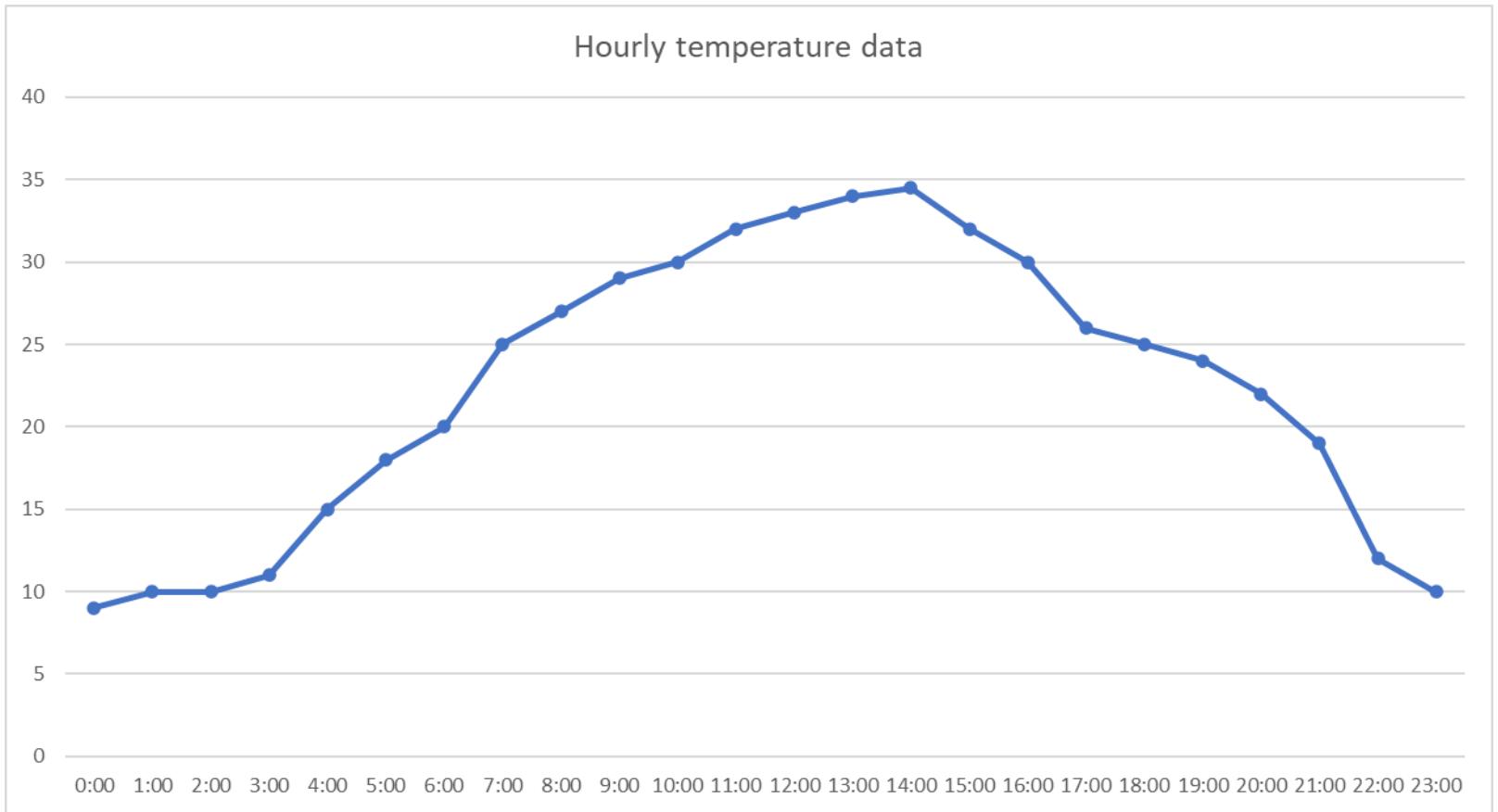


Timestream

- Fully managed, fast, scalable, serverless time-series database service
- Time-series data measures how things change over time
- Store and analyze trillions of time-ordered events per day
- 1,000x faster query performance at 1/10th the cost of relational databases
- Automates rollups, retention, tiering, and compression of data
- Use cases:
 - Log data for DevOps
 - Sensor data for IoT applications
 - Customer clickstream data in web traffic monitoring
 - Industrial telemetry data for equipment maintenance

What is a time series

- Series of data points ordered in time



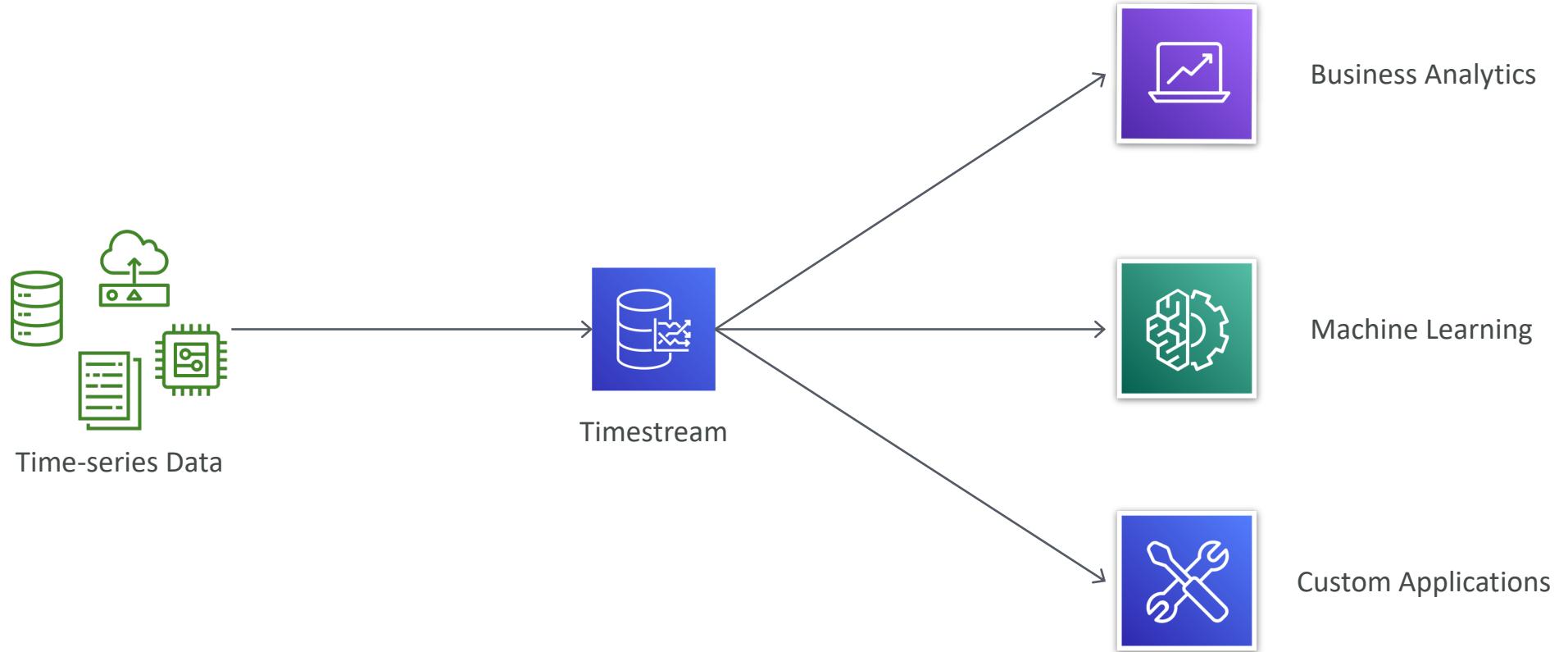
Timestream

- It's a serverless offering
- Automatically scales writes, storage, and query capacity on demand
- Can set the data retention policy for each table
- Data is organized by time intervals
- Uses time-series specific data compression (=less scanning, faster performance)
- Separate data processing layer for data ingestion, storage sharing and queries
- Uses an adaptive query engine that adapts to different time-series datasets
- Offers built-in analytics (interpolation / smoothing / approximation)



Timestream

Timestream - Use cases



Timestream Pricing

- You pay only for what you use
- Writes – per million writes
- Reads (query) – per TB of data scanned by queries
- Storage – per GB (in-memory / SSD / magnetic)
- Data transfer



Timestream

QLDB

Immutable system-of-record that you can trust!



DataCumulus | RIZMAXed

Amazon QLDB – Overview

LEDGER

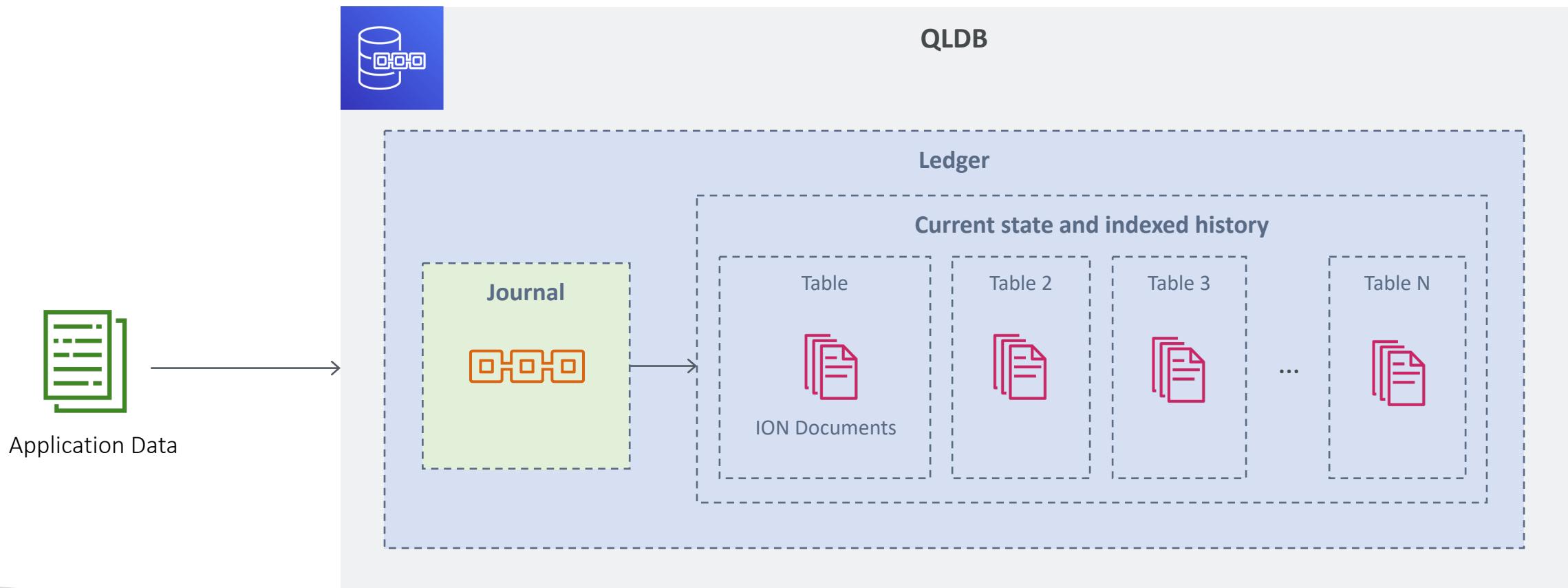


QLDB

- QLDB = Quantum Ledger Database
- Fully managed, serverless ledger database
- Has built-in, immutable journal to record all the change history of your data
- Transparent and cryptographically verifiable ledger
- Tracks each application data change and maintains a complete and verifiable history of changes over time
- Supports ACID transactions
- Uses query language named **PartiQL** (SQL-like, Open standard)
- Uses **Amazon ION** format
 - A Superset of JSON
 - Self-describing, hierarchical data serialization format
 - Offers interchangeable binary and text representations
 - Adds additional data types, type annotations and comments to JSON format
 - Supports nested JSON elements
- Use cases: System of record applications like Banking transactions, HR services records, Insurance claim histories, Vehicle ownership records etc.

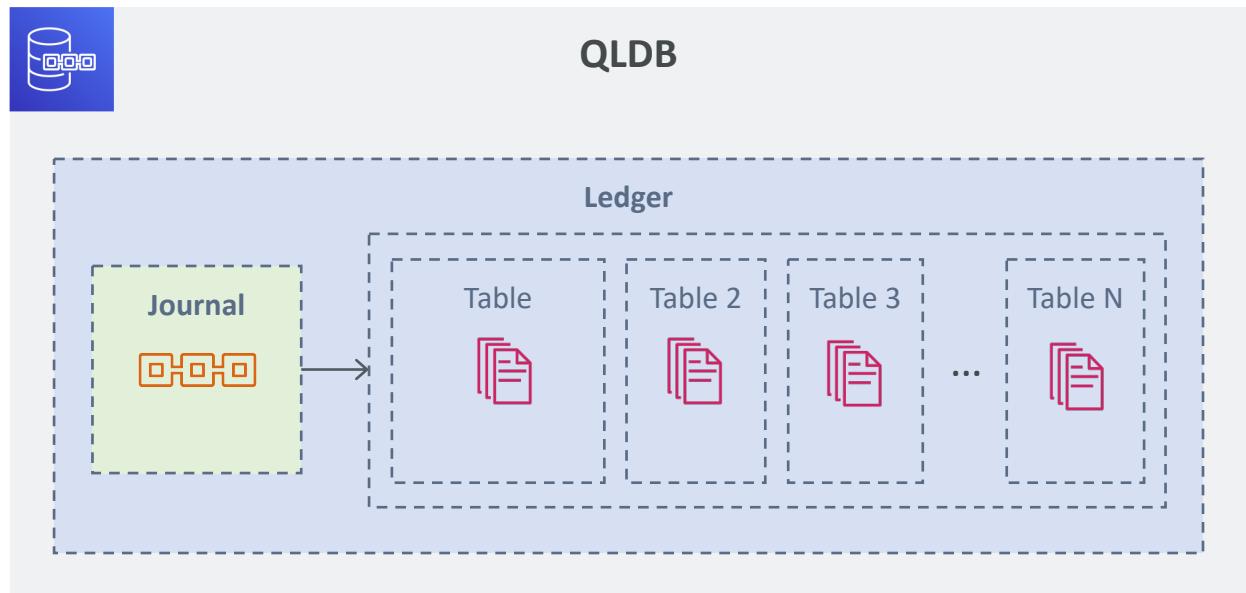
QLDB Architecture

- It's serverless (scales automatically to support the needs of your application)
- Intended to support high-performance OLTP workloads



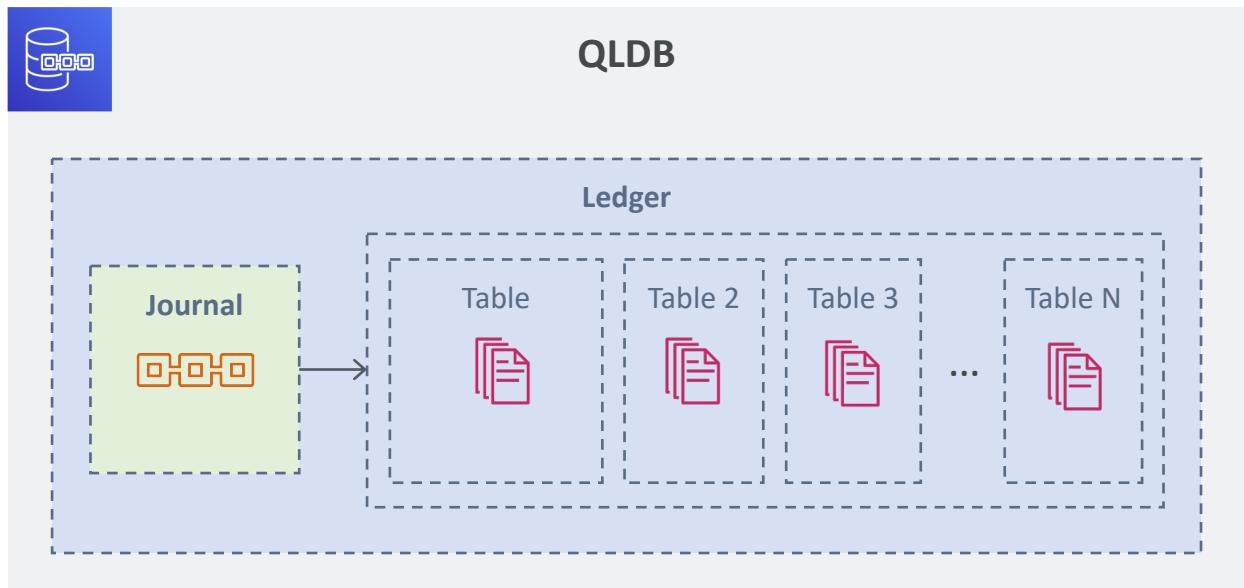
QLDB Architecture

- Ledger = Journal + set of tables
- Journal
 - Is append-only, immutable
 - No updates / overwrites / deletes
 - Stores a sequenced, cryptographically verifiable entry of each change to the table data
 - Changes are chained together as blocks (but not a blockchain implementation)
 - QLDB is centralized and not a distributed ledger (blockchain is used with decentralized use-cases)
 - Even if you delete data from the ledger (table), you can access its change history from the immutable journal



QLDB Architecture

- Tables
 - Collection of documents and their revisions
 - Store the current and historical state of your data (indexed storage)
 - Can include document deletion records
 - Documents are in ION format



Relational vs Ledger

Relational	Ledger (QLDB)
Database	Ledger
Table	Table
Index	Index
Table row	Document (ION format)
Column	Document attribute
SQL	PartiQL
Audit Logs	Journal

QLDB Views

- QLDB offers three views of your data
 - User view
 - Committed view
 - History view
- User view
 - latest version of your data
 - default view
- Committed view
 - user view + system generated metadata
- History view
 - contains all historical document revisions
 - i.e. all change history with metadata

```
SELECT * FROM  
VehicleRegistration
```

User view (default)

```
SELECT * FROM  
_ql_committed_VehicleRegistration
```

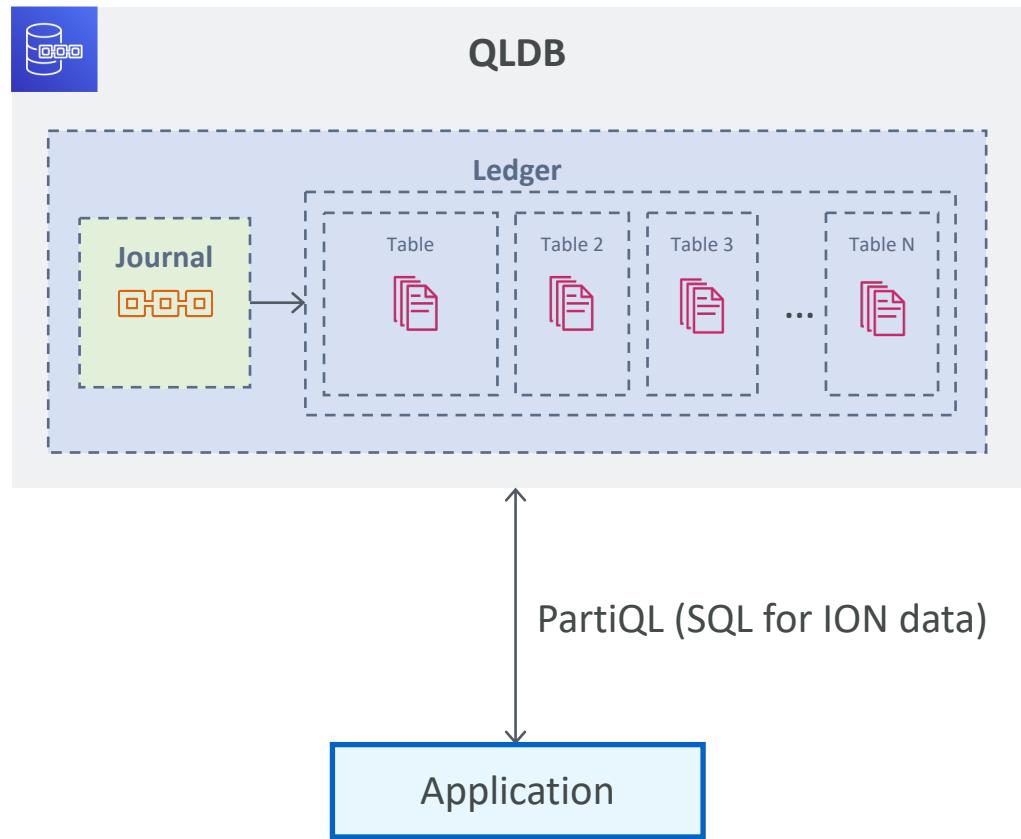
Committed view

```
SELECT * FROM  
history( VehicleRegistration ) AS h  
WHERE h.metadata.id = '11sPfkrm01US0X02Y'
```

History view

Working with QLDB

- You create a ledger and define your tables
- QLDB supports ACID semantics
- We use PartiQL query language to query QLDB
 - It's a SQL-like open standard query language
 - SQL-compatible access to relational, semi-structured, and nested data
 - Extends SQL to support ION documents
 - PartiQL is also used with Redshift / S3 Select / Glacier Select



Amazon ION format

- Is a Superset of JSON
- Self-describing, hierarchical data serialization format (=nested JSON)
- Offers interchangeable binary and text representations
- Adds additional data types, type annotations and comments to JSON format
- Flexible data model

```
1  /* Ion supports comments. */
2 <{
3     // Field names don't always have to be quoted
4     VIN: "1N4AL11D75C109151",
5     MfgDate: "2011-08-01T",
6     Type: "Sedan",
7     Mfgr: "Audi",
8     Model: "A5",
9     Color: "Silver",
10    Specs: {
11        CC: 3000,
12        Weight: 5895
13    }
14 }
```

Data Verification in QLDB

- Journal maintains immutable and verifiable transaction log
- QLDB uses a **digest** for verification
- Digest
 - is a cryptographic representation of your journal
 - or a unique signature of your data's entire change history as of a point in time
 - is generated using SHA-256 hash function with a Merkle tree-based model
- Can verify the integrity of your data by calculating the digest and comparing it with QLDB's digest
- Can verify using the AWS console or QLDB API
- Improper verification requests typically result in `IllegalArgumentException`

```
{  
  "digest": "7BQXP7GRCXlafN+z8UvjvLbQEaYomX+WdVLqDcvFjw=",  
  "digestTipAddress": "{strandId: \"77o9ii8SeAZDOaNu5tyGXZ\", sequenceNo: 84}",  
  "ledger": "vehicle-registration",  
  "date": "2020-06-25T13:49:57.922Z"  
}
```

Digest calculated
`7BQXP7GRCXlafN+z8UvjvLbQEaYomX+WdVLqDcvFjw=` 

Digest
`7BQXP7GRCXlafN+z8UvjvLbQEaYomX+WdVLqDcvFjw=` 

Ledger: vehicle-registration
Digest tip address:
`{strandId: "77o9ii8SeAZDOaNu5tyGXZ", sequenceNo: 84}`
Date: 2020-06-25T13:49:57.922Z

 **Verified: Digest calculated matches Digest**

Creating QLDB Ledger



Demo

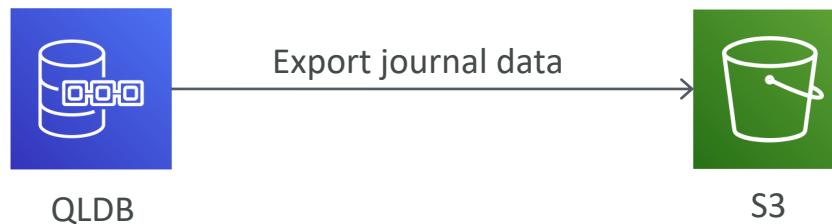
Data verification in QLDB



Demo

QLDB Backup and Restore

- QLDB does not support a backup and restore feature (yet!)
- PITR is also not supported (yet!)
- Can only export your QLDB journal to S3
 - For analytics / auditing / data retention / verification / exporting to other systems
 - limit of two concurrent journal export jobs



No Backup / Restore / PITR

Amazon QLDB > Create export job

Create export job

Choose the date range of journal blocks that you want to export from a ledger. QLDB will write the data into an S3 bucket. [Learn more](#)

Journal blocks

Ledger: vehicle-registration

Start date and time (UTC): 2020/06/24 hh:mm:ss

End date and time (UTC): 2020/06/25 hh:mm:ss

Write journal blocks to S3

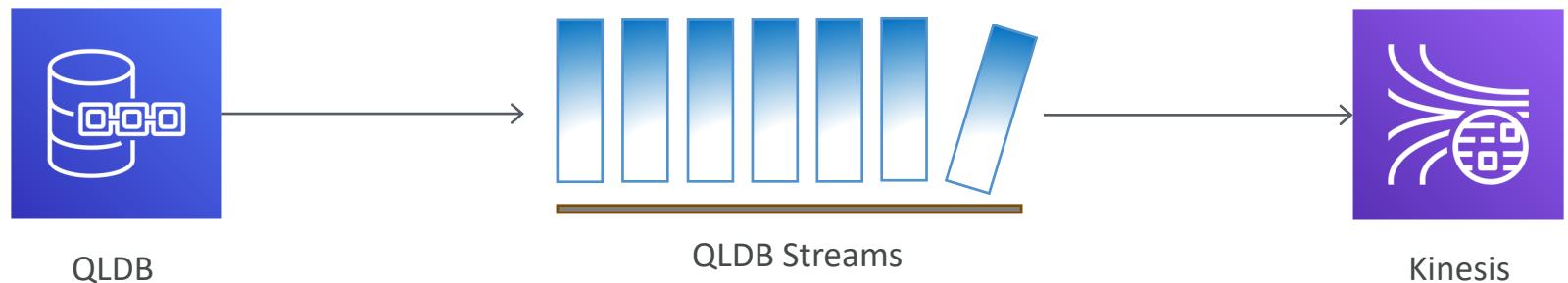
Destination for journal blocks: Enter a destination in Amazon S3 where your journal blocks will be stored. Amazon S3 is object storage built to store and retrieve any amount of data from anywhere. [Learn more](#)

S3Uri: s3://my-exports-bucket

[View](#) [Browse S3](#)

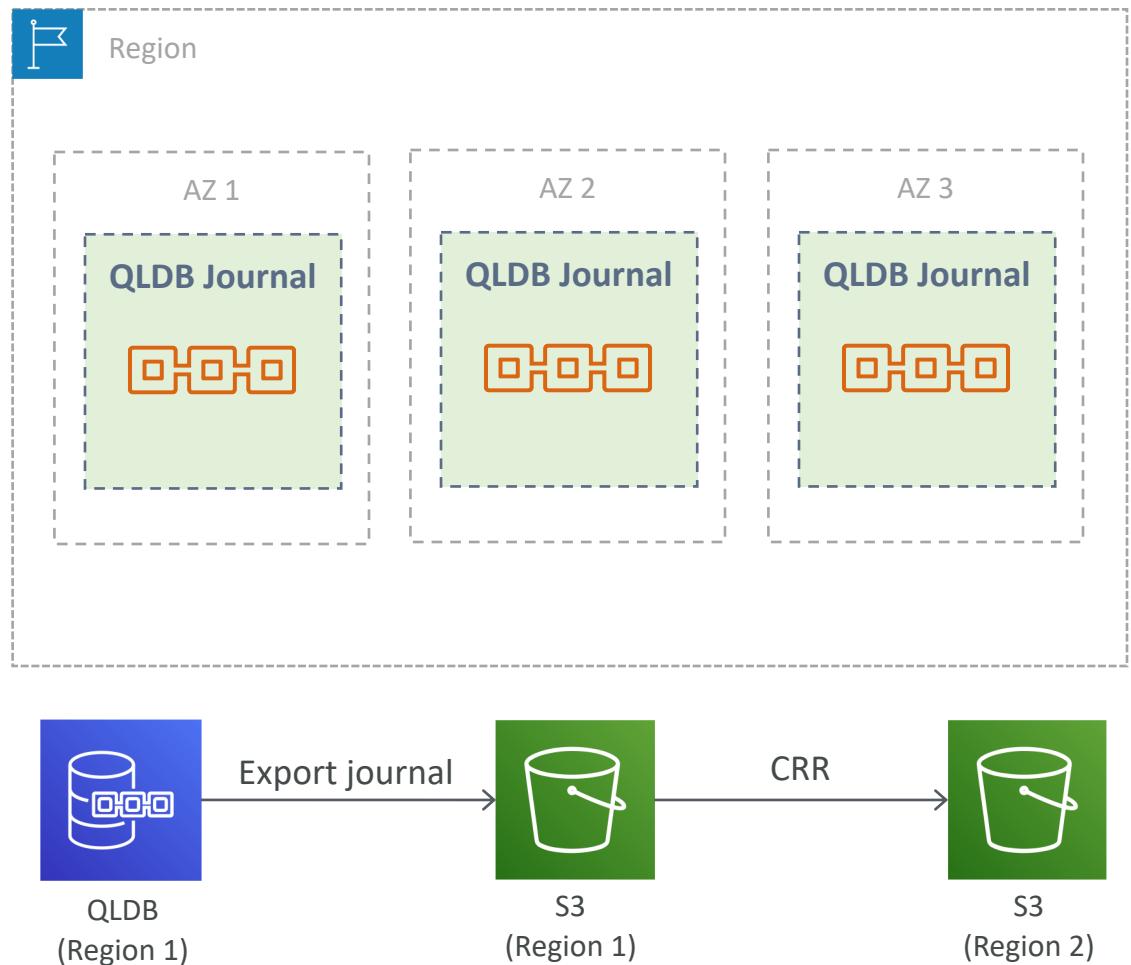
QLDB Streams

- Continuous flow of data from your ledger's journal to a Kinesis data stream
- Provides an at-least-once delivery guarantee
- No ordering guarantees
 - Revisions can be produced in a Kinesis data stream out of order



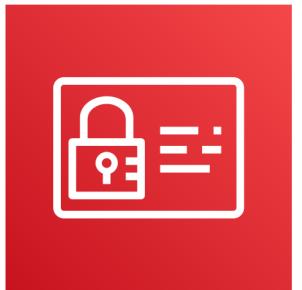
QLDB High Availability and Durability

- QLDB ledger is replicated across multiple AZs within the region (=high availability)
- With multiple copies per AZ (=strong durability)
- Write is acknowledged only after being written to a durable storage in multiple AZs
- CRR is not supported (yet!)
 - QLDB journal can be exported to an S3 bucket
 - S3 bucket can then be configured for CRR



QLDB Security

- IAM is used for authentication and authorization of QLDB resources
- Supports encryption at rest and in transit
- Uses Amazon-owned keys to encrypt QLDB data
- Does not support CMKs



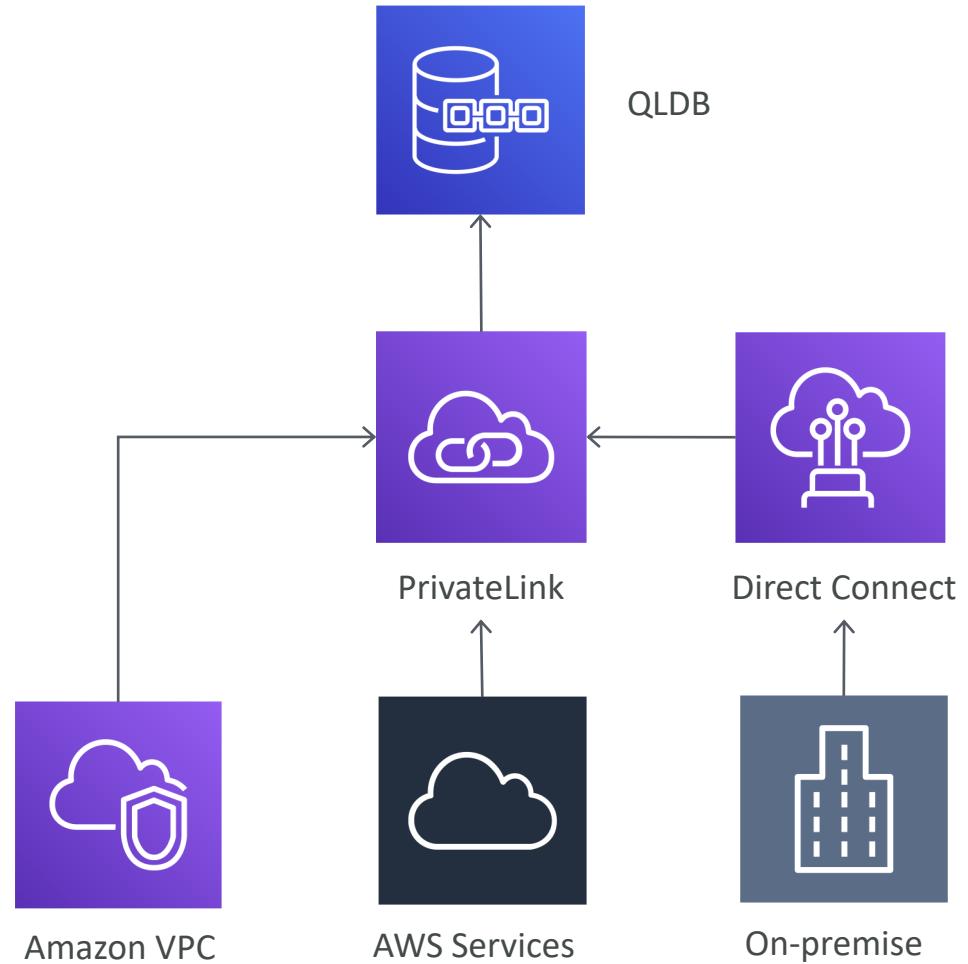
IAM



KMS

QLDB Security - Networking

- Can use an interface VPC endpoint to allow VPC resources to connect to QLDB privately
- Interface VPC endpoints are powered by AWS PrivateLink
- PrivateLink provides private and secured connectivity between VPCs, AWS services, and on-premises applications
- PrivateLink eliminates the need for IG / NAT device /VPN connection / or AWS Direct Connect connection



QLDB Monitoring

- Integrated with CloudWatch (Alarms / Logs / Events)
- Common metrics
 - JournalStorage
 - IndexedStorage
 - ReadIOs
 - WriteIOs
 - CommandLatency
- QLDB log files provide additional information
- API calls and user activity can be logged with CloudTrail



QLDB

QLDB Pricing

- You pay only for what you use
- Storage – per GB per month
 - Journal Storage and Indexed Storage
- IOs – per million requests
 - read IOs and write IOs
- Data transfer



QLDB

Keyspaces

Cassandra on the cloud!

Amazon Keyspaces – Overview

KEY-VALUE



Keyspaces

- A scalable, highly available, and fully-managed database service
- Lets you run your Cassandra workloads on AWS
- Cassandra is an open-source, wide-column, NoSQL data store
- Is serverless, so you pay for what you use + autoscaling
- Supports thousands of requests per second with virtually unlimited throughput and storage
- Compatible with the CQL (Cassandra Query Language) API
- Security through IAM, VPC and KMS
- Data is encrypted by default, supports encryption at rest and in transit
- Supports continuous backups with PITR
- All writes replicated three times across multiple AZs for durability and availability
- Offers 99.99% availability SLA within Region with no scheduled downtime
- Monitoring through CloudWatch, DDL actions logged with CloudTrail
- Use cases: IoT device metadata / User profiles / Time-series data / Transactions data (e.g. ecommerce)

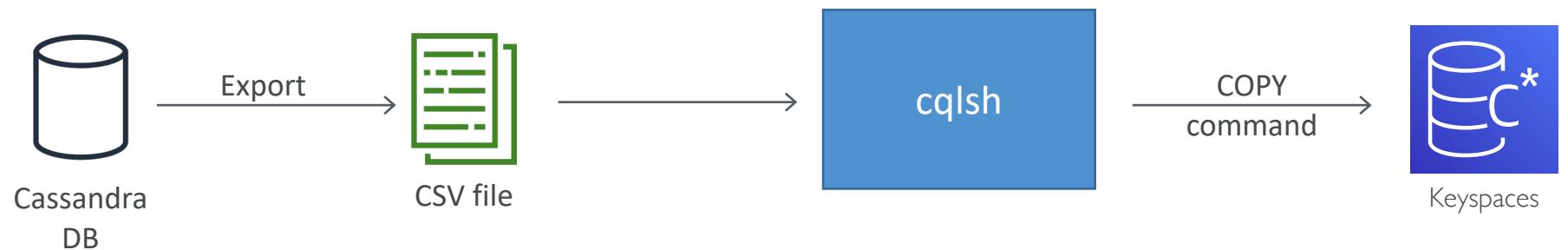
CQL (Cassandra Query Language)

- You use CQL for interacting with Cassandra database (and with Keyspaces)
- To run CQL queries, you can use:
 - CQL editor in the AWS Console
 - cqlsh client (CQL shell)
 - Cassandra client driver (programmatic access)



Migrating from Cassandra to Keyspaces

- Export existing cluster data to CSV files
- Import using cqlsh COPY command



Read and write consistency in Keyspaces

- Two read consistency modes:
 - LOCAL_ONE consistency
 - LOCAL_QUORUM consistency
- LOCAL_ONE optimizes for performance and availability by returning the first returned value from any storage replica
- LOCAL_QUORUM optimizes for data correctness by requiring at least two replicas to return a value before it is returned to your application
- All writes use LOCAL_QUORUM for durability

Keyspaces pricing

On-demand mode

- Uses RRUs and WRUs (read/write request units)
- You pay for the actual reads and writes
- Use with unpredictable application traffic
- 1 RRU = one 4KB read with LOCAL_QUORUM consistency
- 1 RRU = two 4KB reads with LOCAL_ONE consistency
- 1 WRU = one 1KB write with LOCAL_QUORUM consistency
- If a query returns multiple rows, you are billed based on the aggregate size of the data returned
- For example, if your query returns four rows and each row has 2 KB of data (8 KB of data total), you are billed 2 RCUs using LOCAL_QUORUM consistency and 1 RCU using LOCAL_ONE consistency
- Storage, backups and restore, and data transfer costs are additional

Provisioned mode

- Uses RCUs and WCUs (read/write capacity units)
- You specify the number of reads and writes per second
- Lets you optimize costs if you have predictable application traffic and can forecast capacity requirements in advance
- 1 RCU = one 4KB read with LOCAL_QUORUM consistency
- 1 RCU = two 4KB reads with LOCAL_ONE consistency
- 1 WCU = one 1KB write with LOCAL_QUORUM consistency

Working with Keyspaces



Demo

Comparing AWS Databases

There're so many options, which one is right for me?

Comparison of AWS Databases

Database	Type of data	Workload	Size	Performance	Operational overhead
RDS DBs	Structured	Relational / OLTP / simple OLAP	Low TB range	Mid-to-high throughput, low latency	Moderate
Aurora MySQL	Structured	Relational / OLTP / simple OLAP	Mid TB range	High throughput, low latency	Low-to-moderate
Aurora PostgreSQL	Structured	Relational / OLTP / simple OLAP	Mid TB range	High throughput, low latency	Low-to-moderate
Redshift	Structured / Semi-structured	Relational / OLAP / DW	PB range	Mid-to-high latency	Moderate
DynamoDB	Semi-structured	Non-relational / Key-Value / OLTP / Document store	High TB range	Ultra-high throughput, low latency, ultra-low latency with DAX	Low
ElastiCache	Semi-structured / Unstructured	Non-relational / In-memory caching / Key-Value	Low TB range	High throughput, ultra-low latency	Low
Neptune	Graph data	Highly connected graph datasets	Mid TB range	High throughput, low latency	Low

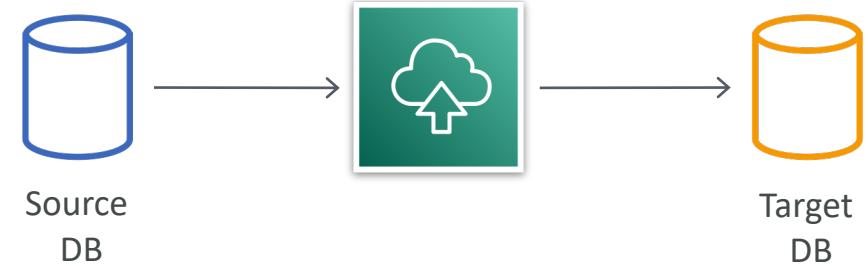
https://youtu.be/asx_VOUTxaU?t=579

Database Migration, DMS and SCT

Who said database migrations were one-time activity? They now run 24x7!

Why database migration? (Use cases)

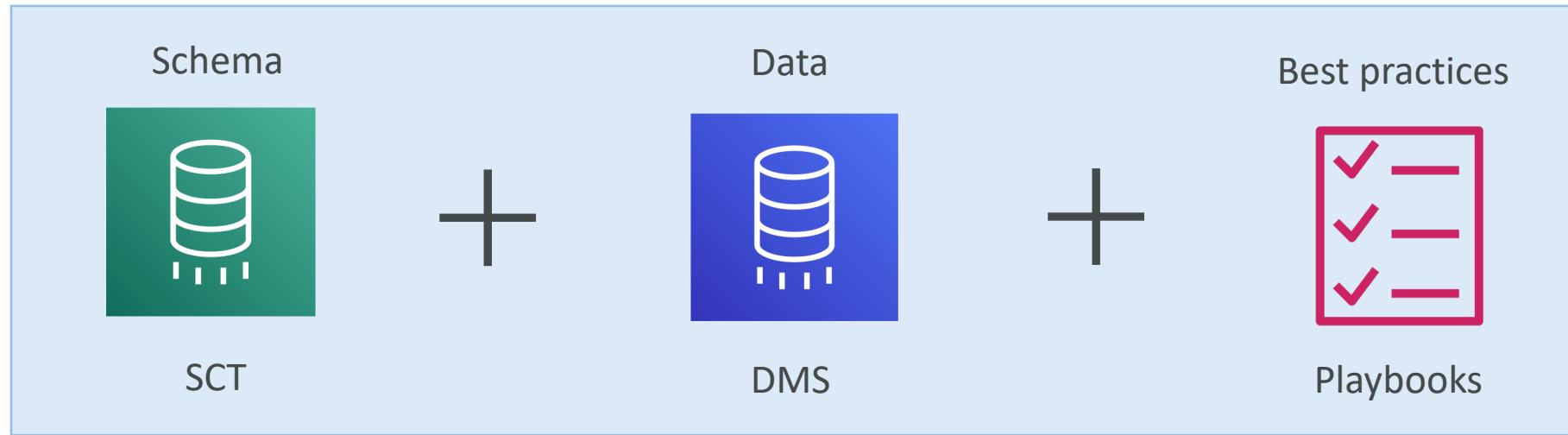
- Move from legacy to cloud
- Switch the database solution
- Replicate databases across regions
- Replicate to streaming platforms
- In place DB upgrades
- Archive data to S3



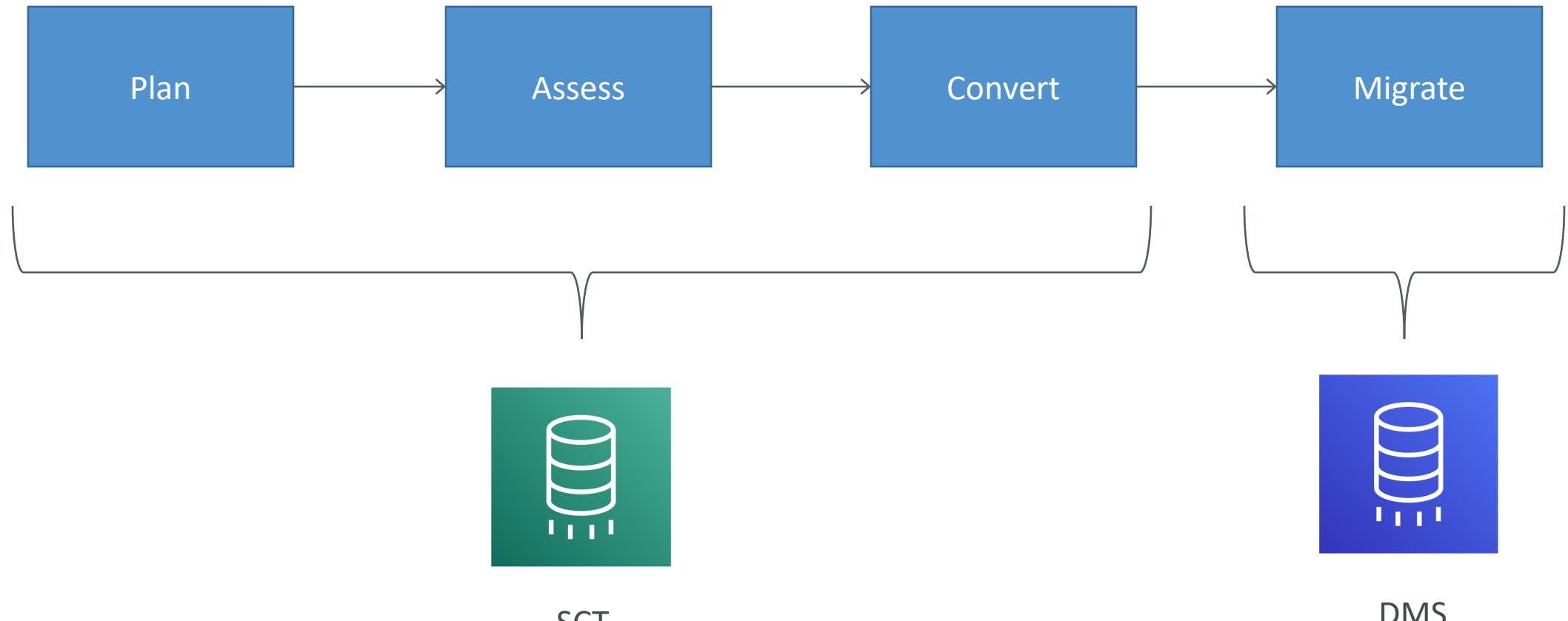
- Migration scenarios:
 - On-premise → AWS cloud
 - Relational → Non-relational
 - DBs hosted on EC2 → Managed AWS services

Migration tools

- Native tools (e.g. mysqldump)
- SCT (Schema Conversion Tool)
- AWS DMS (Database Migration Service)
- Migration Playbooks (migration templates + best practices)

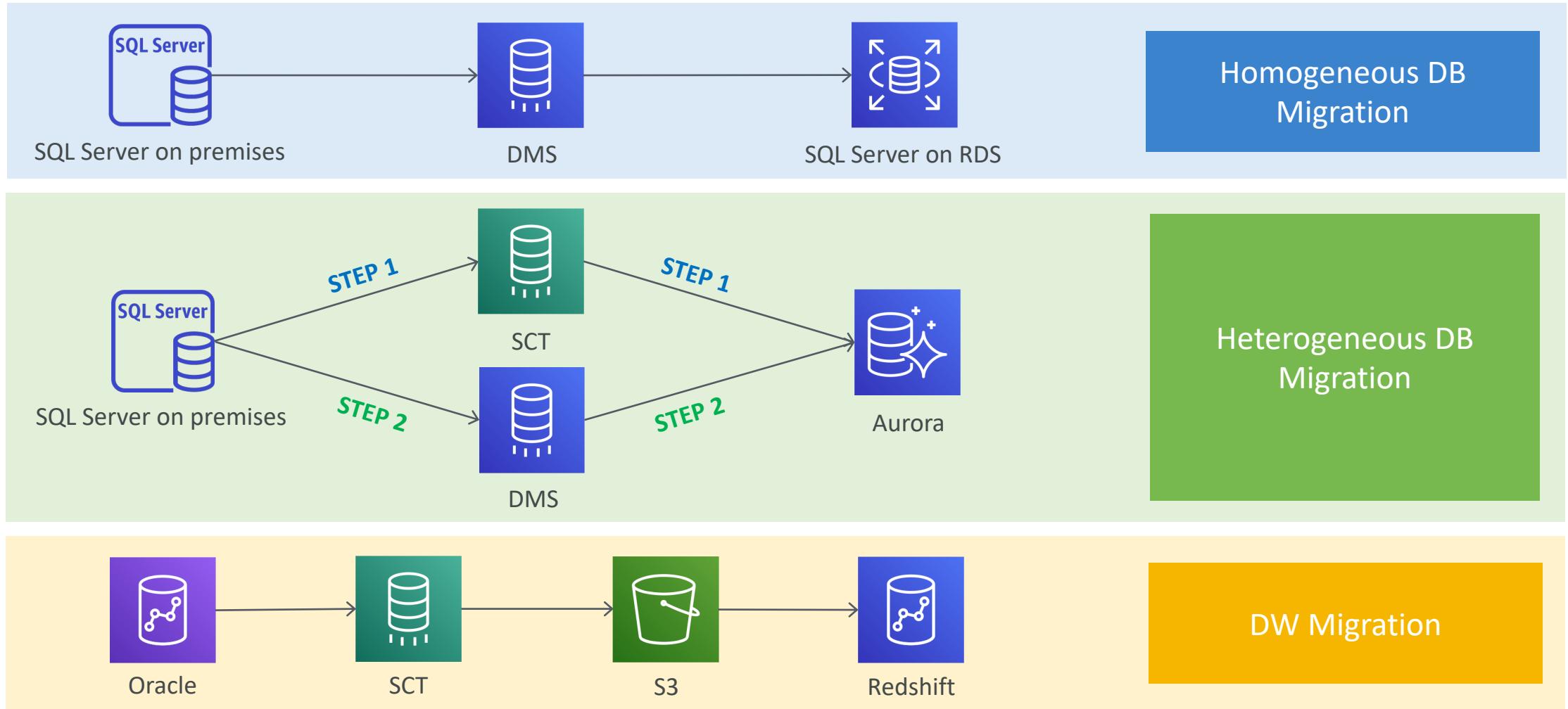


Migration process

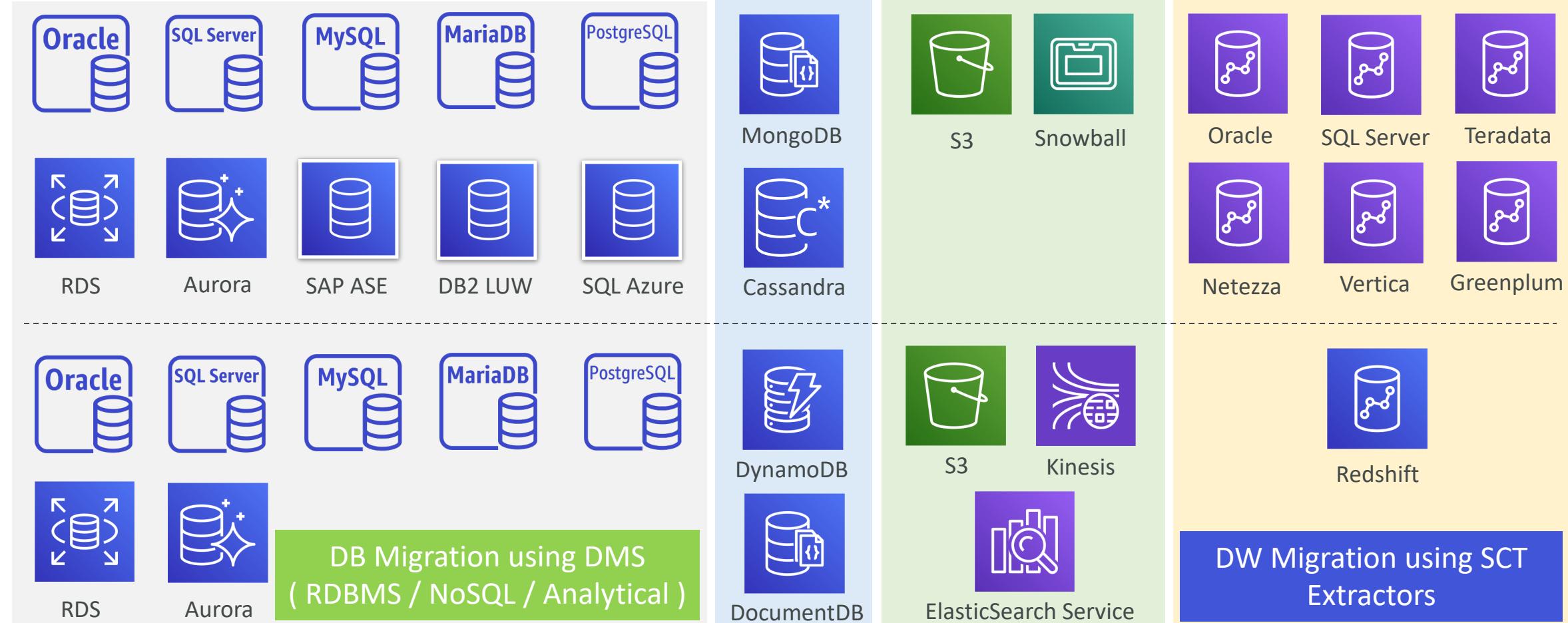


* Generic representation

Migration process flow

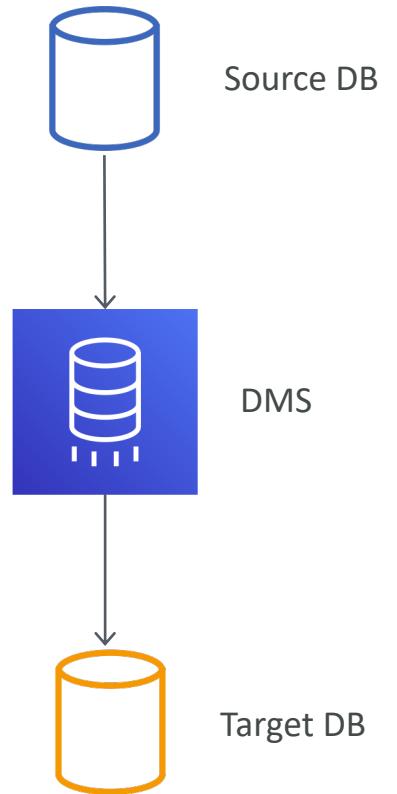


DMS Sources and Targets

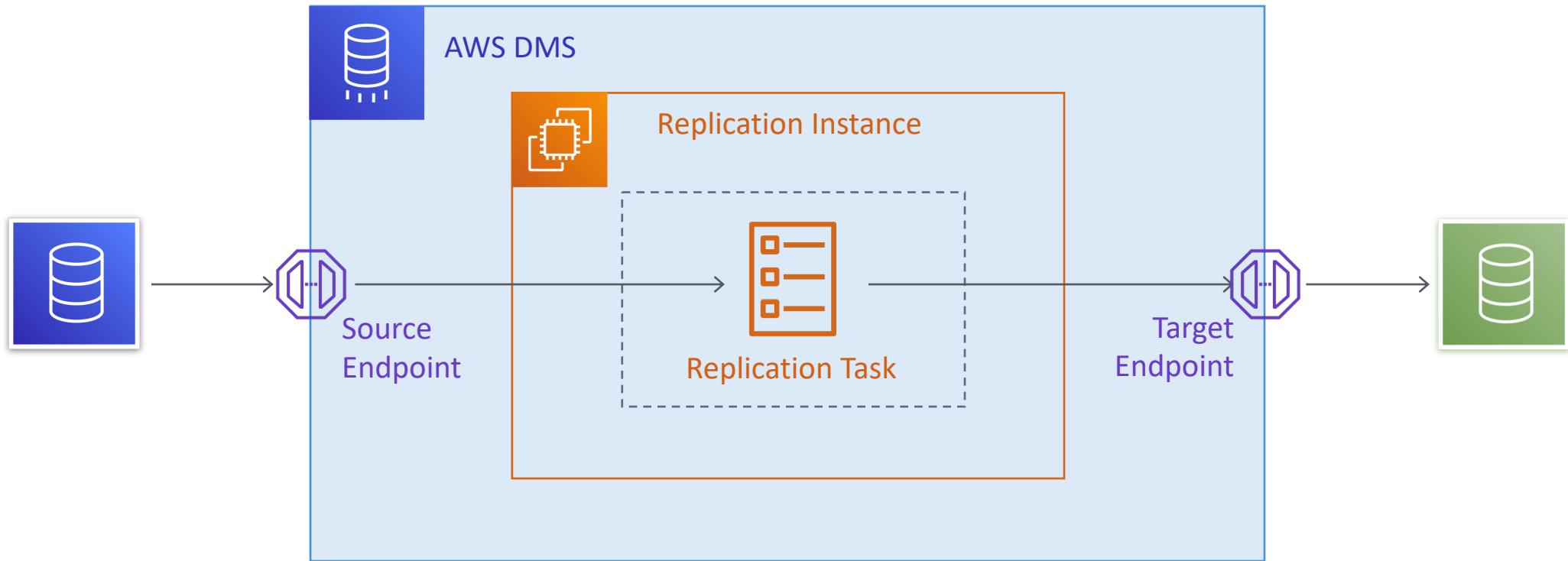


DMS – Database Migration Service

- Quickly and securely migrate databases to AWS cloud, resilient, self healing
- The source database remains available during the migration
- Minimal to zero-downtime
- Costs as low as \$3 / TB
- Supports homogeneous and heterogeneous migrations
- Continuous data replication using CDC (Change data capture)



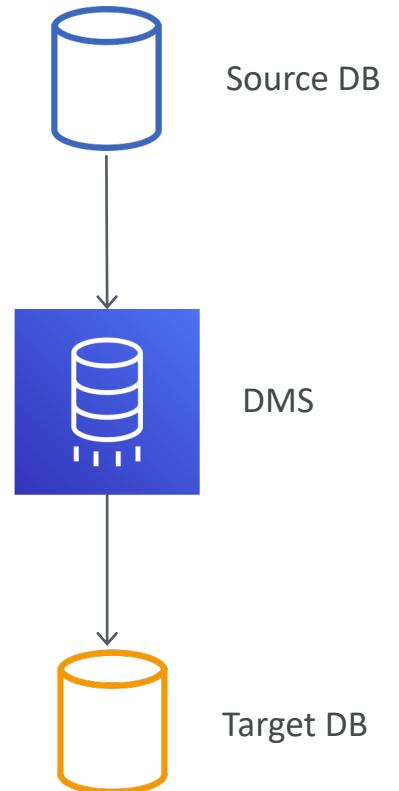
DMS architecture



- Endpoints include DB connection and credentials information
- Replication instance is an EC2 instance running in a VPC, can be **Multi-AZ**
- Replication task specifies actual data tables to migrate and data transformation rules

DMS – Database Migration Service

- DMS can create target tables / primary keys if they don't exist on the target
- Tables/keys can also be created manually
- Creates only those objects required that are required for efficient migration (e.g. primary keys / unique indexes etc.) – This is called as Basic Schema Copy
- Remaining DB elements/schema can be migrated using other tools (e.g. secondary indexes, FK constraints, or data defaults)
 - e.g. use SCT or native schema export tools



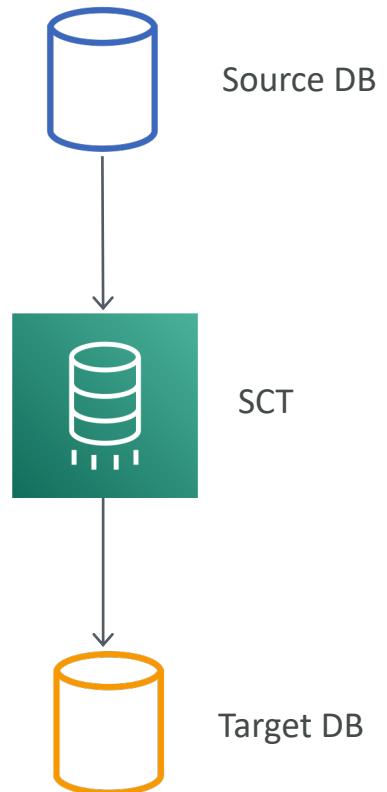
Migration with DMS in action



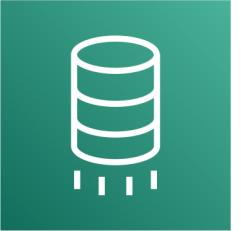
Demo

SCT – AWS Schema Conversion Tool

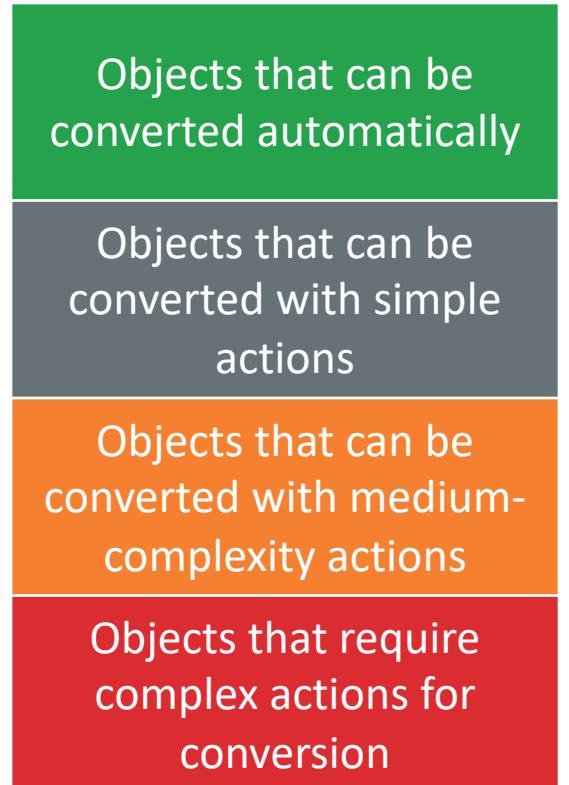
- Converts DB/DW schema from source to target (including procedures / views / secondary indexes / FK and constraints)
- Mainly for heterogeneous DB migrations and DW migrations
- For homogeneous migrations, can use it for migration assessment
- Can be installed on EC2 or locally on your computer (closer to source DB)
- Application conversion – can convert SQL statements embedded in the application code
- Script conversion – can convert ETL scripts (Oracle / Microsoft / Teradata scripts → Aurora / Redshift scripts)
- Can optimize schemas in Redshift (recommends distribution and sort keys, different from native **Redshift advisor**)



SCT – AWS Schema Conversion Tool



- Provides assessment report
 - Granular report showing which objects can be converted automatically and which need manual intervention (with color codes)
 - Can be used to create a business case for migration
 - Identifies issues / limitations / actions for schema conversion
- For objects that cannot be created manually, SCT provides guidance to help you create the equivalent schema manually



Workload Qualification Framework (WQF)

- Standalone app that is included with SCT
- For workload assessment
- Qualifies OLTP workloads based on
 - Ease / complexity of migration
 - Estimated time / effort required
- Recommends migration strategy and tools
- Integrated with SCT and DMS
- Just provide DB connection strings and WQF can generate the assessment reports
- Available as EC2 AMI

<https://aws.amazon.com/blogs/database/classify-oltp-and-olap-workloads-using-aws-sct-with-workload-qualification-framework-wqf/>

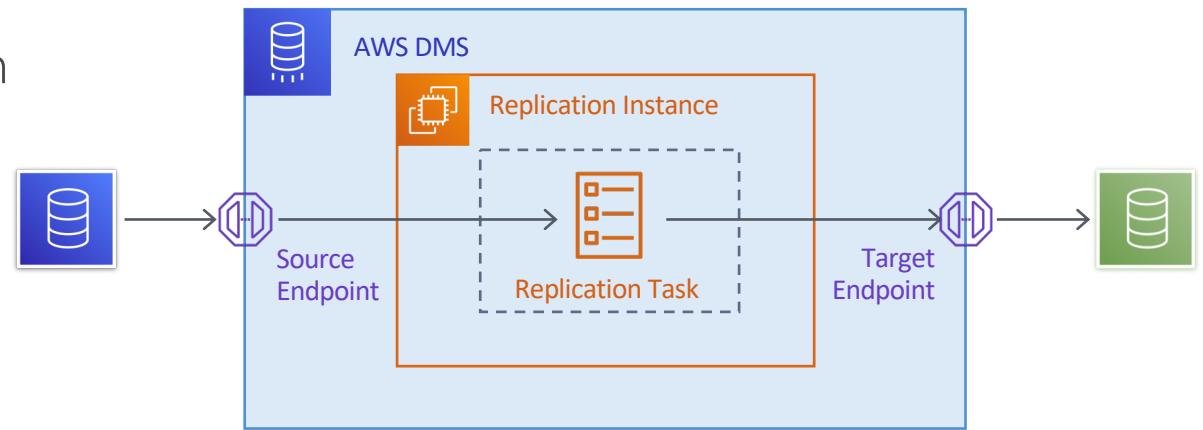
<https://docs.aws.amazon.com/prescriptive-guidance/latest/database-migration-strategy/qualify-workloads.html>

WQF Assessment Report		
Workload category	Workload Category 3: Heavy proprietary feature workloads. 0 significant problems were found.	
Analysis	Migration to Amazon RDS for MySQL	Migration to Amazon RDS for PostgreSQL
Summary of Object Analysis	Low complexity	Low complexity
Database object analysis	We assessed the modules below and here is an aggregated view of the analysis per module: 1. Databases: We analyzed and assessed 0 databases along with all objects in it. 2. Schemas: We analyzed and assessed 2 schemas in total. More detail about each schema object can be found in the detail reports for each engine. 3. Application code modules: We analyzed 1 applications in total. More details about each application can be found in detail reports for each engine.	We assessed the modules below and here is an aggregated view of the analysis per module: 1. Databases: We analyzed and assessed 0 databases along with all objects in it. 2. Schemas: We analyzed and assessed 2 schemas in total. More detail about each schema object can be found in the detail reports for each engine. 3. Application code modules: We analyzed 1 applications in total. More details about each application can be found in detail reports for each engine
Application code analysis	We completed the analysis of your database servers. We found 144 objects in 2 schemas and their average compatibility is 68%. There were 434 Action items found. For more details please see tab 'Migration Efforts for MySQL'.	We completed the analysis of your database servers. We found 144 objects in 2 schemas and their average compatibility is 77%. There were 385 Action items found. For more details please see tab 'Migration Efforts for PG'.
Physical object analysis	We completed the analysis of your application code and estimate that 100% of application code can be moved to Amazon, and connected to Amazon RDS for MySQL. After analyzing your physical objects we found that 100% can be moved to EC2.	We completed the analysis of your application code and estimate that 100% of application code can be moved to Amazon, and connected to Amazon RDS for PG. After analyzing your physical objects we found that 100% can be moved to EC2.
Summary of Migration Effort	Based on the above summary and detail, we estimate that this conversion and migration will take 35 days to complete with a 16 person team. You can get more details on this and change required inputs to look at different effort scenarios in the 'Migration Efforts to MySQL' tab.	Based on the above summary and detail, we estimate that this conversion and migration will take 28 days to complete with a 16 person team. You can get more details on this and change required inputs to look at different effort scenarios in the 'Migration Efforts to PG' tab.

Migration Steps\Roles	WQF Assessment Report									
	Migration Architect		Database Developer		Application Developer		Application QA		Project Manager	
Head Count	Days	Head Count	Days	Head Count	Days	Head Count	Days	Head Count	Days	
1. Envisioning/Assessment	2	2	0	0	0	0	0	1	0	5
2. Database schema conversion	0		5	18	0		0	1	1	90
3. Application conversion/remediation	0		3	0	2	0	0	1	0	0
4. Script conversion	0		3	15	0		0	1	4	49
5. Integration with 3rd party applications	0		0	0	0		0	1	0	0
6. Data migration	0		2	9	0		0	1	2	20
7. Functional testing of the entire system	0		0	0	0		4	11	1	45
8. Performance tuning	0		4	3	0		0	1	1	15
9. Integration and deployment	0		0	0	0		0	1	0	0
10. Training and knowledge transfer	2	2	0	0	0		0	1	0	5
11. Documentation and version control	1	2	0	0	0		0	1	0	3
12. Post production support	1	3	0	0	0		0	1	0	3
Total (days)										236

DMS tasks (replication tasks)

- Lets you specify migration options
 - Table mapping (transformation rules)
 - Filters (to migrate data selectively)
 - Migration type – full load, CDC, or both
- Supports data validation
- Monitoring options
 - task status
 - task's control table
- Allows reloading table data (in case of errors)



DMS task assessment reports

- For pre-migration assessment (optional, recommended)
- To run an assessment, a task must be in a *Stopped* state
- Lets you know potential issues that might occur during a given migration
- Includes JSON output containing summary and details of unsupported data types

```
1 {
2     "summary": {
3         "task-name": "test15",
4         "not-supported": {
5             "data-type": [
6                 "sql-variant"
7             ],
8             "column-count": 3
9         },
10        "partially-supported": {
11            "data-type": [
12                "float8",
13                "jsonb"
14            ],
15            "column-count": 2
16        }
17    },
18    "types": [
19        ...
20    ]
21 }
```

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Tasks.AssessmentReport.html

DMS migration types

Full Load

- Migrate existing data
- performs a one-time migration from source to target

CDC Only

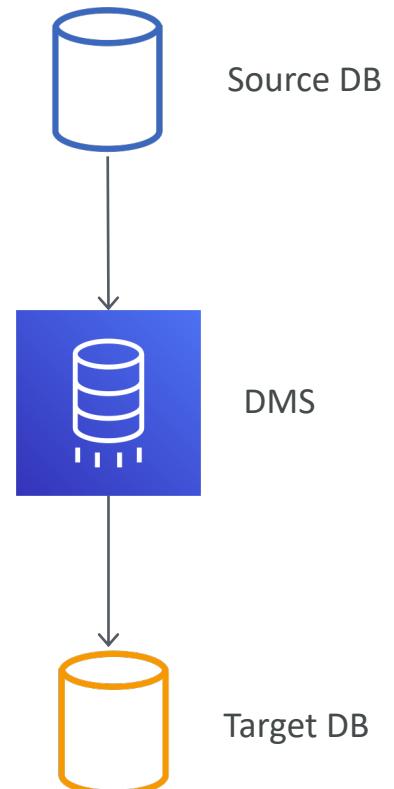
- Ongoing replication
- Only replicates data changes from source to target without migrating existing data
- CDC = Change data capture

Full Load + CDC

- Migrate existing data and replicate ongoing changes
- First, performs a one-time migration from source to target
- Then, continues replicating data changes

DMS – Good things to know

- Recommended to create only the primary keys before full load
- Secondary keys and FKs should be created only after full load is complete
- In full load, multiple tables are loaded in parallel and this can create issues if you have FKs (hence create them later)
- Secondary keys can slow down the full load operation (hence create them later)
- Enable Multi-AZ for ongoing replication (for high availability and failover support)
- DMS can read/write from/to encrypted DBs



Migrating Large Tables

- Break the migration into multiple tasks
- Use row filtering on a key or partition key to create multiple tasks
- Example – if you have integer primary key from 1 to 8000,000
 - Create 8 tasks using row filtering to migrate 1000,000 records each
- Example 2 – if you have a date field as primary key
 - Partition the data by month using row filtering
 - Use full load tasks to migrate data of previous months
 - Use full load + CDC to migrate current month data



DMS Migrating LOBs / CLOBs

- LOB = Large Binary Object
- CLOB = Character Large Object
- DMS migrates LOB data in two phases
 - creates a new row in the target table and populates all data except the LOB data
 - updates the row with the LOB data
- LOB options
 - Don't include LOB columns – LOB data is ignored
 - Full LOB mode – migrates all LOB data, piecewise in chunks (you provide LOB chunk size)
 - Limited LOB mode – truncates each LOB to Max LOB size (is faster)

Include LOB columns in replication [Info](#)

Don't include LOB columns
 Full LOB mode
 Limited LOB mode

LOB chunk size (kb)

64

When using Full LOB Mode LOBS are migrated piecewise in chunks controlled by the LOB chunk size.

Include LOB columns in replication [Info](#)

Don't include LOB columns
 Full LOB mode
 Limited LOB mode

Maximum LOB size (KB) [Info](#)

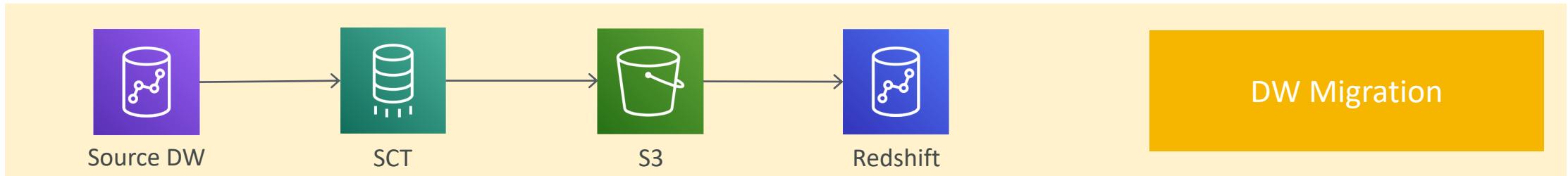
32

Best practices for handling LOBs with DMS

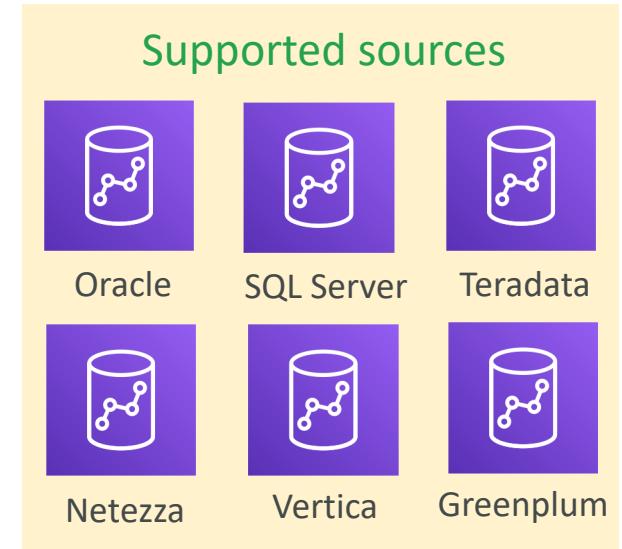
- Full LOB mode performance is slowest, but data is not truncated
- For LOBs < few MBs
 - Use Limited LOB mode with Max LOB size = largest LOB size in your DB
 - Allocate enough memory to the replication instance
- For LOBs > few MBs
 - Create a separate DMS task with Full LOB mode
 - Ideally, separate task on a **new** replication instance
 - LOB chunk size should allow DMS to capture most LOBs in as few chunks as possible
- Inline LOB mode
 - Combines the advantages of both modes (full LOB and limited LOB mode)
 - Migrate without truncating the data or slowing the task's performance
 - You specify **InlineLobMaxSize** (Full LOB mode must be set to true)
 - Small LOBs are transferred inline, large LOBs by using source table lookup
 - Supported only during full load (not during CDC)



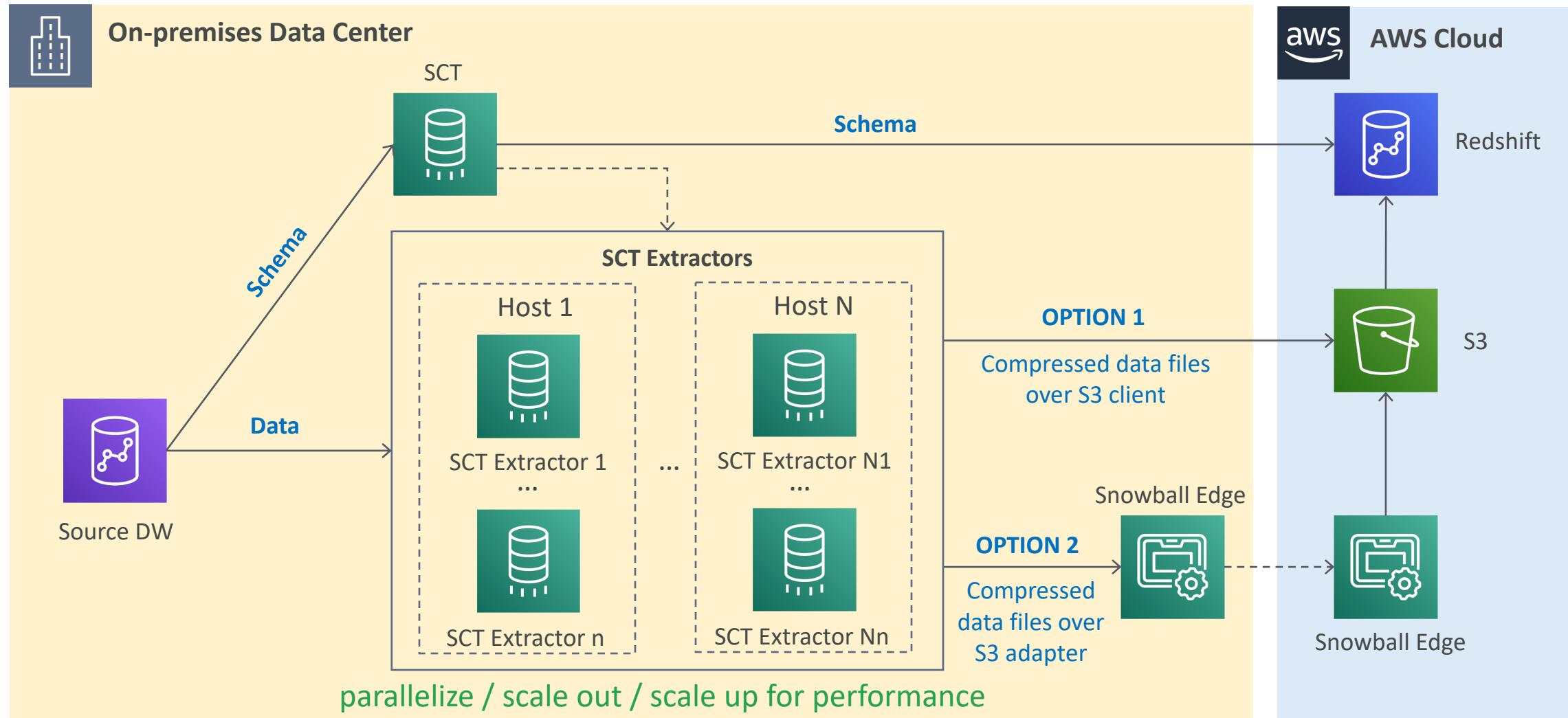
SCT Extractors (=DW migration)



- SCT extractors are migration agents installed locally (or on EC2)
- Extracts data from the source DW in parallel
- Supports encryption
- Data is optimized for Redshift and stored in local files
- Files are then loaded into S3 (using N/W or Snowball Edge)
- Use the COPY command to load data from S3 into Redshift
- If you are using Oracle or SQL Server, you can use DMS to keep your DBs in sync with Redshift / or target engine



DW migration with SCT



Snowball Edge vs Snowmobile

- Both can be used for data migration



Snowball Edge



100 TB storage (80 TB usable)
Transfer within 1 week approx.



Snowmobile

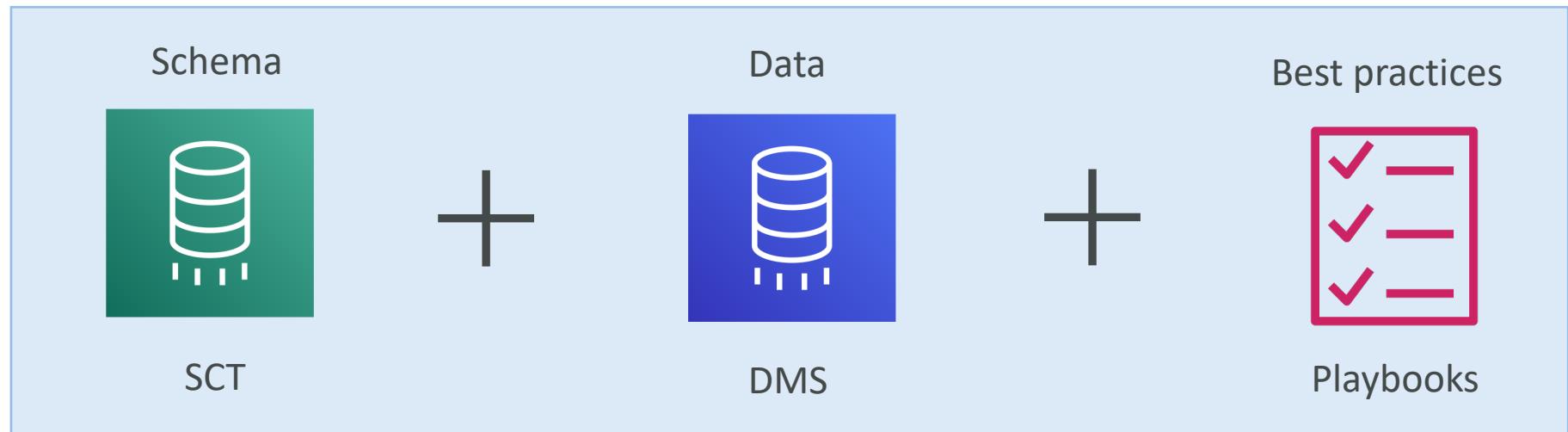


100 PB storage, 45-foot long container
Transfer within few weeks to 6-months approx.

<https://aws.amazon.com/snow/>

Migration playbooks

- Series of step-by-step guides published by AWS
- Provide best practices and blueprints for different heterogeneous migrations
- <https://aws.amazon.com/dms/resources/>



Migration playbooks

MIGRATION PLAYBOOK

Migrate from Microsoft SQL Server to Amazon Aurora MySQL

Migrate your Microsoft SQL Server Database to Amazon Aurora MySQL with minimal downtime.

AWS Database Migration Service, AWS Schema Conversion Tool, Amazon Aurora, Amazon RDS for SQL Server

MIGRATION PLAYBOOK

Migrate from Microsoft SQL Server to Amazon Aurora PostgreSQL

Migrate your Microsoft SQL Server Database to Amazon Aurora PostgreSQL with minimal downtime.

AWS Database Migration Service, AWS Schema Conversion Tool, Amazon Aurora

MIGRATION PLAYBOOK

Migrate from Oracle to Amazon Aurora PostgreSQL

Migrate your Oracle Database to Amazon Aurora PostgreSQL with minimal downtime.

AWS Database Migration Service, AWS Schema Conversion Tool, Amazon Aurora

11 STEPS

Migrate from Oracle to Amazon Aurora MySQL

Migrate your Oracle Database to Amazon Aurora MySQL with minimal downtime.

AWS Database Migration Service, AWS Schema Conversion Tool, Amazon Aurora, Amazon RDS for Oracle

11 STEPS

Migrate from Oracle to Amazon Redshift

Migrate your Oracle Data Warehouse to Amazon Redshift with minimal downtime.

AWS Database Migration Service, AWS Schema Conversion Tool, Amazon Redshift, Amazon RDS for Oracle

Source: <https://aws.amazon.com/dms/resources/>

Service Substitutions

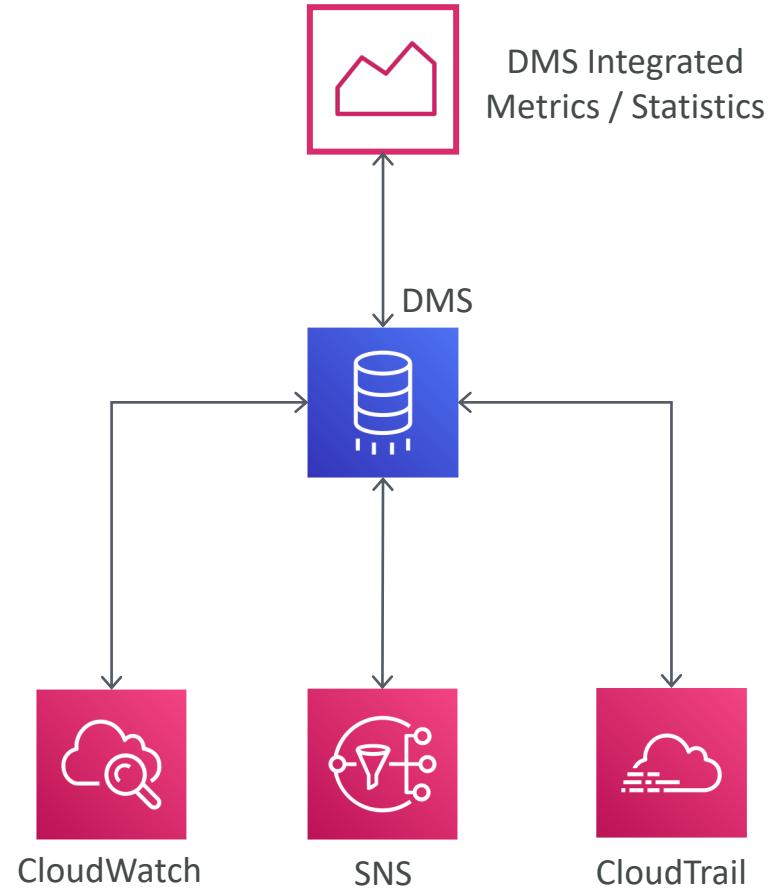
- Service Substitutions are **highlighted** in playbooks
- SQL server – DB mail to send email based on certain events
 - This feature is not available in open source engines
 - We can use Lambda functions through Aurora MySQL along with SNS to emulate the email functionality
- Similar substitutions are possible with queues, file management etc.
- Can use AWS glue to manage ETL pipelines
- SCT can convert your Teradata / Oracle scripts and move them into AWS Glue by auto-generating necessary python code



Playbooks

Monitoring DMS

- Can monitor task progress by:
 - checking the task status
 - using the task's control table
 - or with CloudWatch
- DMS Task Logs and service metrics / statistics are provided by CloudWatch
- Task monitoring tab shows CloudWatch metrics
- Table statistics tab shows statistics for each table
- Can subscribe to event notifications (uses SNS)
- API calls are logged in CloudTrail



DMS Task Logs

- Certain DMS issues / warnings / error messages appear only in the task log
 - e.g. data truncation issues or row rejections due to FK violations are only written to the task log
- Must enable CloudWatch logs while creating replication task



CloudWatch

DMS Validation

- DMS can validate the migrated data for RDBMS migrations
- Supports partial validation of LOBs
- You enable validation in the DMS task settings
- Tracks the progress of the migration and incrementally validates new data as it is written to the target (by comparing source and target data)
- Table must have a primary key or unique index for validation to work
- Requires additional time to complete
- Validation is recommended during CDC (but can also be done during full load)
- SCT extractors do not support validation. Only DMS does.

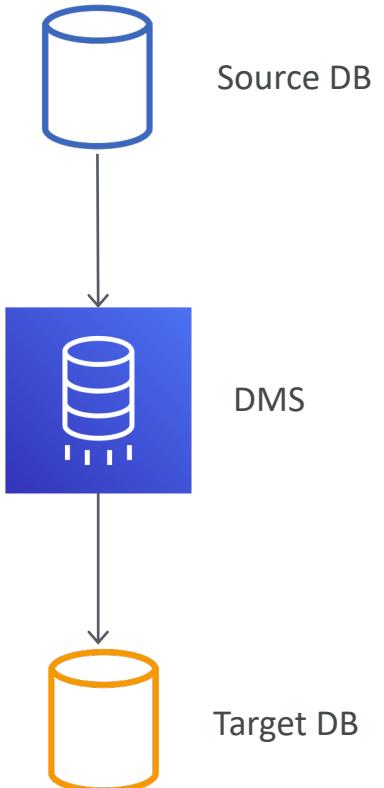


Table statistics

- Table statistics tab shows the table state (performance statistics) for each table that's being migrated
- Use the command
 - **describe-table-statistics** to receive the data validation report in JSON format

```
{  
    "ReplicationTaskArn": "arn:aws:dms:us-west-2:5731014:task:VFPFTYKK2RYSI",  
    "TableStatistics": [  
        {  
            "ValidationPendingRecords": 2,  
            "Inserts": 25,  
            "ValidationState": "Pending records",  
            "ValidationSuspendedRecords": 0,  
            "LastUpdateTime": 1510181065.349,  
            "FullLoadErrorRows": 0,  
            "FullLoadCondtnlChkFailedRows": 0,  
            "Ddls": 0,  
            "TableName": "t_binary",  
            "ValidationFailedRecords": 0,  
            "Updates": 0,  
            "FullLoadRows": 10,  
            "TableState": "Table completed",  
            "SchemaName": "d_types_s_sqlserver",  
            "Deletes": 0  
        }  
    ]  
}
```

https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Validating.html

Task Statistics

- If you enable data validation, DMS provides table-level statistics for the given task
- Indicates the validation state for each table
- You can revalidate tables from Table Statistics tab, if desired
- Validation errors and diagnostic info is written to a table named **awsdms_validation_failures_v1** at the target endpoint
- Example – to troubleshoot validation errors, run this query:

```
SELECT * FROM awsdms_validation_failures_v1 WHERE  
TASK_NAME = 'C89TDNZRYUKH56DR5RGNM'
```

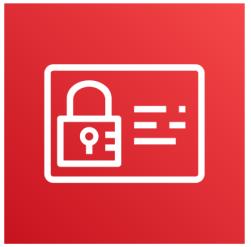
Control tables

- Help with troubleshooting DMS migrations
- Provide useful statistics to help you plan and manage current / future migration tasks
- Can be created when creating the replication task
- Common control tables:

Control Table Name	Description
dmslogs.awsdms_apply_exceptions	Apply Exceptions table, always created. Contains error logs.
dmslogs.awsdms_status	Replication status of the current task
dmslogs.awsdms_suspended_tables	List of suspended tables with reason for suspension
dmslogs.awsdms_history	Replication history along with statistics

DMS Security – IAM and Encryption

- Use IAM for managing DMS access and resource permissions
- Can encrypt DMS endpoints using SSL certificates
 - Can assign a certificate to an endpoint (via DMS console or API)
 - Each endpoint may need different SSL configuration depending on the DB engine
 - Ex. Redshift already uses an SSL connection, does not require DMS SSL
 - Oracle SSL requires you to upload Oracle wallet instead of certificate (.pem) files
- Encryption at rest (for storage) uses KMS keys



IAM



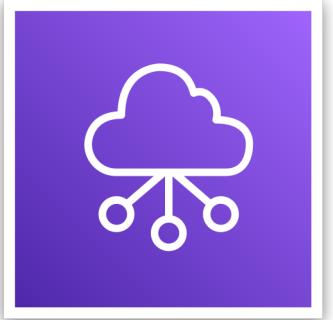
SSL



KMS

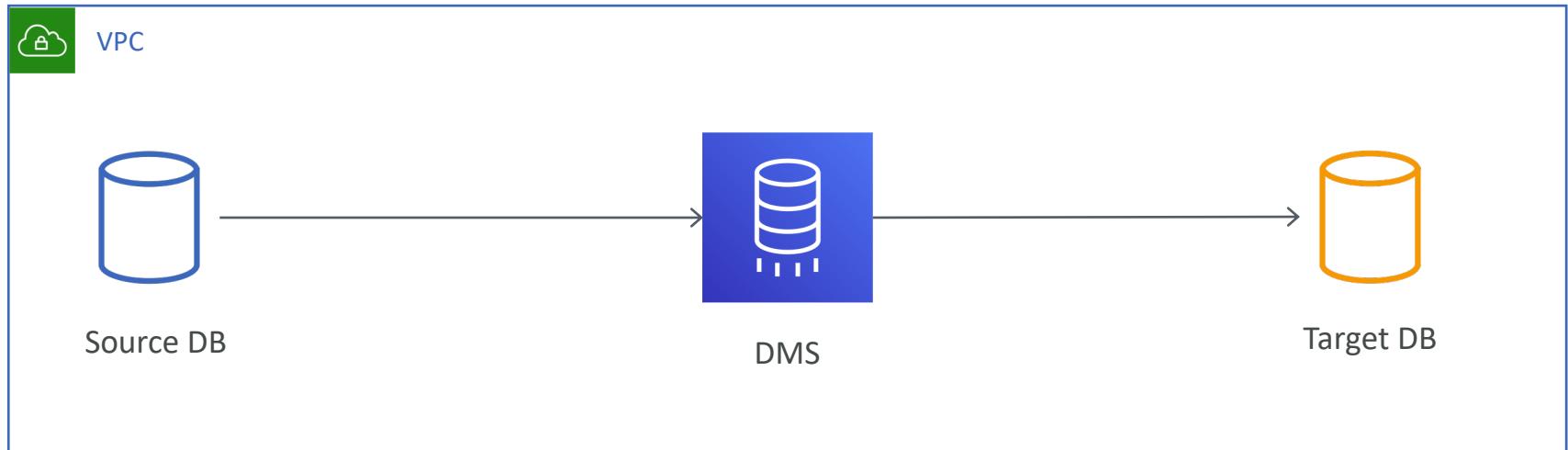
DMS Security - Networking

- DMS replication instance is always created within a VPC
- The DB endpoints must include NACLs / SG config to allow incoming access from the replication instance
- Network configurations for DMS
 - Single VPC
 - Two VPCs
 - On-premises Network to VPC (using DX / VPN / Internet)
 - RDS outside VPC (on EC2) to a DB inside VPC (via ClassicLink)



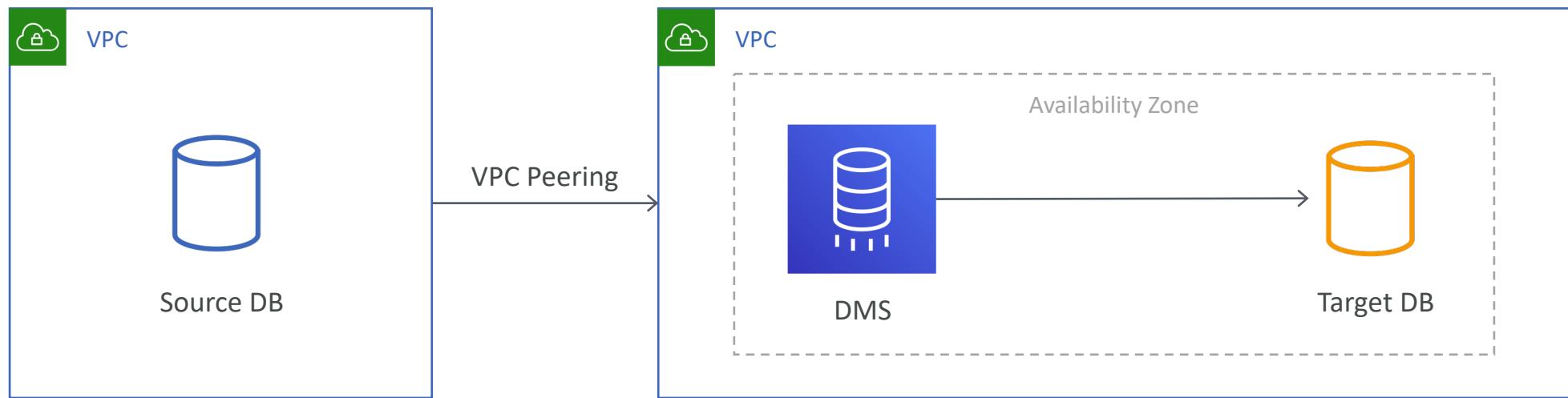
DMS Networking – Single VPC

- Simplest network configuration – all components within the same VPC



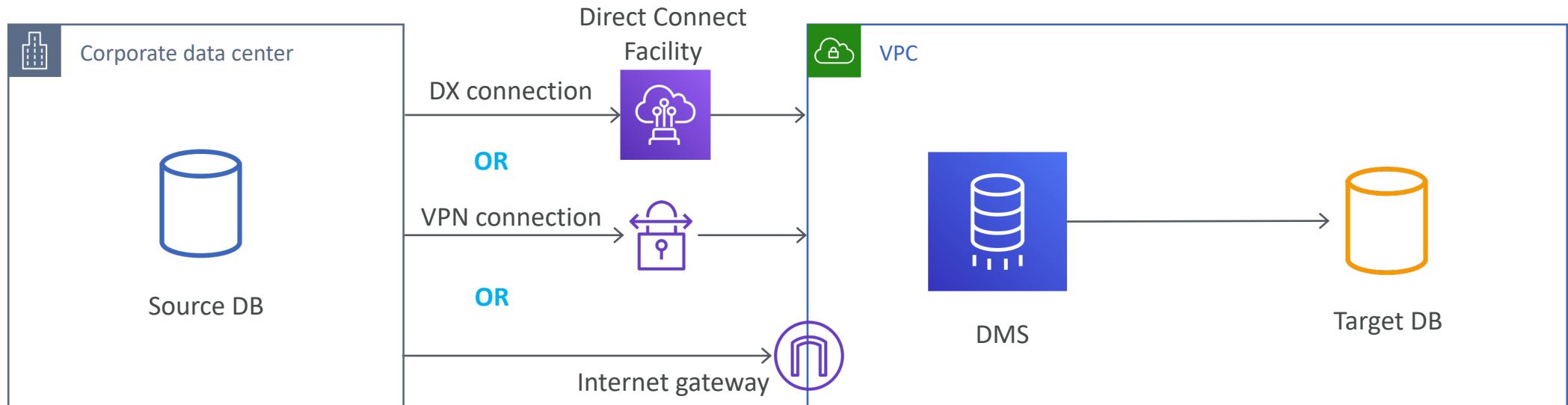
DMS Networking – Two VPCs

- Source and target endpoints in different VPCs
- Create replication instance in one of the VPCs and use VPC peering
- Generally, you'd get better performance by placing primary DMS replication instance is in the same AZ as the target DB



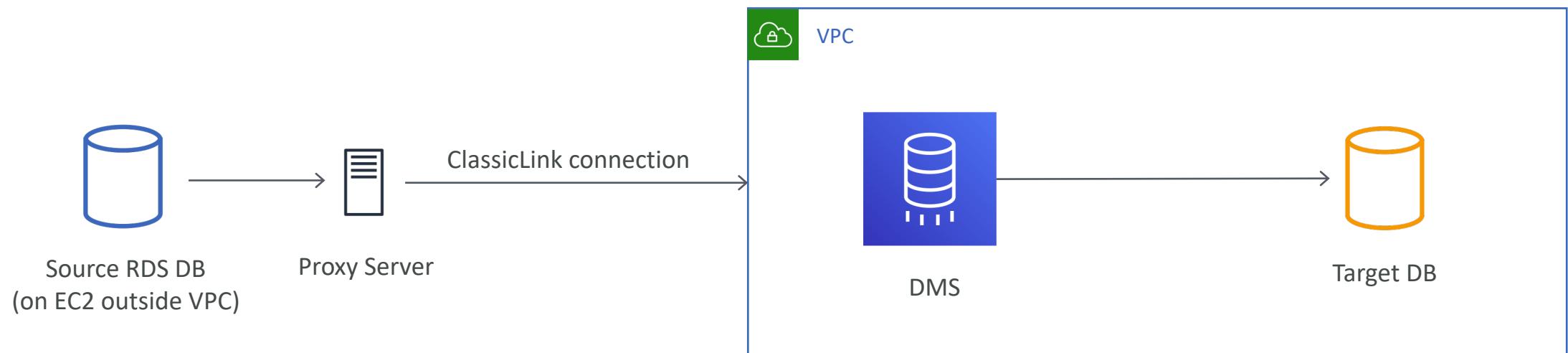
DMS Networking – On-premises to VPC

- Can use either DX or VPN
- Use Internet Gateway if DX or VPN cannot be used
- Using IG = public replication instance in a VPC



DMS Networking – RDS outside VPC to VPC

- Use ClassicLink with a proxy server
- Replication instance in the VPC cannot use ClassicLink directly (hence the need for proxy)
- Port forwarding on the proxy server allows communication between source and target



DMS Pricing

- You only pay for
 - replication instances
 - additional log storage
 - data transfer



DMS general best practices

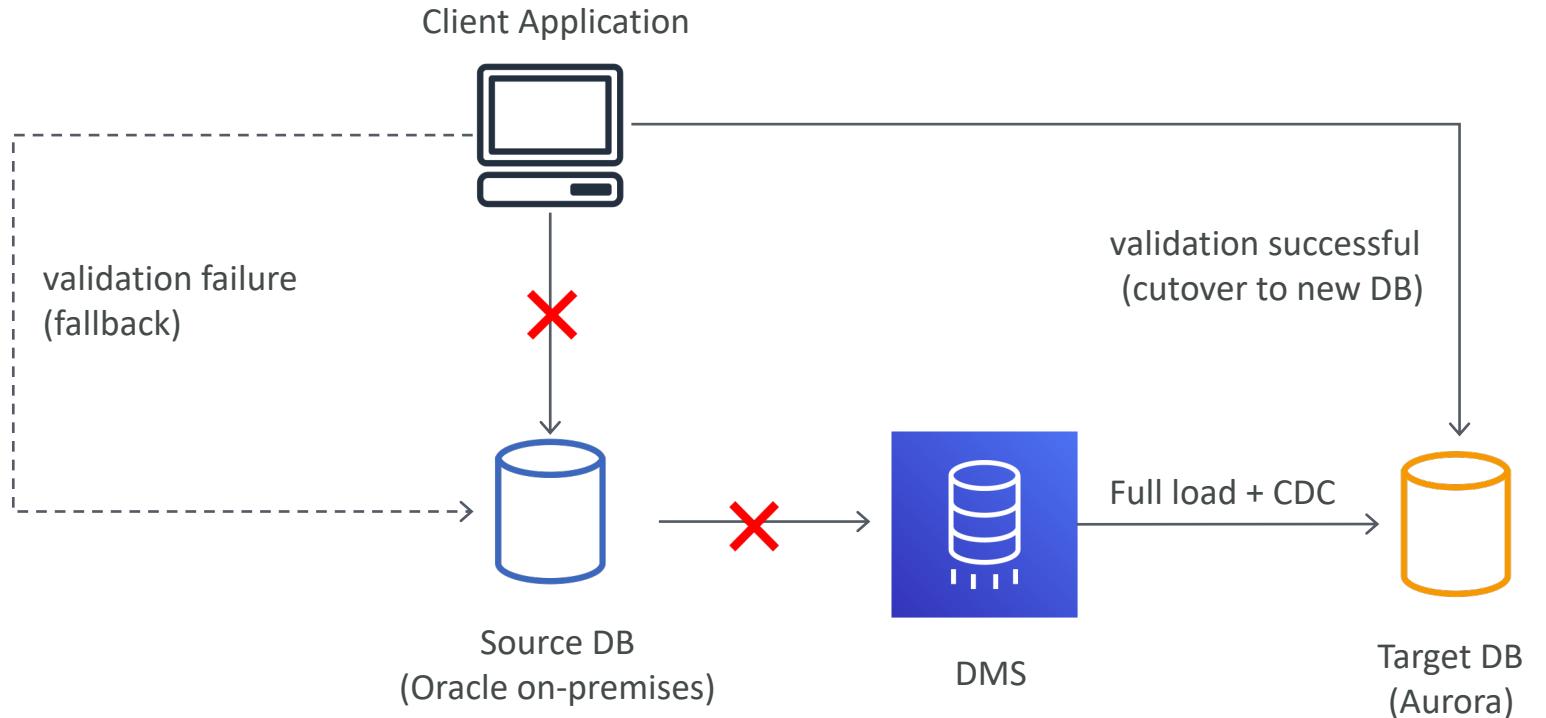
- Disable backups and transaction logs during migration
- Carry out validation during CDC (instead of during full-load)
- Use multi-AZ deployment for replication instances
- Provision appropriate instance resources
- https://docs.aws.amazon.com/dms/latest/userguide/CHAP_BestPractices.html



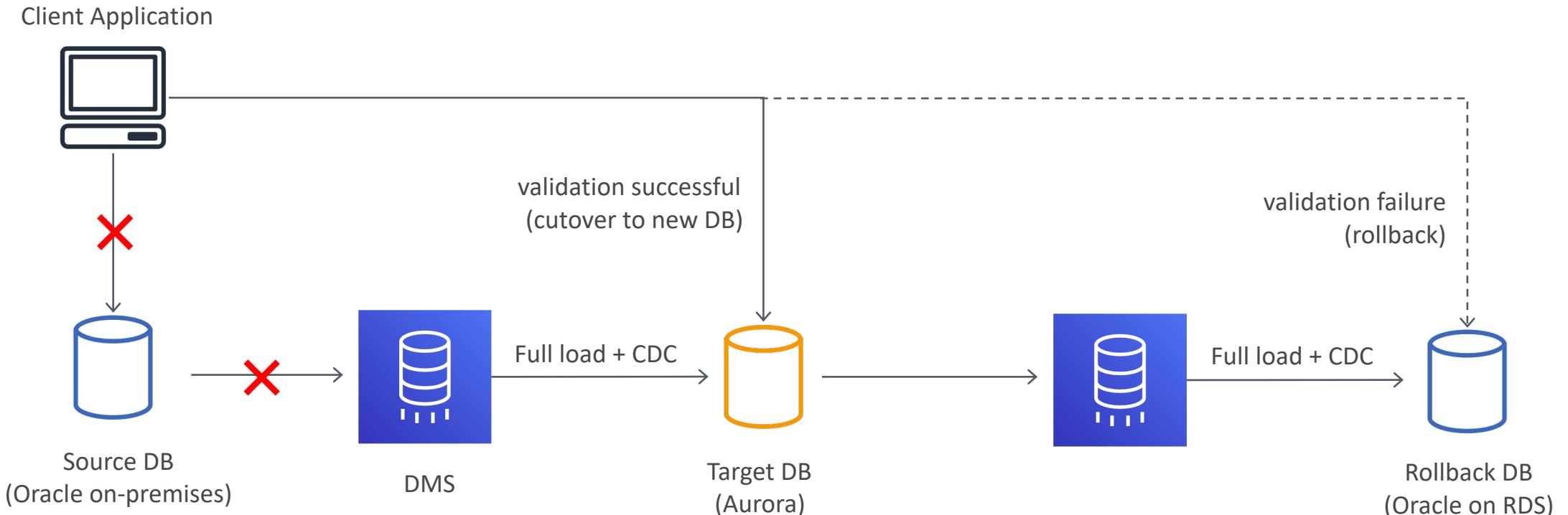
Minimizing downtime due to migration

- Solution architectures for migration with DMS
 - Fallback
 - Roll forward / Fall forward
 - Dynamic connections
 - Dual write
- Minimal downtime / near-zero downtime / zero downtime
- Zero downtime => Full load + CDC

Basic fallback approach

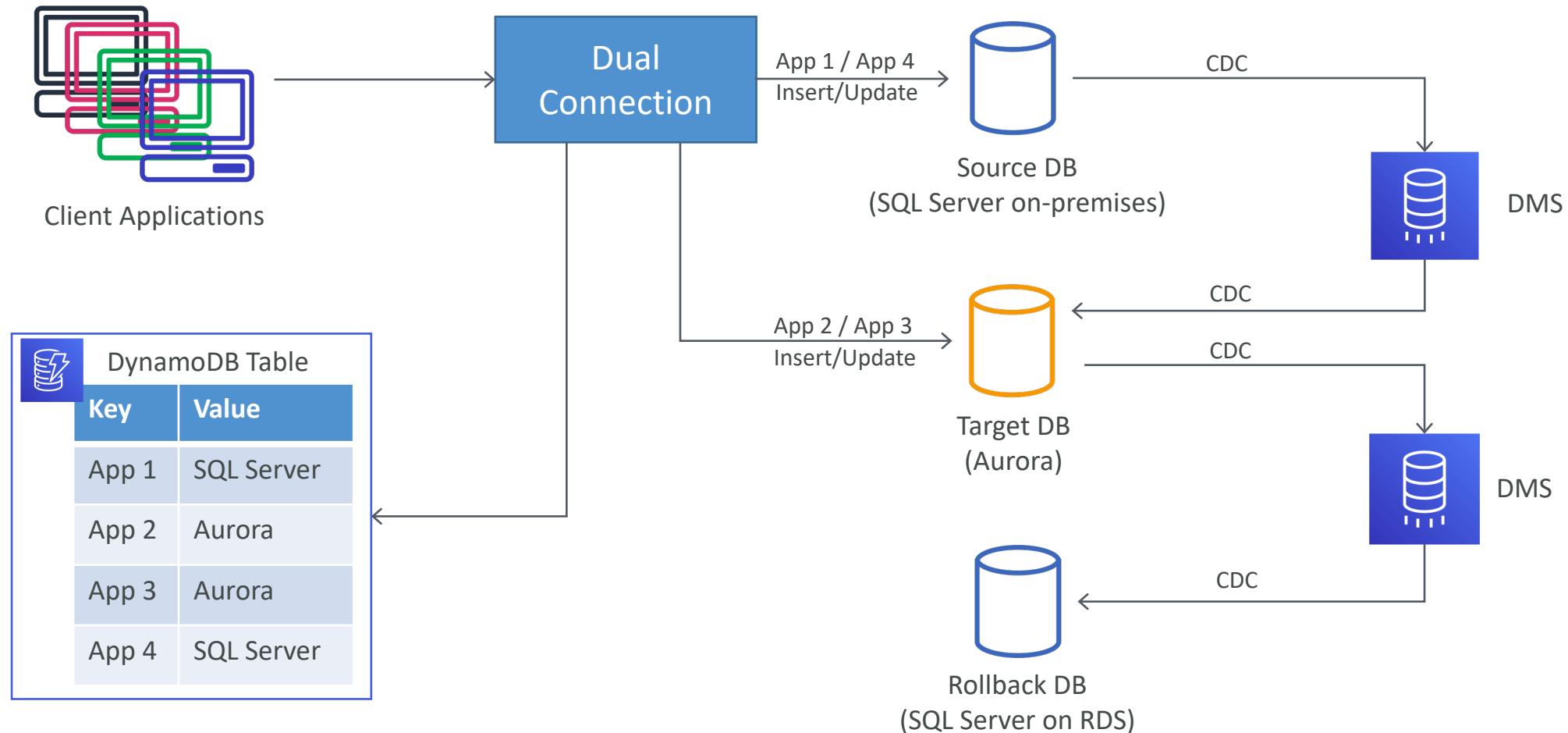


Roll forward / Fall forward approach

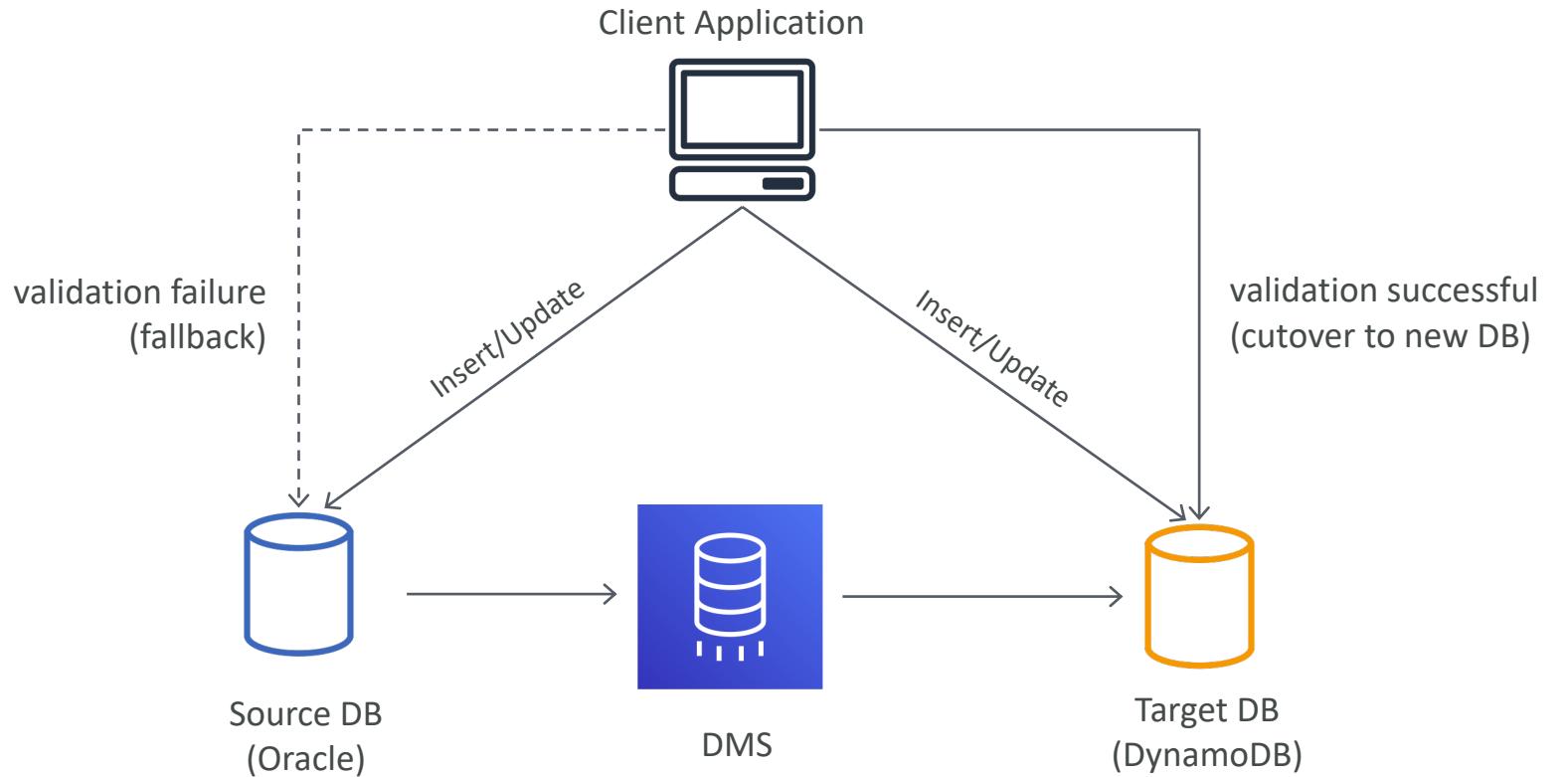


Roll-forward with Dynamic Connection

- Applications write to both old and new databases. Each application can cutover separately.



Dual write approach

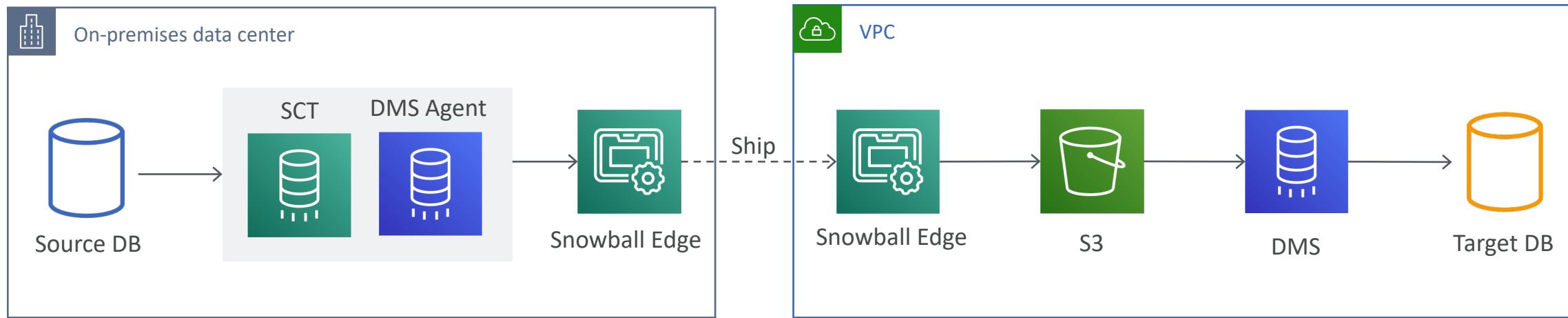


More migration architectures

- <https://aws.amazon.com/blogs/database/rolling-back-from-a-migration-with-aws-dms/>

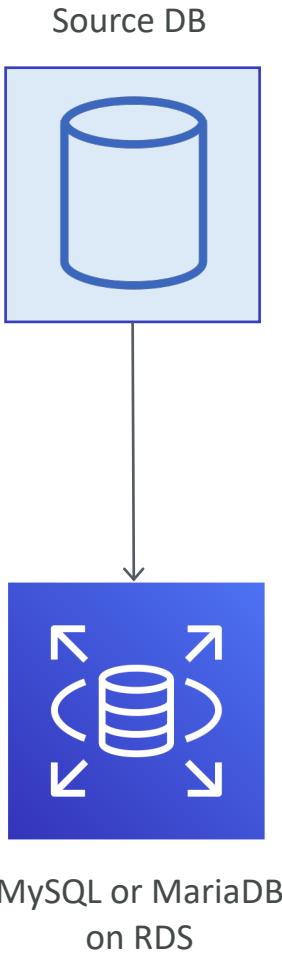
Migrating large databases

- Use multiphase migration
- Copy static tables first (before migrating active tables)
- Cleanup old unwanted data to reduce DB size
- Alternatively, use Snowball Edge to move data to S3 and then migrate using DMS



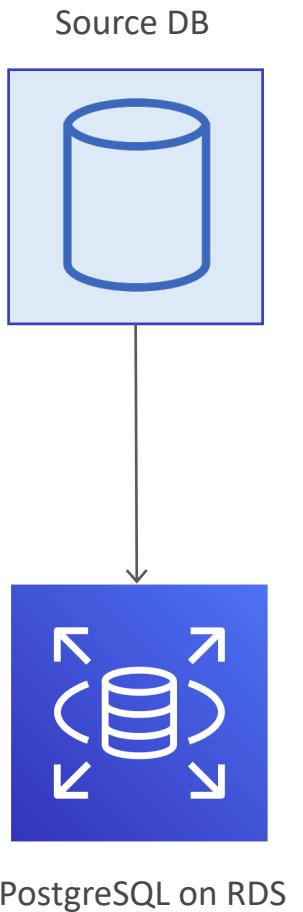
Migrating to MySQL / MariaDB on RDS

- From MySQL / MariaDB (on-premises / S3 / EC2)
 - Small to medium DBs – use mysqldump / mysqlimport utilities (some downtime)
 - One time – Restore from backup (or data dump) stored on S3 (some downtime)
 - Ongoing – Configure binlog replication from existing source (minimal downtime)
- From MySQL / MariaDB on RDS
 - One time / Ongoing – Promote a read replica to be a standalone instance
- From any DB
 - One time / Ongoing – Use DMS (minimal downtime)



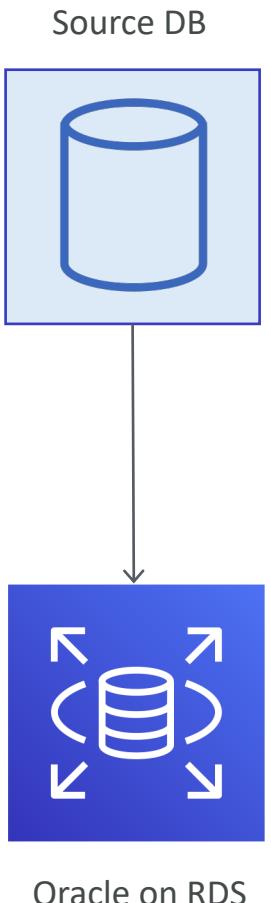
Migrating to PostgreSQL on RDS

- From PostgreSQL (on-premises / EC2)
 - One time – use `pg_dump` / `pg_restore` (some downtime)
- From CSV data stored on S3
 - Use `aws_s3` PostgreSQL extension and import data using the `aws_s3.table_import_from_s3` function (some downtime)
- From PostgreSQL on RDS (large DBs)
 - Use `pg_transport` extension (streams data, is extremely fast, minimal downtime)
- From any DB
 - One time / Ongoing – Use DMS (minimal downtime)



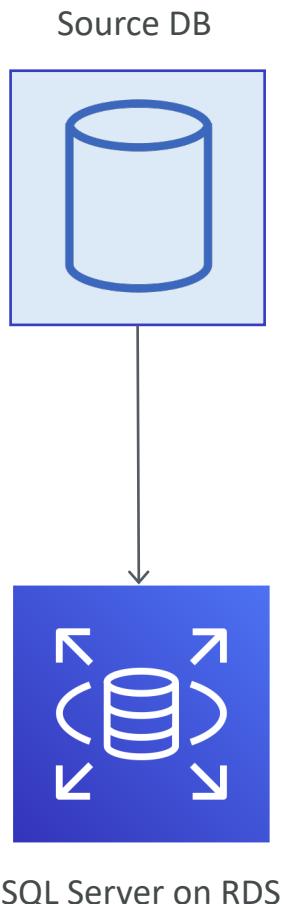
Migrating to Oracle on RDS

- For small DBs – use Oracle SQL Developer tool (freeware)
 - Perform Database Copy with Oracle SQL Developer
 - Supports Oracle and MySQL as source
- For large DBs – use **Oracle Data Pump**
 - Can export and import between Oracle DBs (on-prem / EC2 / RDS)
 - Can use S3 to transfer the dump file (use option group with option S3_INTEGRATION)
 - Can also transfer using creating a database link between source and target
- From any DB
 - One time / Ongoing – Use DMS (minimal downtime)



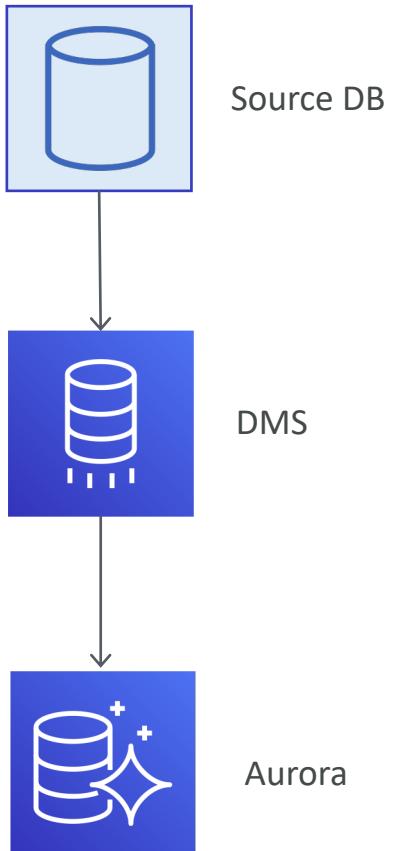
Migrating to SQL Server on RDS

- From SQL Server (on-premises / EC2)
 - Use native backup and restore (.bak backup files stored on S3)
 - Supports encryption and compression
 - Or use SQL Server Management Studio
- Microsoft SQL Server Management Studio (freeware, has three options)
 - Generate and Publish Scripts wizard – creates a script with schema and/or data
 - Import and Export wizard
 - Bulk copy
- From SQL Server (on RDS)
 - Restore from a snapshot
 - Or use native backup and restore feature
 - Or use SQL Server Management Studio
- From any DB
 - One time / Ongoing – Use DMS (minimal downtime)



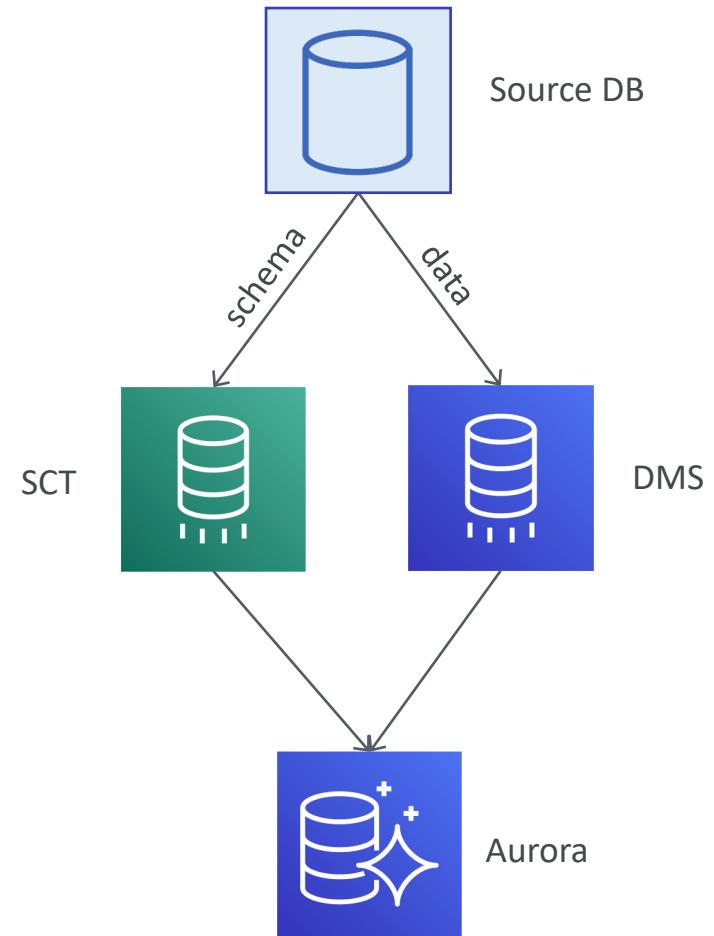
Homogenous Migration to Aurora

- MySQL 5.6 compliant DBs (MySQL / MariaDB / Percona)
- Homogenous Migration with Downtime
 - Restore from RDS snapshot
 - Full load using native DB tools
 - Full load using DMS (migrate schema with native tools first)
- Homogenous Migration with Near-Zero Downtime (minimal downtime)
 - Restore from RDS snapshot + MySQL binlog replication
 - Full load using native tools + MySQL binlog replication
 - Full load + CDC using DMS
 - Create an Aurora replica from MySQL / PostgreSQL on RDS and promote it to a standalone DB



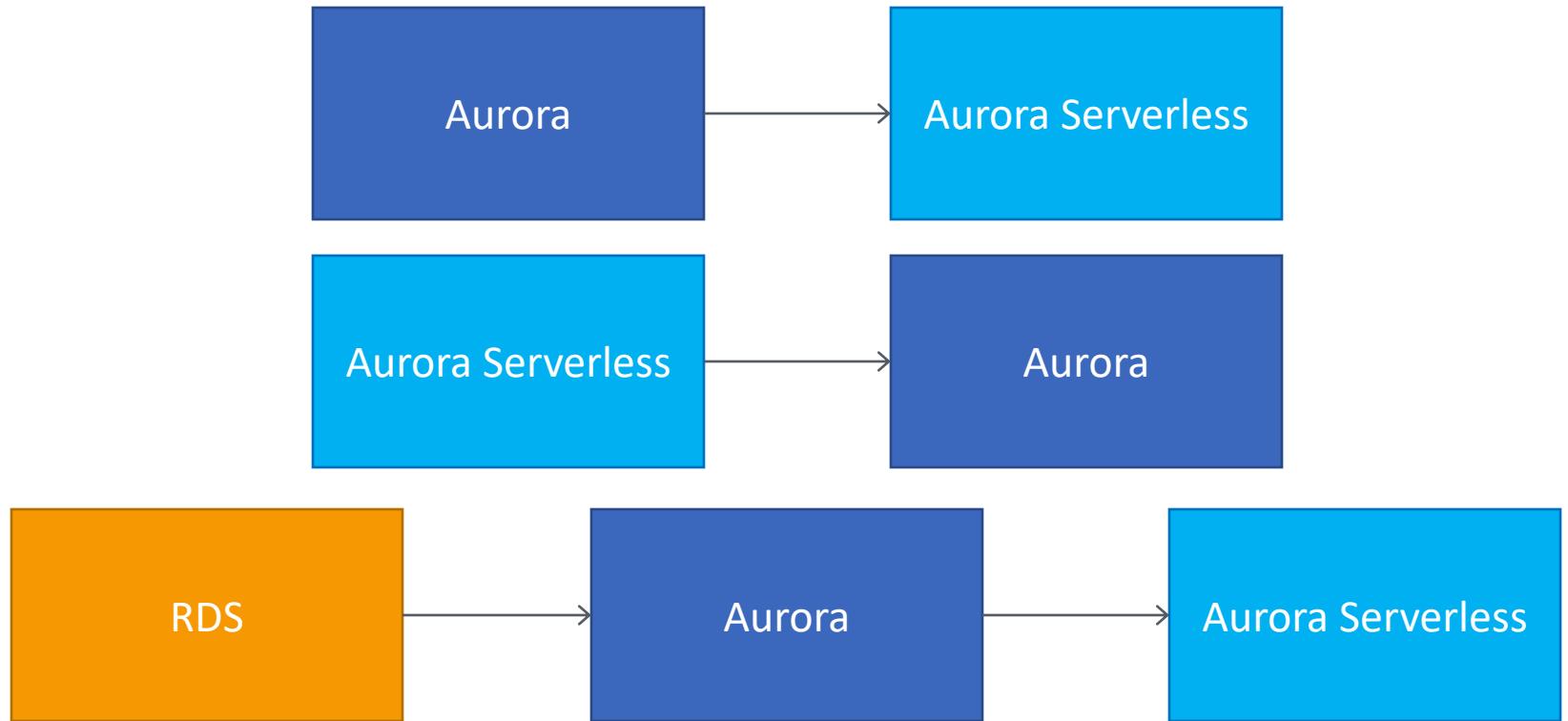
Heterogeneous Migration to Aurora

- Schema migration – use SCT
- Data Migration – use DMS
- For near-zero downtime – use continuous replication using DMS (Full load + CDC)



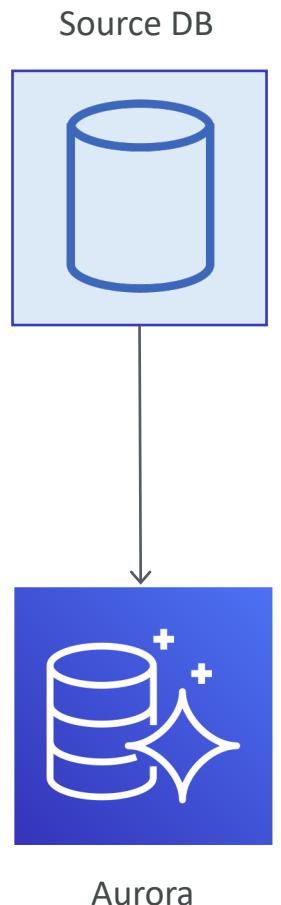
Migrating to Aurora Serverless

- Can migrate between Aurora provisioned cluster and Aurora Serverless cluster
- Can migrate from RDS to Aurora to Aurora Serverless (not directly)



Strategies for Migration to Aurora

- MySQL / PostgreSQL on RDS to Aurora (Homogeneous)
 - Restore from a snapshot to Aurora (manual or automated snapshot)
 - Replicate to an Aurora read replica and then promote the replica to a standalone DB
 - Use DMS
- MySQL on EC2 or MySQL on-premise to Aurora (Homogeneous)
 - Restore from backup files stored on S3
 - Restore from text files (CSV / XML) stored in S3
 - Restore using mysqldump utility
 - Use DMS
- PostgreSQL on EC2 or PostgreSQL on-premise to Aurora (Homogeneous)
 - Migrate to PostgreSQL to RDS and then migrate to Aurora
 - Use DMS
- MariaDB / Oracle / SQL Server to Aurora (Heterogeneous)
 - Use DMS and SCT

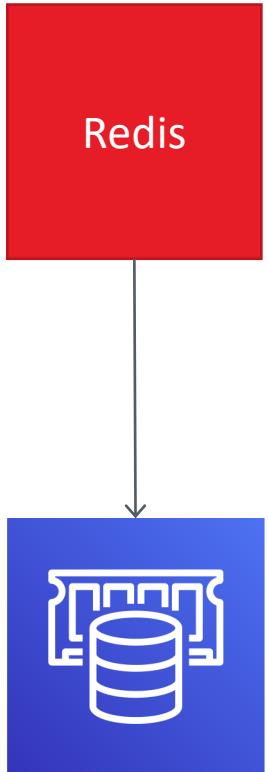


Additional reading

- White paper - Migrating Your Databases to Amazon Aurora
 - <https://dl.awsstatic.com/whitepapers/RDS/Migrating%20your%20databases%20to%20Amazon%20Aurora.pdf>

Migrating Redis workloads to ElastiCache

- Two approaches
 - Offline migration (using backup)
 - Online migration (migrate data from endpoint)



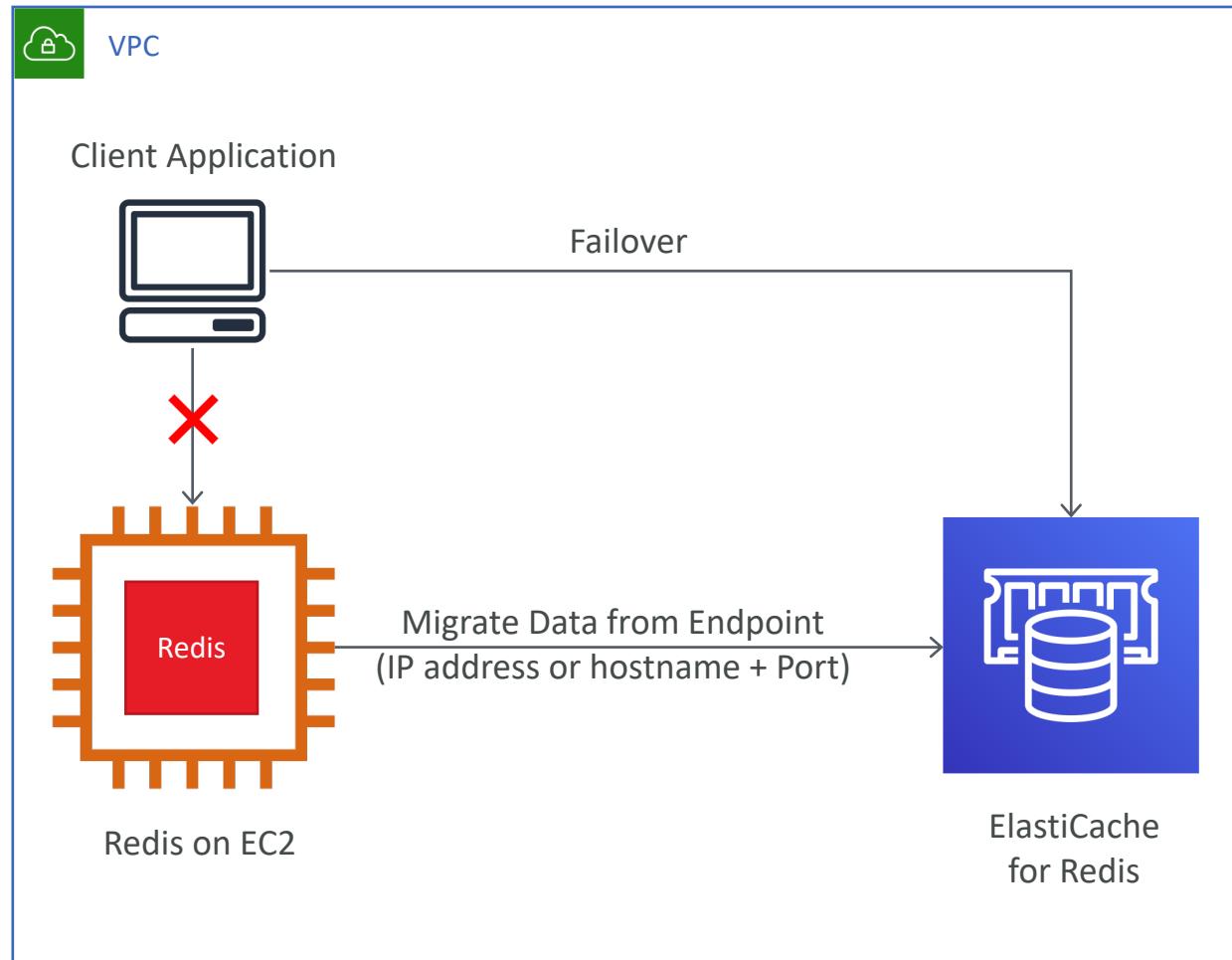
Offline migration to ElastiCache for Redis

- Create a Redis backup (.RDB file)
- Create an S3 bucket
- Upload the backup to S3
- Create the target ElastiCache cluster and choose the option to seed the RDB file from S3 location
- ElastiCache should have read access to the RDB file



Online migration to ElastiCache for Redis

- Real-time data migration
- Migrate self-hosted Redis on EC2 to ElastiCache
- Create the target Redis cluster (or choose an existing one)
- Under Actions, choose Migrate Data from Endpoint
- Monitor using the Events section on the ElastiCache console
- Failover to the new DB
- You can decide when to failover



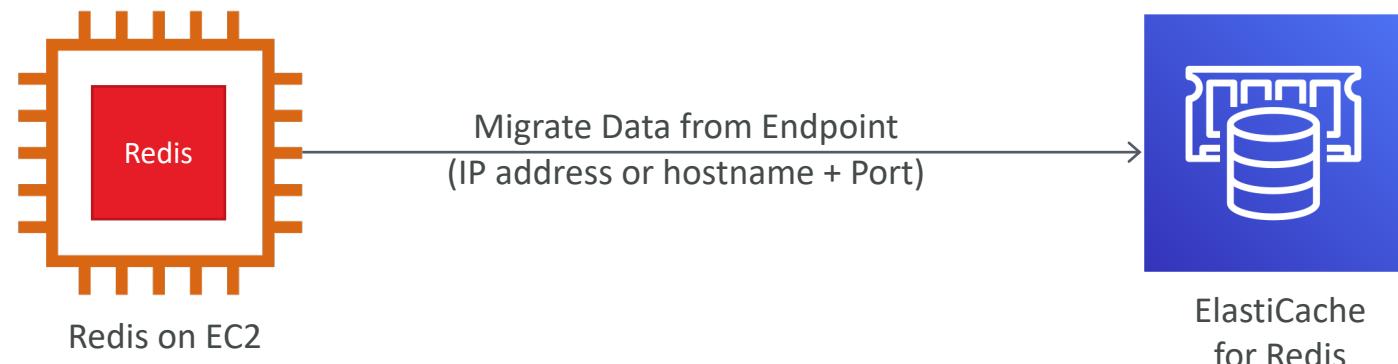
Good to know – Online Redis migration

- Source cluster

- must have Redis AUTH disabled
- must have “protected-mode” disabled
- “bind” config if present should allow requests from ElastiCache

- Target cluster

- must have Cluster-mode disabled
- must have Multi-AZ enabled
- must have Encryption disabled
- must have sufficient memory



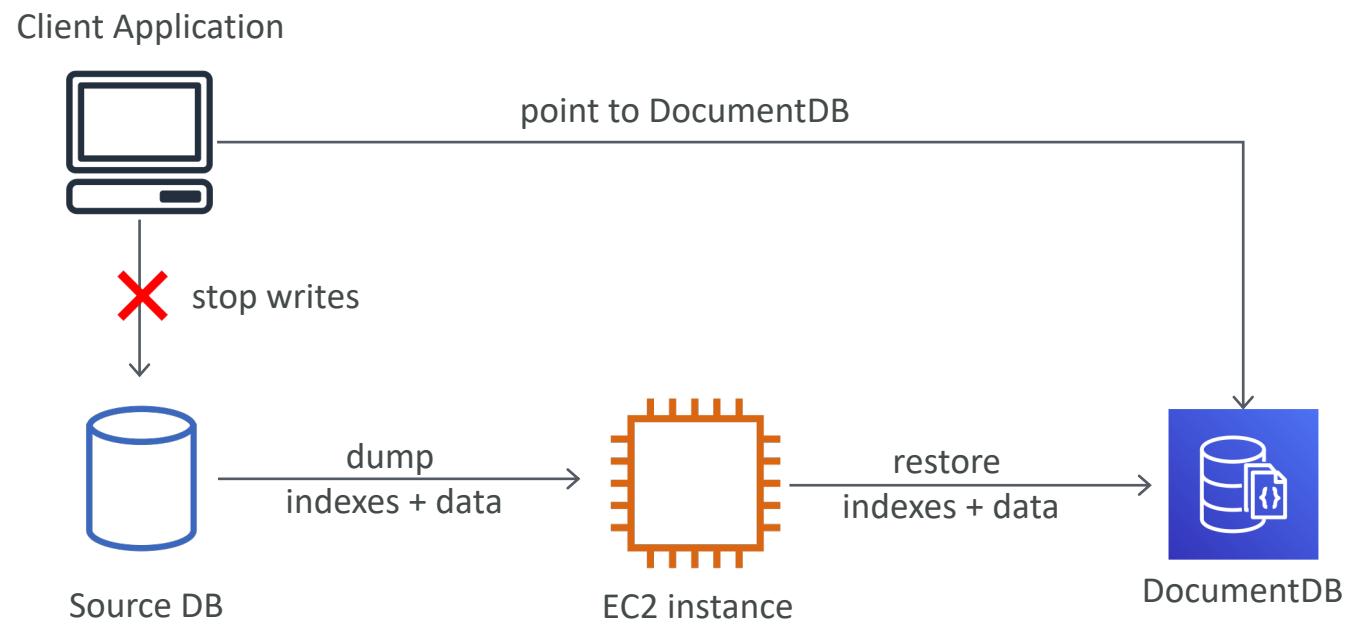
Migrating to DocumentDB

- Four approaches
 - Offline migration
 - Online migration
 - Hybrid Approach
 - Dual write approach



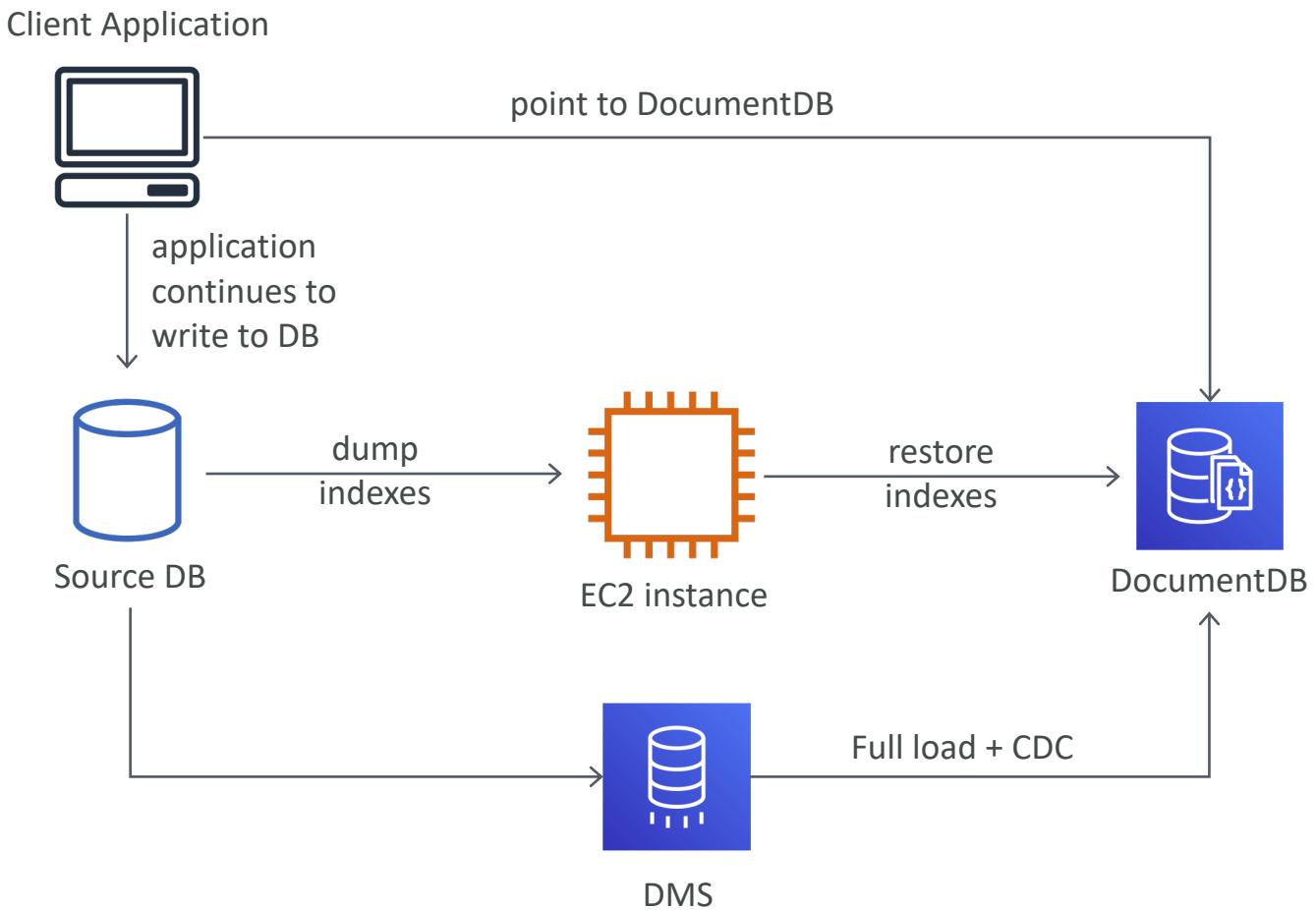
Offline migration to DocumentDB

- Simplest and fastest, but with the longest downtime
- Non-critical / non-production workloads
- Homogeneous – mongodump / mongorestore (BSON)
- Homo / Heterogeneous – mongoexport / mongoimport (JSON / CSV)
- Use DocumentDB index tool to export/import indexes



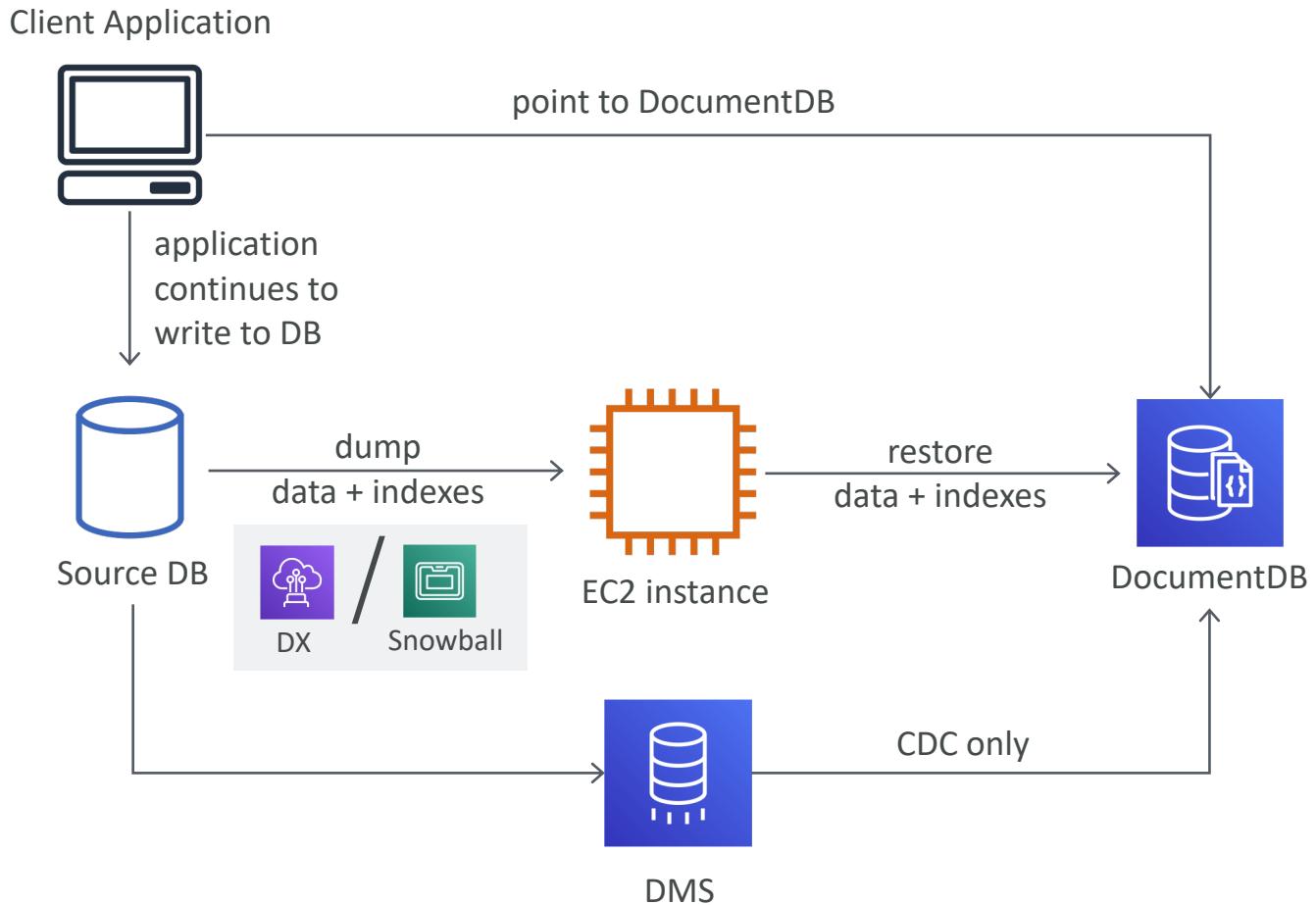
Online migration to DocumentDB

- Medium complexity, slowest, and with minimal downtime
- For production workloads
- Uses DMS
- Migrate indexes using DocumentDB index tool
- DMS does not migrate indexes



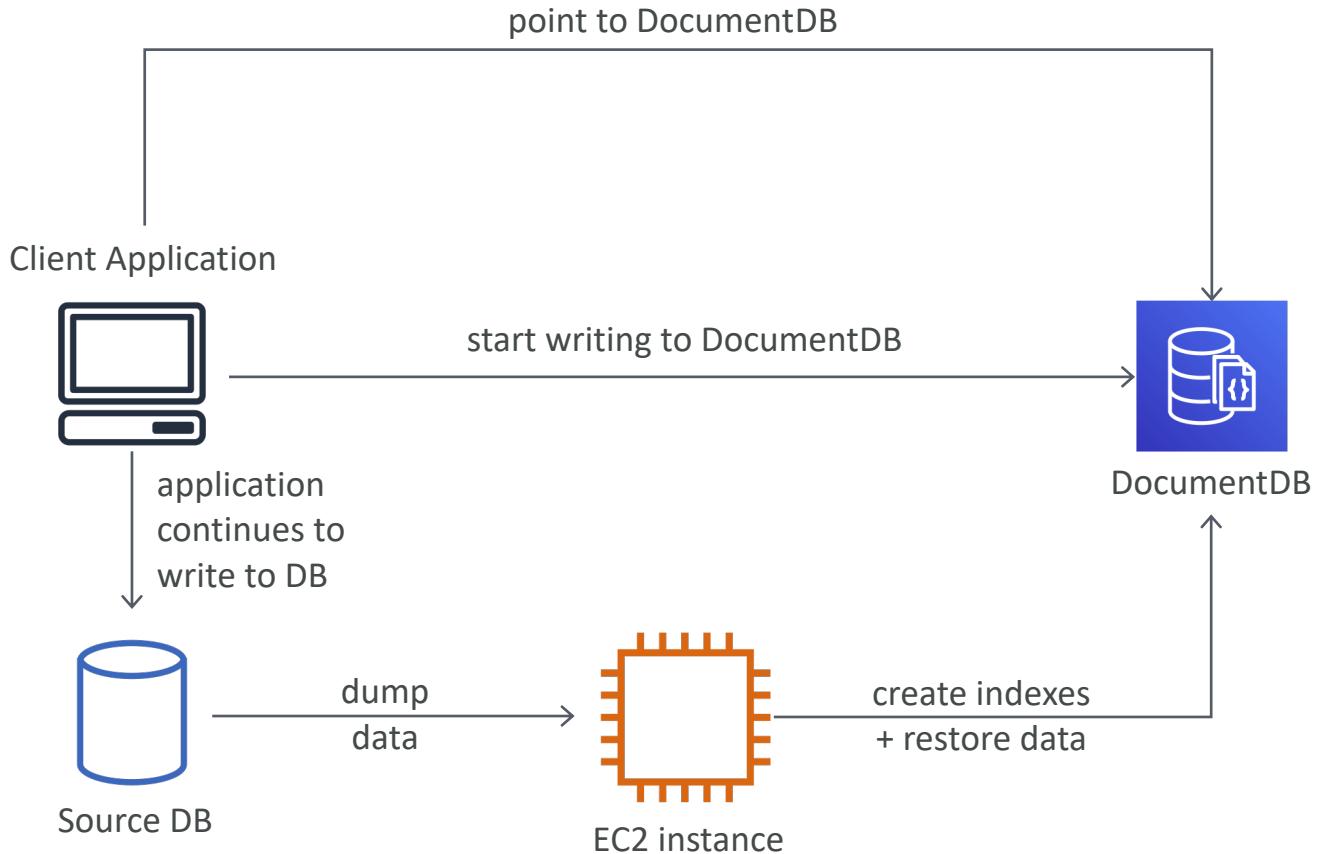
Hybrid Approach to DocumentDB Migration

- Mix of offline and online migration
- Most complex, faster than online, and with minimal downtime
- For production workloads
 - when DB size is large
 - or if you don't have enough network bandwidth
- Phase 1
 - Export with mongodump
 - Transfer it to AWS if on-premise (Direct Connect / Snowball)
 - Migrate indexes using DocumentDB index tool
 - Restore to DocumentDB
- Phase 2
 - Use DMS in CDC mode



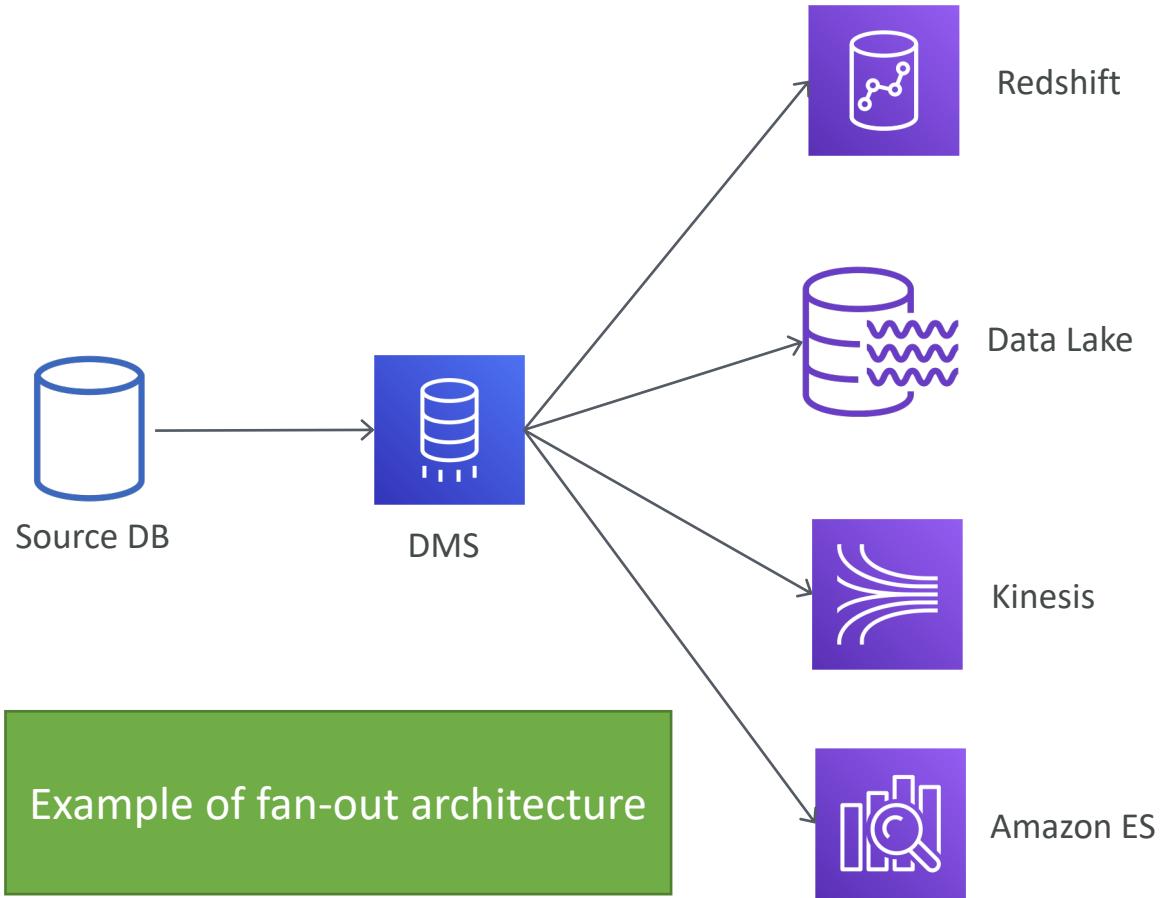
Dual write approach for DocumentDB Migration

- Typically used for heterogeneous migrations
- Example – RDBMS to DocumentDB
- Create indexes manually (in case of heterogeneous migration)

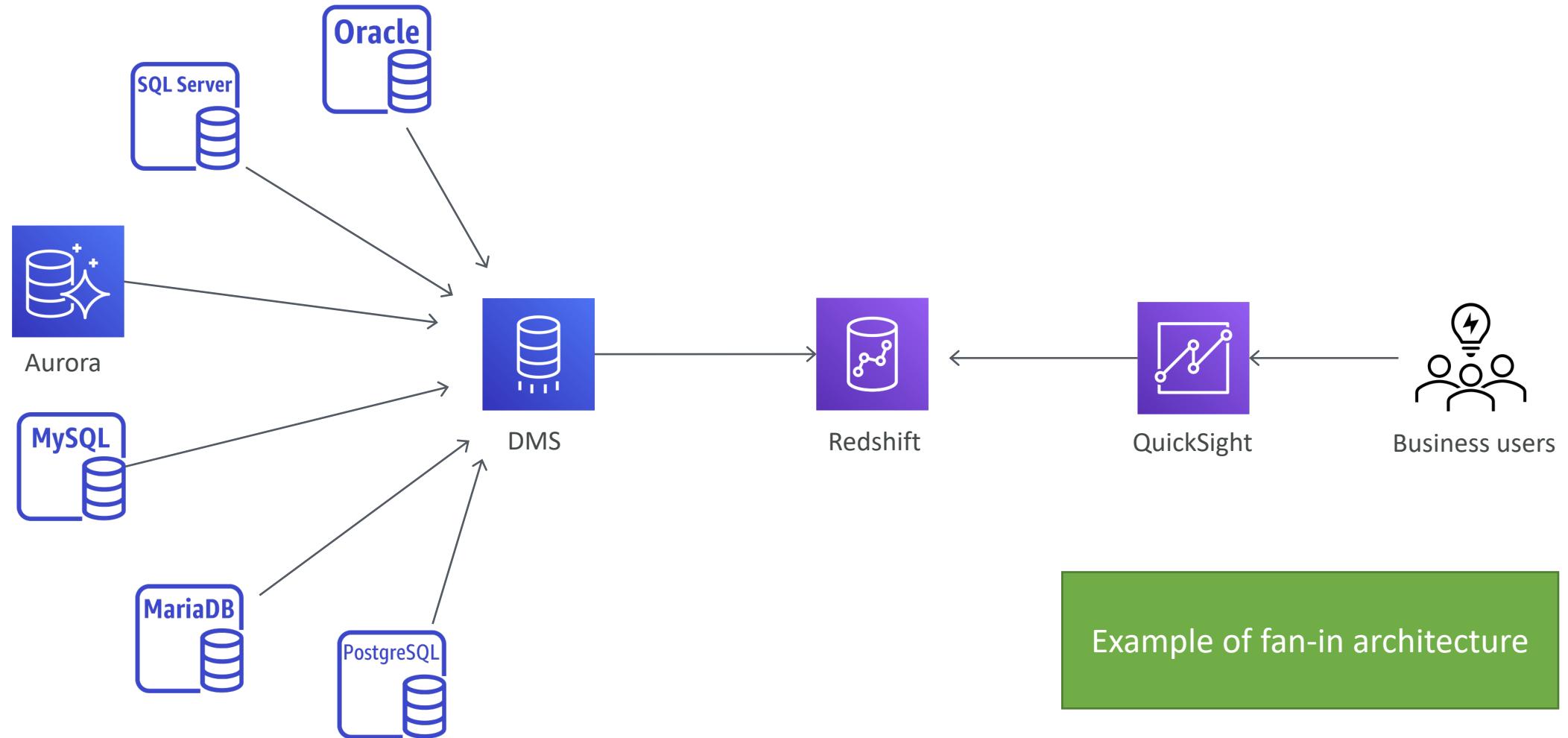


Streaming use cases for DMS

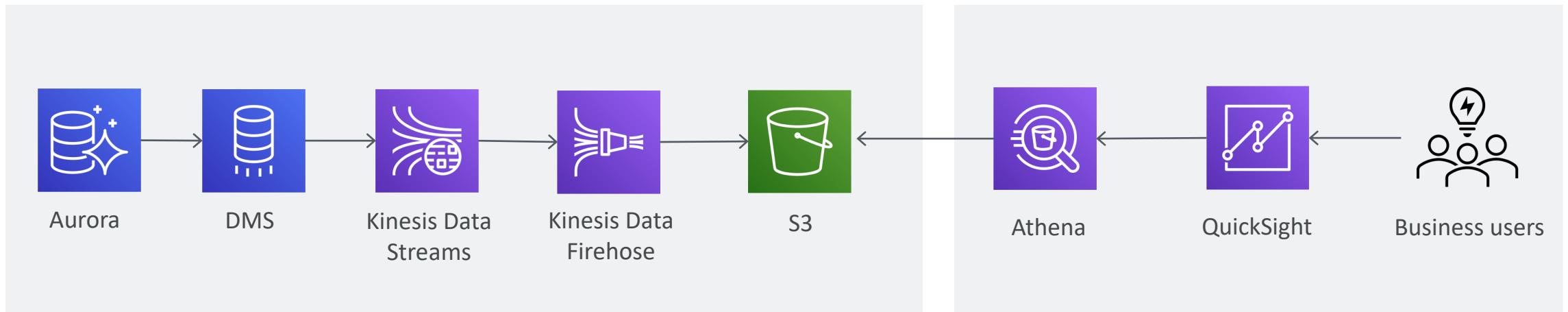
- Can stream data from source to target (=ongoing replication)
 - Redshift - Stream from OLTP to redshift for analytics
 - S3 data lakes – hydrate data lakes
 - Stream to Kinesis data streams
 - Stream to ElasticSearch Service
- Can use fan-out architecture
- Can also use fan-in architecture



Streaming use cases for DMS



Streaming use cases for DMS



Streaming + Replication + Analytics

Monitoring, Encryption, Security and Auditing

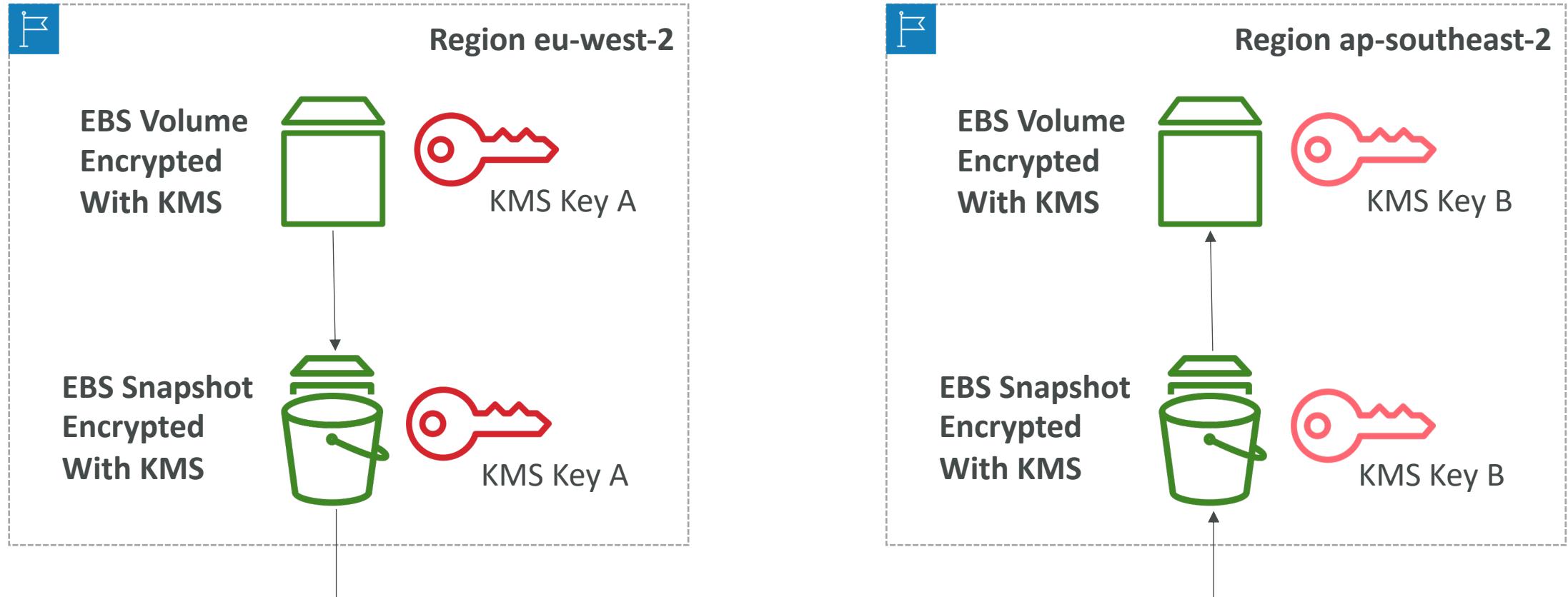
It's pays to be aware and in the knowing!

AWS KMS (Key Management Service)



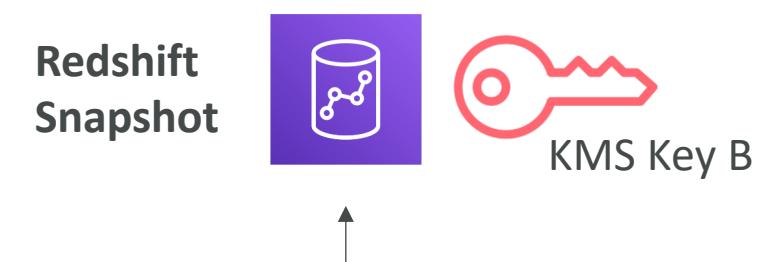
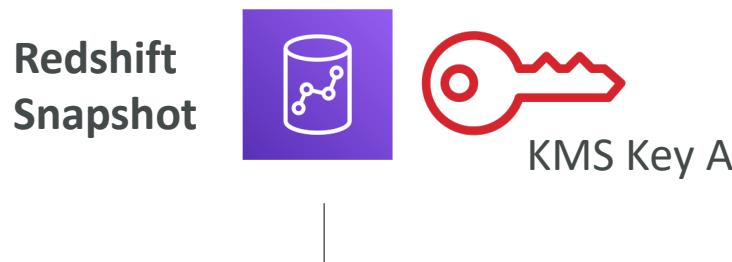
- Anytime you hear “encryption” for an AWS service, it’s most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
 - Amazon EBS: encrypt volumes
 - Amazon S3: Server side encryption of objects
 - Amazon Redshift: encryption of data
 - Amazon RDS: encryption of data
 - Amazon SSM: Parameter store
 - Etc...
- But you can also use the CLI / SDK

Copying Snapshots across regions



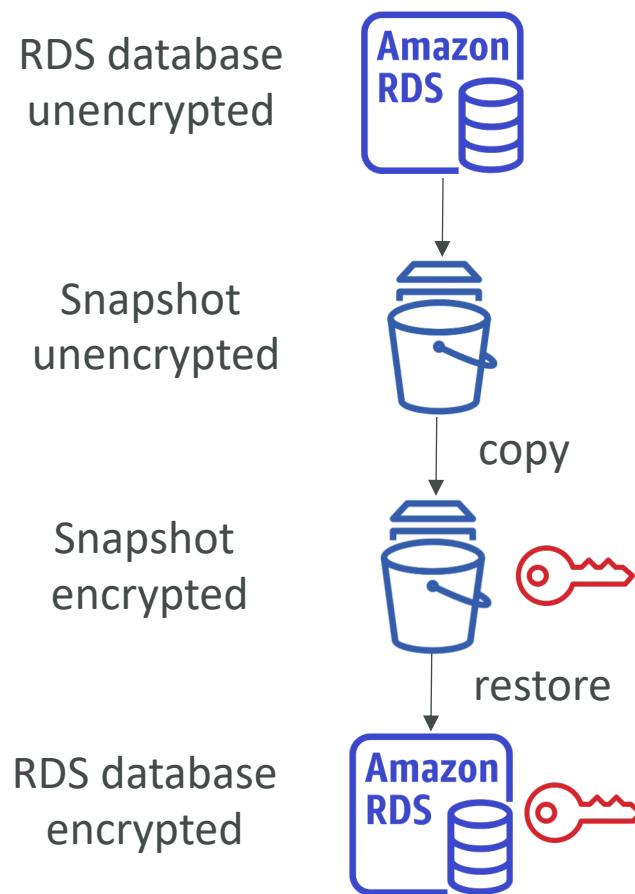
Cross Region Snapshots for Redshift Snapshot Copy Grant

- <https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-snapshots.html#cross-region-snapshot-copy>
- In the destination AWS Region, create a snapshot copy grant by doing the following:
 - Create a KMS key in the destination region
 - Specify a name for the snapshot copy grant. This name must be unique in that AWS Region for your AWS account.
 - Specify the AWS KMS key ID for which you are creating the grant
- In the source AWS Region, enable copying of snapshots and specify the name of the snapshot copy grant that you created in the destination AWS Region.



Snapshot thanks to key grant in destination region

Encrypting an un-encrypted RDS database



1. Take a snapshot of the RDS database
 - The snapshot will be un-encrypted
2. Create an encrypted copy of the snapshot
 - Using the KMS key of your choosing
3. Restore a database from the encrypted snapshot
 - The database will be encrypted !



Database Logging – RDS

- Engine log files available (Oracle, MSSQL, PostgreSQL, MySQL, MariaDB)
 - List log files: `aws rds describe-db-log-files`
 - Download log files: `aws rds download-db-log-file-portion`
- Normal log retention in RDS up to 7 days (configurable per DB)
- Logs can be published into **CloudWatch Logs**
 - you can perform real-time analysis of the log data
 - store the data in highly durable storage (retention period configurable / infinite)
 - From CloudWatch Logs you can export to S3
 - Must create a custom parameter group (can't modify default)



Database Logging – Aurora

- Engine log files available (PostgreSQL, MySQL)
 - List log files: `aws rds describe-db-log-files`
 - Download log files: `aws rds download-db-log-file-portion`
- Normal log retention in RDS up to 7 days (configurable per DB)
- Logs can be published into **CloudWatch Logs**
 - you can perform real-time analysis of the log data
 - store the data in highly durable storage (retention period configurable / infinite)
 - From CloudWatch Logs you can export to S3
 - Not available for transaction logs



Database Logging – Redshift

- Amazon Redshift logs information about connections and user activities in your database (for troubleshooting + audit)
 - *Connection log* — logs authentication attempts, and connections and disconnections.
 - *User log* — logs information about changes to database user definitions.
 - *User activity log* — logs each query before it is run on the database.
- The logs are stored in Amazon S3 buckets (must be enabled)
 - Set lifecycle policies accordingly
 - Ensure S3 bucket policies allows for Redshift to write to the bucket

Database Logging – DynamoDB



- All API calls to DynamoDB are logged into CloudTrail
- From CloudTrail you can send to :
 - CloudWatch Logs
 - Amazon S3 buckets
- There are no "log files" in DynamoDB, it's a proprietary technology



Database Logging – DocumentDB

- You can audit DocumentDB Events (must opt in)
- Examples of logged events
 - successful and failed authentication attempts
 - dropping a collection in a database
 - creating an index
 - Data Definition Language (DDL)
- Logs are sent into CloudWatch Logs
- To opt in, set the `audit_logs` parameter to `enabled` (parameter group)

Database Logging - Other

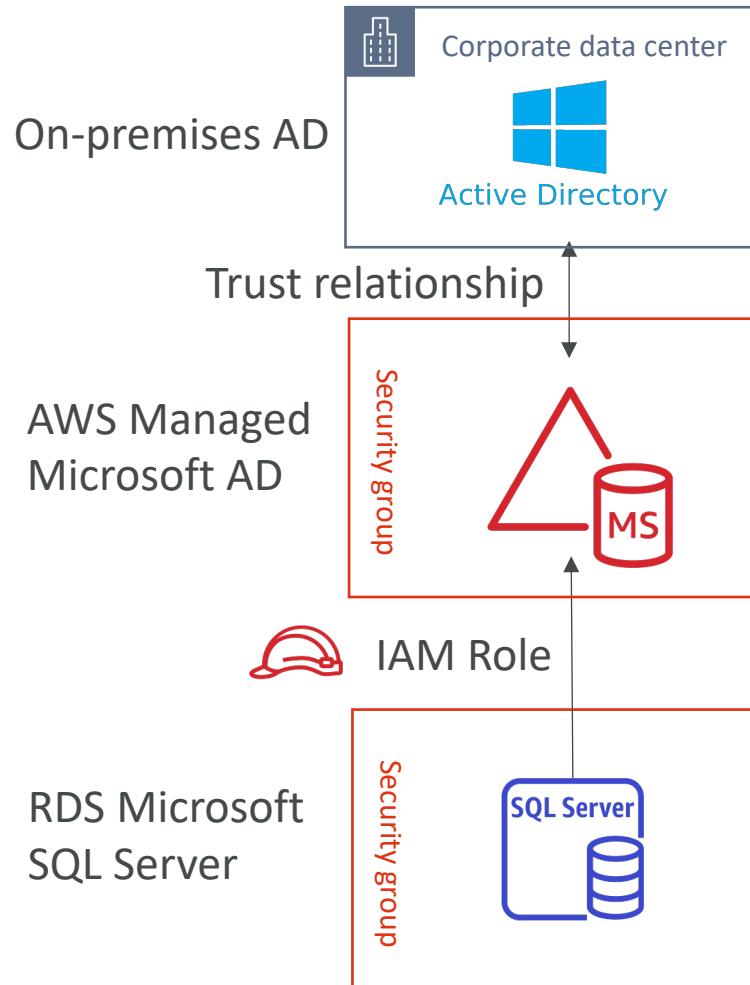
- ElastiCache: no logs to access yet
- Neptune:
 - publish audit log data to a log group in Amazon CloudWatch Logs
 - Use the `neptune_enable_audit_log` parameter to enable (1) or disable (0)
- QLDB:
 - No logs are accessible
- DMS:
 - Can set task-logging level to `LOGGER_SEVERITY_DETAILED_DEBUG` (most detailed log)

AWS Secrets Manager



- Newer service, meant for storing secrets
- Capability to force **rotation of secrets** every X days
- Uses a Lambda function
 - Generated by AWS for integrated DB types (Amazon RDS, Redshift, DocumentDB)
 - For other secrets, you need to code a Lambda function to generate the next secret
- Secrets are encrypted using KMS
- Mostly meant for RDS integration

Active Directory with RDS SQL Server



- Use AWS Managed Microsoft AD
 - If you need to join your on-premises AD, create a trust relationship
- Create an IAM role with `AmazonRDSDirectoryServiceAccess` to access AWS Managed Microsoft AD
- Create and configure users and groups in the AWS Managed Microsoft AD directory
- Create a new or modify an existing Amazon RDS instance with
 - A reference to the IAM role that has access to the AD
 - A reference to the AWS Managed Microsoft AD
 - No need to stop the DB if you are modifying it
- Ensure security groups allow communication between RDS and Microsoft AD
- Log in to the DB using the Master User Credentials and create logins

CloudFormation and Automation

Few lines of code is all you need to create or terminate your databases!

Infrastructure as Code

- Currently, we have been doing a lot of manual work
- All this manual work will be very tough to reproduce:
 - In another region
 - in another AWS account
 - Within the same region if everything was deleted
- Wouldn't it be great, if all our infrastructure was... code?
- That code would be deployed and create / update / delete our infrastructure



What is CloudFormation

- CloudFormation is a declarative way of outlining your AWS Infrastructure, for any resources (most of them are supported).
- For example, within a CloudFormation template, you say:
 - I want a security group
 - I want two EC2 machines using this security group
 - I want two Elastic IPs for these EC2 machines
 - I want an S3 bucket
 - I want a load balancer (ELB) in front of these machines
- Then CloudFormation creates those for you, in the **right order**, with the **exact configuration** that you specify

Benefits of AWS CloudFormation (1/2)

- Infrastructure as code
 - No resources are manually created, which is excellent for control
 - The code can be version controlled for example using git
 - Changes to the infrastructure are reviewed through code
- Cost
 - Each resources within the stack is tagged with an identifier so you can easily see how much a stack costs you
 - You can estimate the costs of your resources using the CloudFormation template
 - Savings strategy: In Dev, you could automation deletion of templates at 5 PM and recreated at 8 AM, safely

Benefits of AWS CloudFormation (2/2)

- Productivity
 - Ability to destroy and re-create an infrastructure on the cloud on the fly
 - Automated generation of Diagram for your templates!
 - Declarative programming (no need to figure out ordering and orchestration)
- Separation of concern: create many stacks for many apps, and many layers. Ex:
 - VPC stacks
 - Network stacks
 - App stacks
- Don't re-invent the wheel
 - Leverage existing templates on the web!
 - Leverage the documentation

How CloudFormation Works

- Templates have to be uploaded in S3 and then referenced in CloudFormation
- To update a template, we can't edit previous ones. We have to re-upload a new version of the template to AWS
- Stacks are identified by a name
- Deleting a stack deletes every single artifact that was created by CloudFormation.

Deploying CloudFormation templates

- Manual way:
 - Editing templates in the CloudFormation Designer
 - Using the console to input parameters, etc
- Automated way:
 - Editing templates in a YAML file
 - Using the AWS CLI (Command Line Interface) to deploy the templates
 - Recommended way when you fully want to automate your flow

CloudFormation Building Blocks

Templates components (one course section for each):

1. Resources: your AWS resources declared in the template (MANDATORY)
2. Parameters: the dynamic inputs for your template
3. Mappings: the static variables for your template
4. Outputs: References to what has been created
5. Conditionals: List of conditions to perform resource creation
6. Metadata

Templates helpers:

1. References
2. Functions

Note:

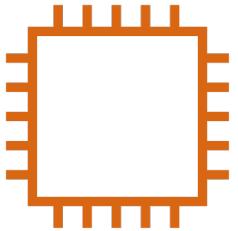
This is an introduction to CloudFormation

- It can take over 3 hours to properly learn and master CloudFormation
- This section is meant so you get a good idea of how it works
- We'll be slightly less hands-on than in other sections

- We'll learn everything we need to answer questions for the exam
- The exam does not require you to actually write CloudFormation
- The exam expects you to understand how to read CloudFormation

Introductory Example

- We're going to create a simple EC2 instance.
 - Then we're going to create to add an Elastic IP to it
 - And we're going to add two security groups to it
 - For now, forget about the code syntax.
 - We'll look at the structure of the files later on
-
- We'll see how in no-time, we are able to get started with CloudFormation!



EC2 Instance

YAML Crash Course

```
1  invoice:      34843
2  date   :     2001-01-23
3  bill-to:
4    given  :   Chris
5    family :  Dumars
6    address:
7      lines: |
8        458 Walkman Dr.
9        Suite #292
10       city   : Royal Oak
11       state  : MI
12       postal : 48046
13 product:
14   - sku        : BL394D
15     quantity   : 4
16     description: Basketball
17     price      : 450.00
18   - sku        : BL4438H
19     quantity   : 1
20     description: Super Hoop
21     price      : 2392.00
```

- YAML and JSON are the languages you can use for CloudFormation.
 - JSON is horrible for CF
 - YAML is great in so many ways
 - Let's learn a bit about it!
-
- Key value Pairs
 - Nested objects
 - Support Arrays
 - Multi line strings
 - Can include comments!

What are resources?

- Resources are the core of your CloudFormation template (MANDATORY)
- They represent the different AWS Components that will be created and configured
- Resources are declared and can reference each other
- AWS figures out creation, updates and deletes of resources for us
- There are over 224 types of resources (!)
- Resource types identifiers are of the form:

AWS::aws-product-name::data-type-name

How do I find resources documentation?

- I can't teach you all of the 224 resources, but I can teach you how to learn how to use them.
- All the resources can be found here:
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>
- Then, we just read the docs ☺
- Example here (for an EC2 instance):
<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>

Analysis of CloudFormation Template

- Going back to the example of the introductory section, let's learn why it was written this way.
- Relevant documentation can be found here:
 - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html>
 - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-security-group.html>
 - <http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-eip.html>

FAQ for resources

- Can I create a dynamic amount of resources?
 - No, you can't. Everything in the CloudFormation template has to be declared. You can't perform code generation there
- Is every AWS Service supported?
 - Almost. Only a select few niches are not there yet
 - You can work around that using AWS Lambda Custom Resources

What are parameters?

- Parameters are a way to provide inputs to your AWS CloudFormation template
- They're important to know about if:
 - You want to reuse your templates across the company
 - Some inputs can not be determined ahead of time
- Parameters are extremely powerful, controlled, and can prevent errors from happening in your templates thanks to types.

When should you use a parameter?

- Ask yourself this:
 - Is this CloudFormation resource configuration likely to change in the future?
 - If so, make it a parameter.
- You won't have to re-upload a template to change its content ☺

Parameters:

SecurityGroupDescription:

Description: Security Group Description
(Simple parameter)

Type: String



DataCumulus | RIZMAXed

Parameters Settings

Parameters can be controlled by all these settings:

- **Type:**
 - String
 - Number
 - CommaDelimitedList
 - List<Type>
 - AWS Parameter (to help catch invalid values – match against existing values in the AWS Account)
- **Description**
- **Constraints**
 - ConstraintDescription (String)
 - Min/MaxLength
 - Min/MaxValue
 - Defaults
 - AllowedValues (array)
 - AllowedPattern (regexp)
 - NoEcho (Boolean)

How to Reference a Parameter

- The `Fn::Ref` function can be leveraged to reference parameters
- Parameters can be used anywhere in a template.
- The shorthand for this in YAML is `!Ref`
- The function can also reference other elements within the template

```
DbSubnet1:  
  Type: AWS::EC2::Subnet  
  Properties:  
    VpcId: !Ref MyVPC
```

Concept: Pseudo Parameters

- AWS offers us pseudo parameters in any CloudFormation template.
- These can be used at any time and are enabled by default

Reference Value	Example Return Value
AWS::AccountId	1234567890
AWS::NotificationARNs	[arn:aws:sns:us-east-1:123456789012:MyTopic]
AWS::NoValue	Does not return a value.
AWS::Region	us-east-2
AWS::StackId	arn:aws:cloudformation:us-east-1:123456789012:stack/MyStack/1c2fa620-982a-11e3-aff7-50e2416294e0
AWS::StackName	MyStack

What are mappings?

- Mappings are fixed variables within your CloudFormation Template.
- They're very handy to differentiate between different environments (dev vs prod), regions (AWS regions), AMI types, etc
- All the values are hardcoded within the template
- Example:

```
Mappings:  
  Mapping01:  
    Key01:  
      Name: Value01  
    Key02:  
      Name: Value02  
    Key03:  
      Name: Value03
```

```
RegionMap:  
  us-east-1:  
    "32": "ami-6411e20d"  
    "64": "ami-7a11e213"  
  us-west-1:  
    "32": "ami-c9c7978c"  
    "64": "ami-cfc7978a"  
  eu-west-1:  
    "32": "ami-37c2f643"  
    "64": "ami-31c2f645"
```

When would you use mappings vs parameters ?

- Mappings are great when you know in advance all the values that can be taken and that they can be deduced from variables such as
 - Region
 - Availability Zone
 - AWS Account
 - Environment (dev vs prod)
 - Etc...
- They allow safer control over the template.
- Use parameters when the values are really user specific

Fn::FindInMap

Accessing Mapping Values

- We use **Fn::FindInMap** to return a named value from a specific key
- **!FindInMap [MapName, TopLevelKey, SecondLevelKey]**

```
AWSTemplateFormatVersion: "2010-09-09"
Mappings:
  RegionMap:
    us-east-1:
      "32": "ami-6411e20d"
      "64": "ami-7a11e213"
    us-west-1:
      "32": "ami-c9c7978c"
      "64": "ami-cfc7978a"
    eu-west-1:
      "32": "ami-37c2f643"
      "64": "ami-31c2f645"
    ap-southeast-1:
      "32": "ami-66f28c34"
      "64": "ami-60f28c32"
    ap-northeast-1:
      "32": "ami-9c03a89d"
      "64": "ami-a003a8a1"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", 32]
      InstanceType: m1.small
```

What are outputs?

- The Outputs section declares *optional* outputs values that we can import into other stacks (if you export them first)!
- You can also view the outputs in the AWS Console or in using the AWS CLI
- They're very useful for example if you define a network CloudFormation, and output the variables such as VPC ID and your Subnet IDs
- It's the best way to perform some collaboration cross stack, as you let expert handle their own part of the stack
- You can't delete a CloudFormation Stack if its outputs are being referenced by another CloudFormation stack

Outputs Example

- Creating a SSH Security Group as part of one template
- We create an output that references that security group

Outputs:

StackSSHSecurityGroup:

Description: The SSH Security Group for our Company

Value: !Ref MyCompanyWideSSHSecurityGroup

Export:

Name: SSHSecurityGroup

Cross Stack Reference

- We then create a second template that leverages that security group
- For this, we use the **Fn::ImportValue** function
- You can't delete the underlying stack until all the references are deleted too.

```
Resources:  
  MySecureInstance:  
    Type: AWS::EC2::Instance  
    Properties:  
      AvailabilityZone: us-east-1a  
      ImageId: ami-a4c7edb2  
      InstanceType: t2.micro  
      SecurityGroups:  
        - !ImportValue SSHSecurityGroup
```

What are conditions used for?

- Conditions are used to control the creation of resources or outputs based on a condition.
- Conditions can be whatever you want them to be, but common ones are:
 - Environment (dev / test / prod)
 - AWS Region
 - Any parameter value
- Each condition can reference another condition, parameter value or mapping

How to define a condition?

Conditions:

```
| CreateProdResources: !Equals [ !Ref EnvType, prod ]
```

- The logical ID is for you to choose. It's how you name condition
- The intrinsic function (logical) can be any of the following:
 - Fn::And
 - Fn::Equals
 - Fn::If
 - Fn::Not
 - Fn::Or

Using a Condition

- Conditions can be applied to resources / outputs / etc...

```
Resources:
```

```
  MountPoint:
```

```
    Type: "AWS::EC2::VolumeAttachment"
```

```
    Condition: CreateProdResources
```

CloudFormation

Must Know Intrinsic Functions

- Ref
- Fn::GetAtt
- Fn::FindInMap
- Fn::ImportValue
- Fn::Join
- Fn::Sub
- Condition Functions (Fn::If, Fn::Not, Fn::Equals, etc...)

Fn::Ref

- The `Fn::Ref` function can be leveraged to reference
 - Parameters => returns the value of the parameter
 - Resources => returns the physical ID of the underlying resource (ex: EC2 ID)
- The shorthand for this in YAML is `!Ref`

```
DbSubnet1:  
  Type: AWS::EC2::Subnet  
  Properties:  
    VpcId: !Ref MyVPC
```

Fn::GetAtt

- Attributes are attached to any resources you create
- To know the attributes of your resources, the best place to look at is the documentation.
- For example: the AZ of an EC2 machine!

```
Resources:
```

```
EC2Instance:
```

```
Type: "AWS::EC2::Instance"
```

```
Properties:
```

```
ImageId: ami-1234567
```

```
InstanceType: t2.micro
```

```
NewVolume:
```

```
Type: "AWS::EC2::Volume"
```

```
Condition: CreateProdResources
```

```
Properties:
```

```
Size: 100
```

```
AvailabilityZone:
```

```
!GetAtt EC2Instance.AvailabilityZone
```

Fn::FindInMap

Accessing Mapping Values

- We use **Fn::FindInMap** to return a named value from a specific key
- **!FindInMap [*MapName*, *TopLevelKey*, *SecondLevelKey*]**

```
AWSTemplateFormatVersion: "2010-09-09"
Mappings:
  RegionMap:
    us-east-1:
      "32": "ami-6411e20d"
      "64": "ami-7a11e213"
    us-west-1:
      "32": "ami-c9c7978c"
      "64": "ami-cfc7978a"
    eu-west-1:
      "32": "ami-37c2f643"
      "64": "ami-31c2f645"
    ap-southeast-1:
      "32": "ami-66f28c34"
      "64": "ami-60f28c32"
    ap-northeast-1:
      "32": "ami-9c03a89d"
      "64": "ami-a003a8a1"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap [RegionMap, !Ref "AWS::Region", 32]
      InstanceType: m1.small
```

Fn::ImportValue

- Import values that are exported in other templates
- For this, we use the **Fn::ImportValue** function

```
Resources:  
  MySecureInstance:  
    Type: AWS::EC2::Instance  
    Properties:  
      AvailabilityZone: us-east-1a  
      ImageId: ami-a4c7edb2  
      InstanceType: t2.micro  
      SecurityGroups:  
        - !ImportValue SSHSecurityGroup
```

Fn::Join

- Join values with a delimiter

```
!Join [ delimiter, [ comma-delimited list of values ] ]
```

- This creates “a:b:c”

```
!Join [ ":", [ a, b, c ] ]
```

Function Fn::Sub

- Fn::Sub, or !Sub as a shorthand, is used to substitute variables from a text. It's a very handy function that will allow you to fully customize your templates.
- For example, you can combine Fn::Sub with References or AWS Pseudo variables!

```
!Sub
- String
- { Var1Name: Var1Value, Var2Name: Var2Value }
```

```
!Sub String
```

Condition Functions

Conditions:

```
| CreateProdResources: !Equals [ !Ref EnvType, prod ]
```

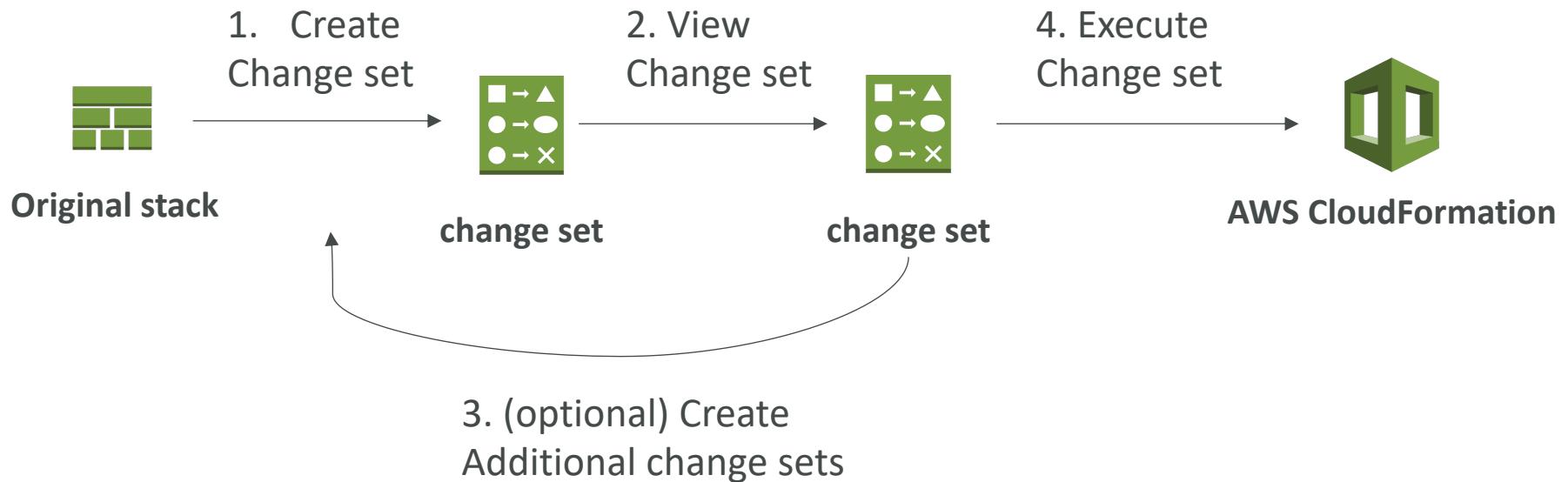
- The logical ID is for you to choose. It's how you name condition
- The intrinsic function (logical) can be any of the following:
 - Fn::And
 - Fn::Equals
 - Fn::If
 - Fn::Not
 - Fn::Or

CloudFormation Rollbacks

- Stack Creation Fails:
 - Default: everything rolls back (gets deleted). We can look at the log
 - Option to disable rollback and troubleshoot what happened
- Stack Update Fails:
 - The stack automatically rolls back to the previous known working state
 - Ability to see in the log what happened and error messages

ChangeSets

- When you update a stack, you need to know what changes before it happens for greater confidence
- ChangeSets won't say if the update will be successful



From: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-changesets.html>

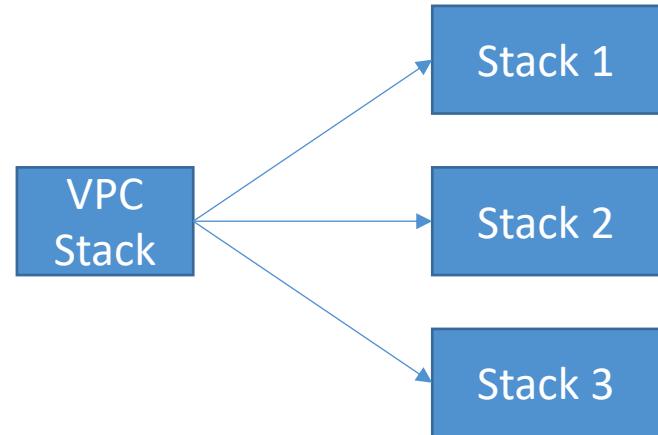
Nested stacks

- Nested stacks are stacks as part of other stacks
- They allow you to isolate repeated patterns / common components in separate stacks and call them from other stacks
- Example:
 - Load Balancer configuration that is re-used
 - Security Group that is re-used
- Nested stacks are considered best practice
- To update a nested stack, always update the parent (root stack)

CloudFormation – Cross vs Nested Stacks

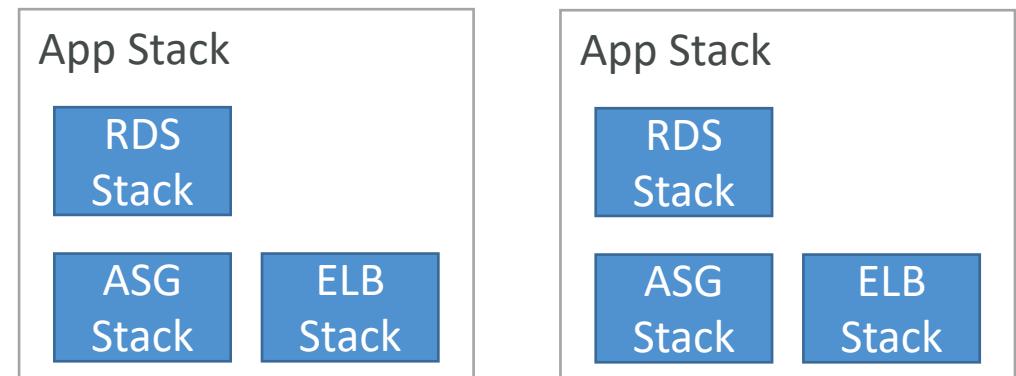
- **Cross Stacks**

- Helpful when stacks have different lifecycles
- Use Outputs Export and Fn::ImportValue
- When you need to pass export values to many stacks (VPC Id, etc...)



- **Nested Stacks**

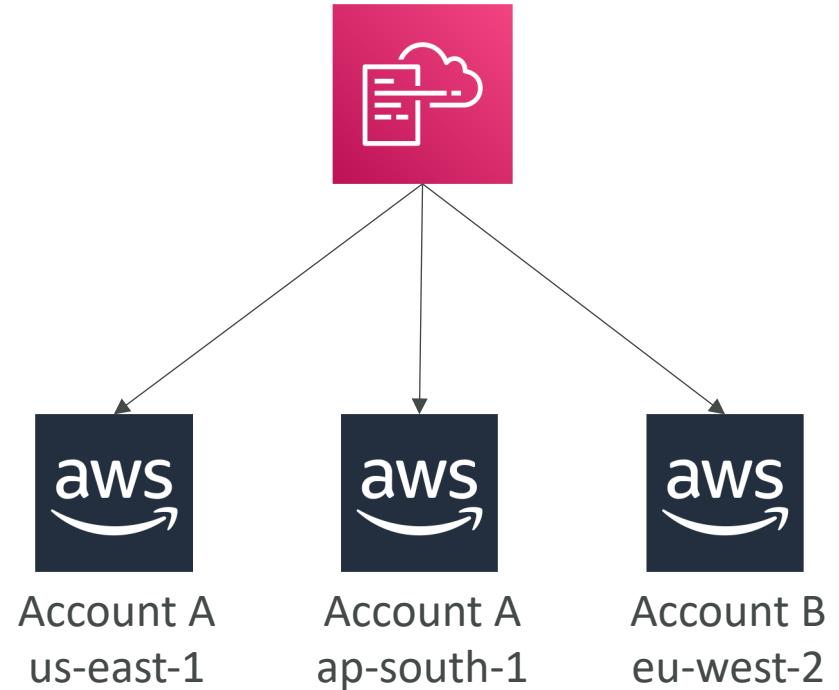
- Helpful when components must be re-used
- Ex: re-use how to properly configure an Application Load Balancer
- The nested stack only is important to the higher level stack (it's not shared)



CloudFormation - StackSets

- Create, update, or delete stacks across multiple accounts and regions with a single operation
- Administrator account to create StackSets
- Trusted accounts to create, update, delete stack instances from StackSets
- When you update a stack set, *all* associated stack instances are updated throughout all accounts and regions.

CloudFormation **StackSet**
Admin Account



CloudFormation for DB Specialty Exam

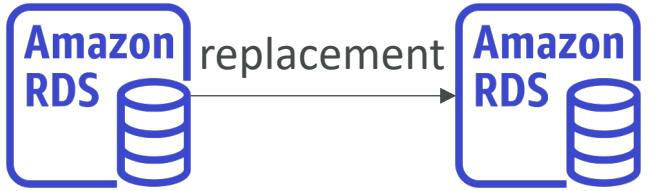


RDS and CloudFormation

- AWS::RDS::DBInstance
<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-rds-database-instance.html>
- Important Parameters:
 - **DeleteAutomatedBackups**: remove automated backups immediately after the DB instance is deleted. Default value: *remove all automated backups*
 - **DeletionProtection**: enable deletion protection on the DB in RDS. The database can't be deleted while it has DeletionProtection on.
 - **MasterUsername / MasterPassword**: must be provided as text values (we'll see how to handle that securely later on)
 - **EnableIAMDatabaseAuthentication**: can be enabled, does not replace Master username & password
 - **DBClusterIdentifier**: (optional) if the DB belongs to an Aurora DB cluster

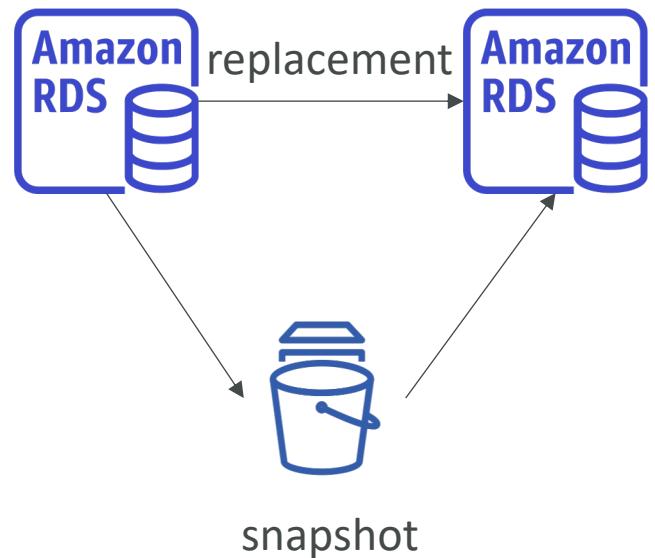
RDS and CloudFormation – DB Updates

- If Update Requires: Replacement:
 - New DB is created
 - Links are updated
 - Old DB is deleted
- If a DB instance is deleted or replaced during an update, AWS CloudFormation **deletes all automated snapshots**.
- However, it **retains manual DB snapshots**
- During an update that requires replacement, you can apply a **stack policy** to prevent DB instances from being replaced



RDS and CloudFormation – database update

- Take a snapshot before updating an update. If you don't, you lose the data when AWS CloudFormation replaces your DB instance.
To preserve your data:
 - Deactivate any applications that are using the DB instance so that there's no activity on the DB instance.
 - Create a snapshot of the DB instance
 - If you want to restore your instance using a DB snapshot, modify the updated template with your DB instance changes and add the **DBSnapshotIdentifier** property with the ID of the DB snapshot that you want to use.
 - After you restore a DB instance with a **DBSnapshotIdentifier** property, you must specify the same **DBSnapshotIdentifier** property for any future updates to the DB instance.
 - Update the stack.
- **DBSnapshotIdentifier** is also useful for mass creating dev/test databases with some data from prod

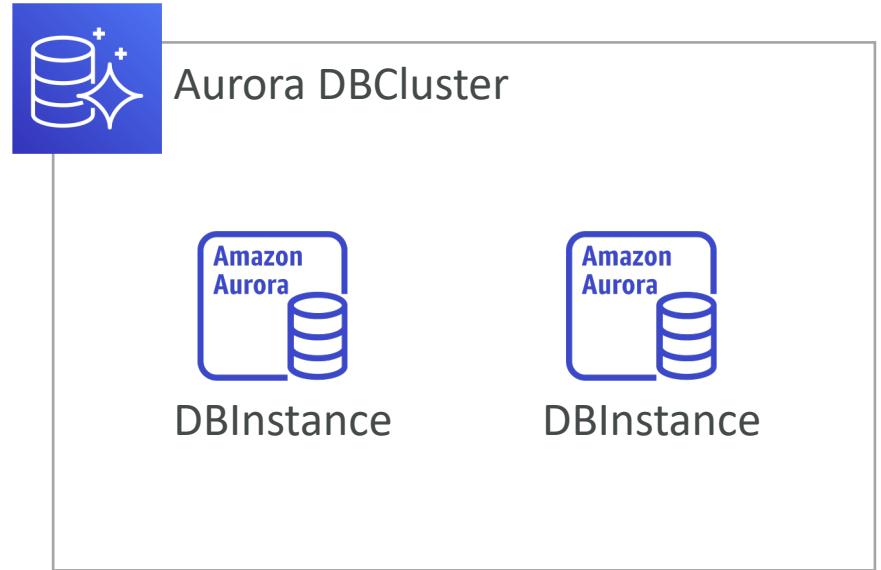


CloudFormation protections

- CloudFormation Stack Termination Protection: can't delete the stack without removing it (but you can still update the stack)
- CloudFormation Stack Policy: prevent updates to CloudFormation templates.
- CloudFormation DeletionPolicy: what happens if a resource gets deleted
 - Delete: delete the resource (default behaviour for most resources)
 - Retain: do not delete the resources
 - Snapshot: (only to resources that support snapshot) snapshot then delete
- The default CloudFormation behavior for RDS depends on DBClusterIdentifier :
 - If DBClusterIdentifier not specified (= non Aurora), CloudFormation will snapshot
 - If DBClusterIdentifier is specified (=Aurora), CloudFormation deletes the DB instance

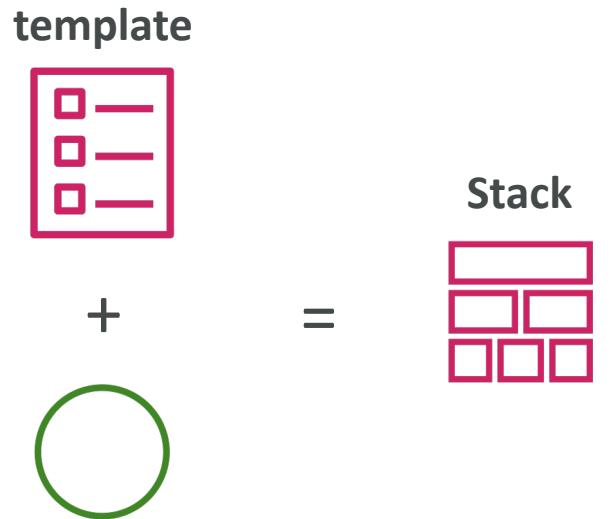
Aurora with CloudFormation

- First create an AWS::RDS::DBCluster
- Then add AWS::RDS::DBInstance
- Default **DeletionPolicy** of RDS::DBCluster is Snapshot
 - Means if individual DBInstance are deleted, no snapshot is made (default)
 - If the DBCluster is deleted, a snapshot will be made
- RDS::DBInstance will inherit configurations



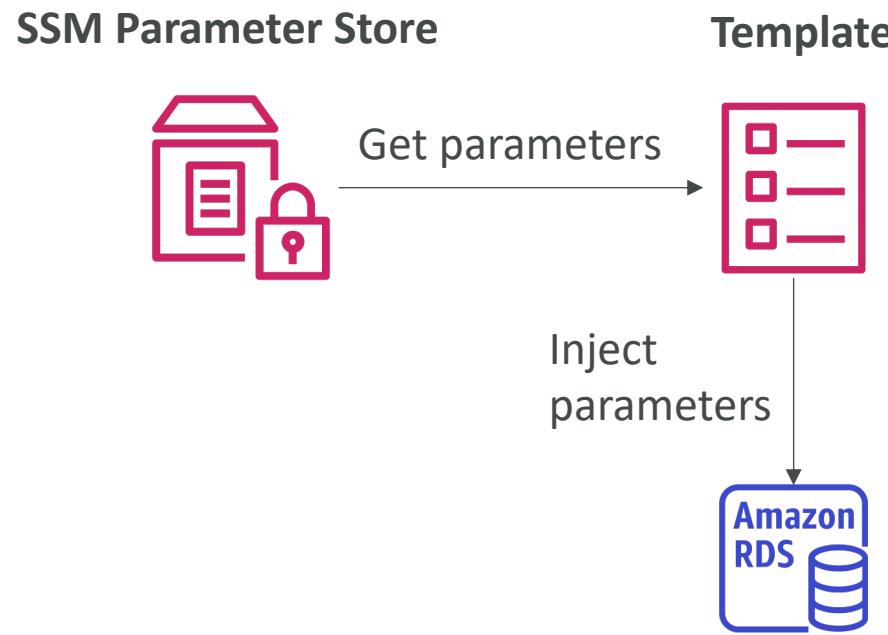
CloudFormation Parameters

- Parameters are referenced using !Ref
- They can have 4 different sources:
 - (static) Given using the CLI / Management console
 - (static) Given using a flat file from local filesystem or S3
 - (dynamic) From the SSM Parameter Store
 - (dynamic) From AWS Secrets Manager
- The last two are the most secure way
- Exam trap: you cannot pass in KMS encrypted parameters through
 - Use SSM Parameter Store or Secrets Manager instead !



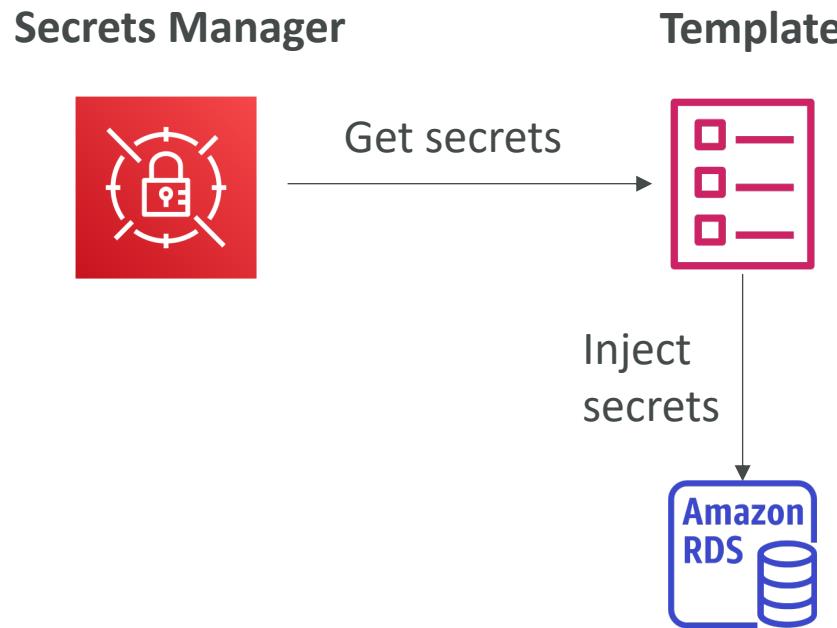
- Parameters file**
- Local File
 - Amazon S3
 - CLI / Console parameter

CloudFormation Parameters - SSM



- Retrieve a String (plaintext) or SecureString (KMS encrypted) from SSM Parameter Store at runtime!
 - Amazing to store environment specific variables
 - Hierarchical way of storing parameters
- **SSM CF parameters:** will be resolved every time you run the template, does not work for SecureString.
- **Dynamic parameter pattern:** '{{resolve:ssm:parameter-name:version}}', you must specify the exact version, works for SecureString.

CF Parameters – Secrets Manager

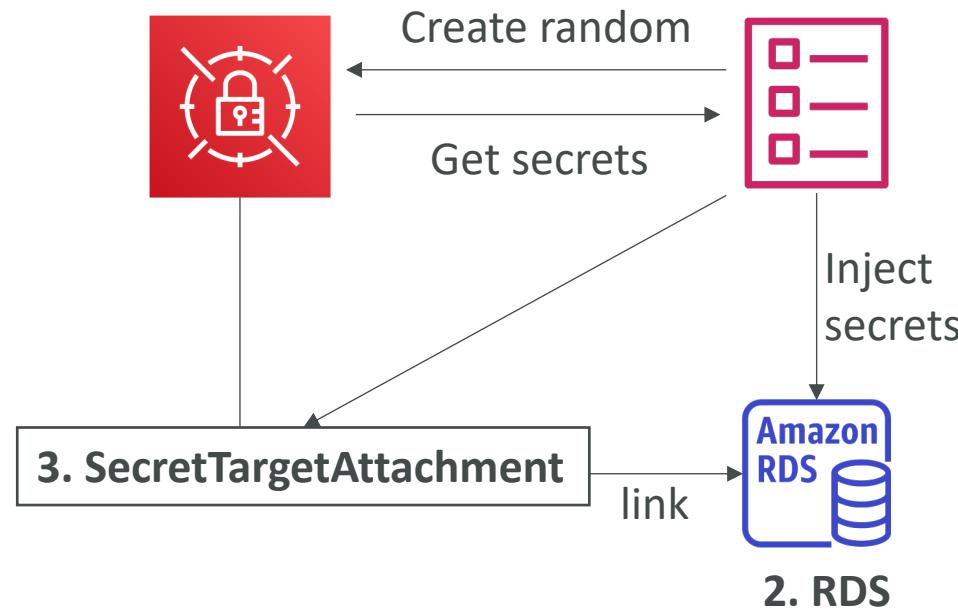


- Secrets rotation + integration with RDS = ❤️
- Dynamic parameter pattern: `'{{resolve:ssm:parameter-name:version}}'`, you must specify the exact version
- Neither Secrets Manager nor CloudFormation logs or persists any resolved secret value
- Updating a secret in Secrets Manager does not automatically update the secret in CloudFormation
 - To manage updating the secret in your template, consider using `version-id` to specify the version of your secret.
- `{{resolve:secretsmanager:secret-id:secret-string:json-key:version-stage:version-id}}`

Secrets Manager + RDS

1. Secrets Manager

Template



I. Create a **AWS::SecretsManager::Secret**

- Specify GenerateSecretString to randomly generate a secret

2. Create an **AWS::RDS::DBInstance** that references that secret

- Using dynamic references

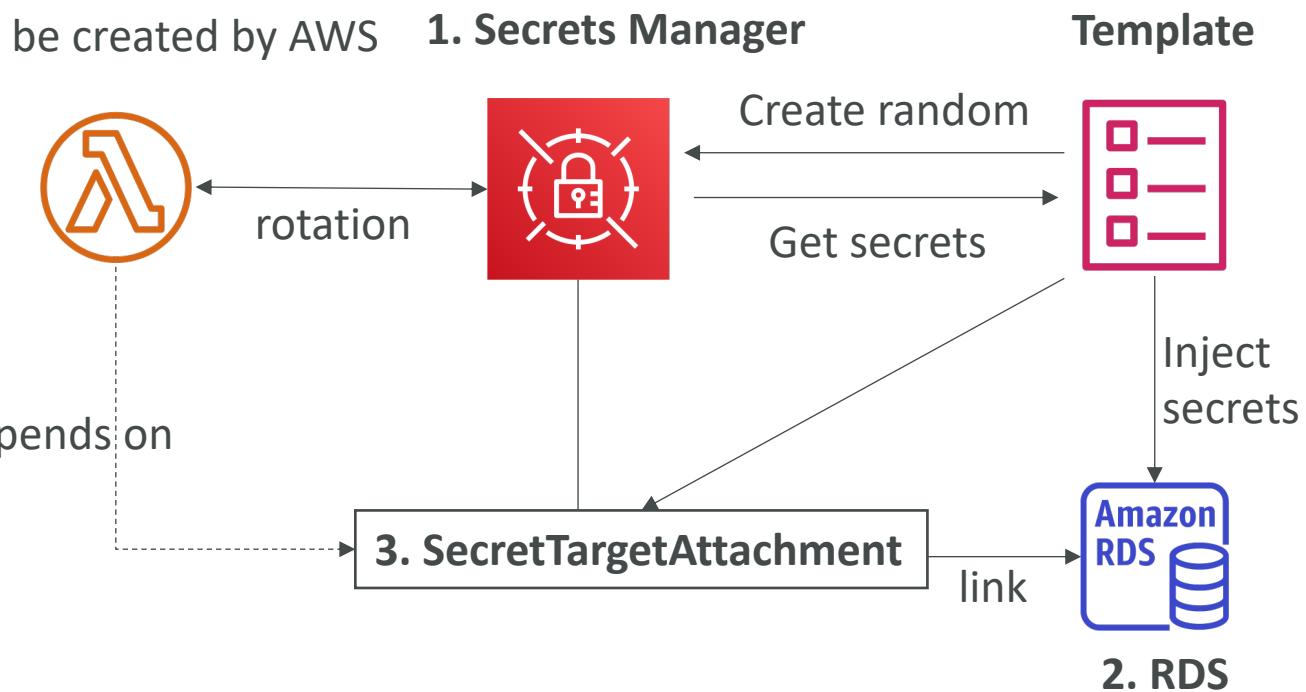
3. Create a **AWS::SecretsManager::SecretTargetAttachment**

- Link to Secrets Manager secret and the RDS database together
- Will add RDS properties to Secret Manager JSON

Secrets Manager Rotation Schedule

4. Lambda Function

Can be created by AWS



4. Create a

AWS::SecretsManager::RotationSchedule

- It will create a Lambda function
- You must do this after creating the secret attachment

Other CloudFormation concepts

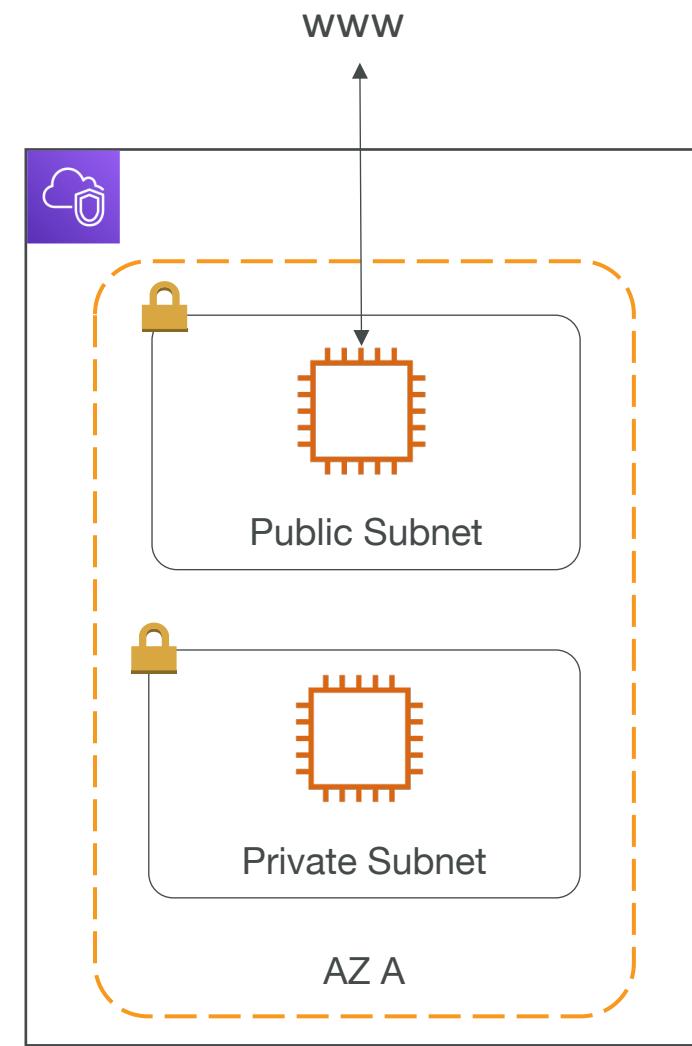
- CloudFormation Drift:
 - See if the resource configurations have changed from the template over time
- Stack Set:
 - Apply the same CloudFormation template into many accounts / regions
- Change Set:
 - Evaluate the changes that will be applied before applying them (safety)

VPC Primer

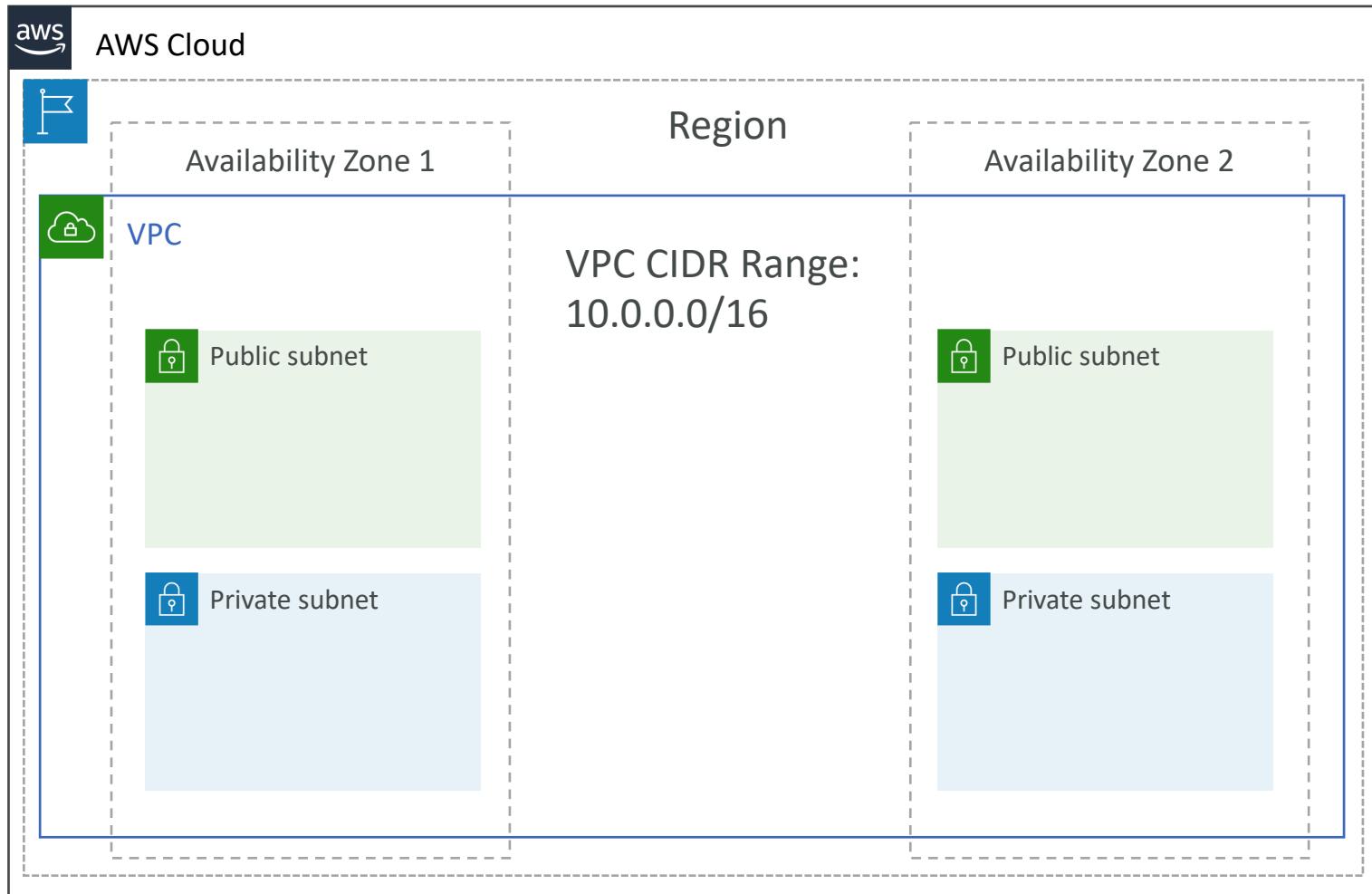
Quick Overview of VPC

VPC & Subnets Primer

- **VPC**: private network to deploy your resources (regional resource)
- **Subnets** allow you to partition your network inside your VPC (Availability Zone resource)
- A **public subnet** is a subnet that is accessible from the internet
- A **private subnet** is a subnet that is not accessible from the internet
- To define access to the internet and between subnets, we use **Route Tables**.

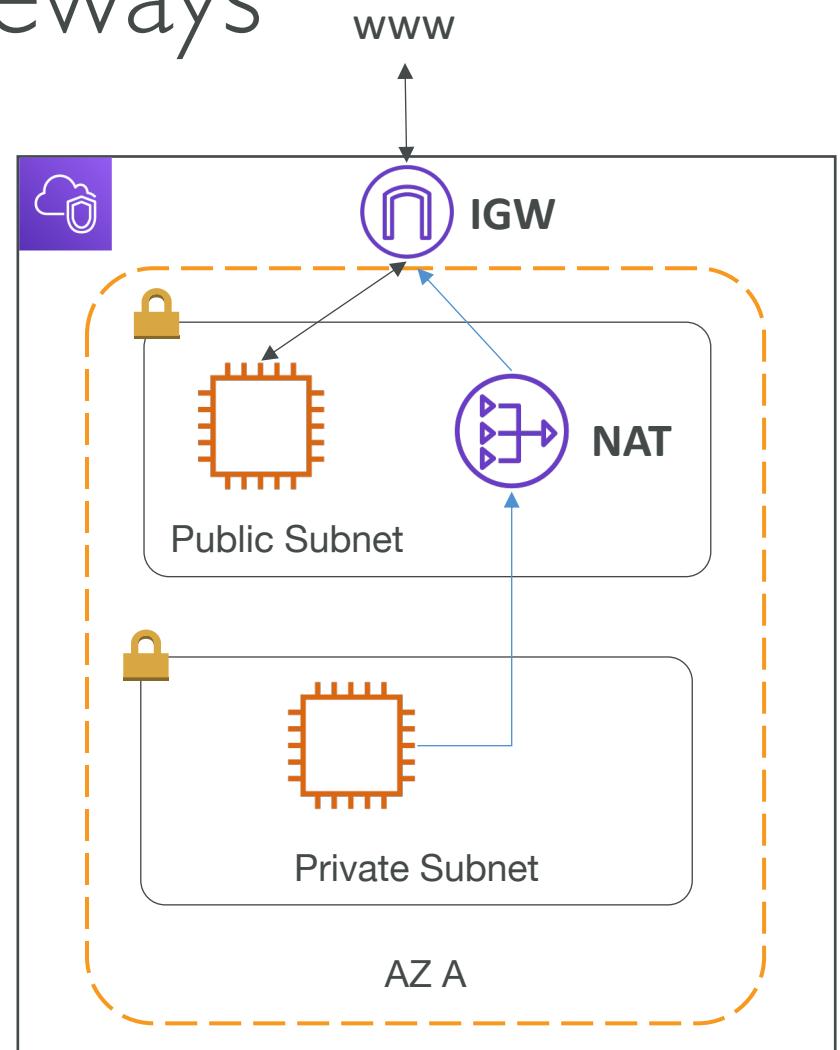


VPC Diagram



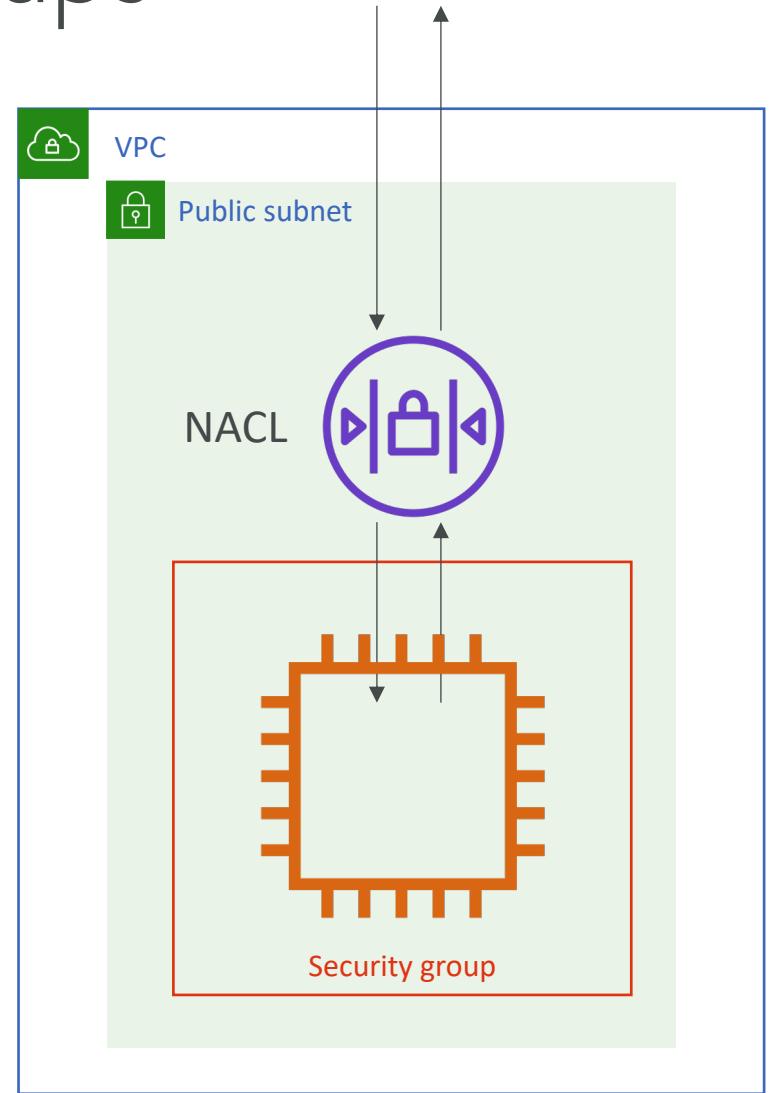
Internet Gateway & NAT Gateways

- Internet Gateways helps our VPC instances connect with the internet
- Public Subnets have a route to the internet gateway.
- NAT Gateways (AWS-managed) & NAT Instances (self-managed) allow your instances in your Private Subnets to access the internet while remaining private



Network ACL & Security Groups

- NACL (Network ACL)
 - A firewall which controls traffic from and to subnet
 - Can have ALLOW and DENY rules
 - Are attached at the **Subnet** level
 - Rules only include IP addresses
- Security Groups
 - A firewall that controls traffic to and from **an ENI / an EC2 Instance**
 - Can have only ALLOW rules
 - Rules include IP addresses and other security groups



Network ACLs vs Security Groups

Security Group	Network ACL
Operates at the instance level	Operates at the subnet level
Supports allow rules only	Supports allow rules and deny rules
Is stateful: Return traffic is automatically allowed, regardless of any rules	Is stateless: Return traffic must be explicitly allowed by rules
We evaluate all rules before deciding whether to allow traffic	We process rules in number order when deciding whether to allow traffic
Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on	Automatically applies to all instances in the subnets it's associated with (therefore, you don't have to rely on users to specify the security group)

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Security.html#VPC_Security_Comparison



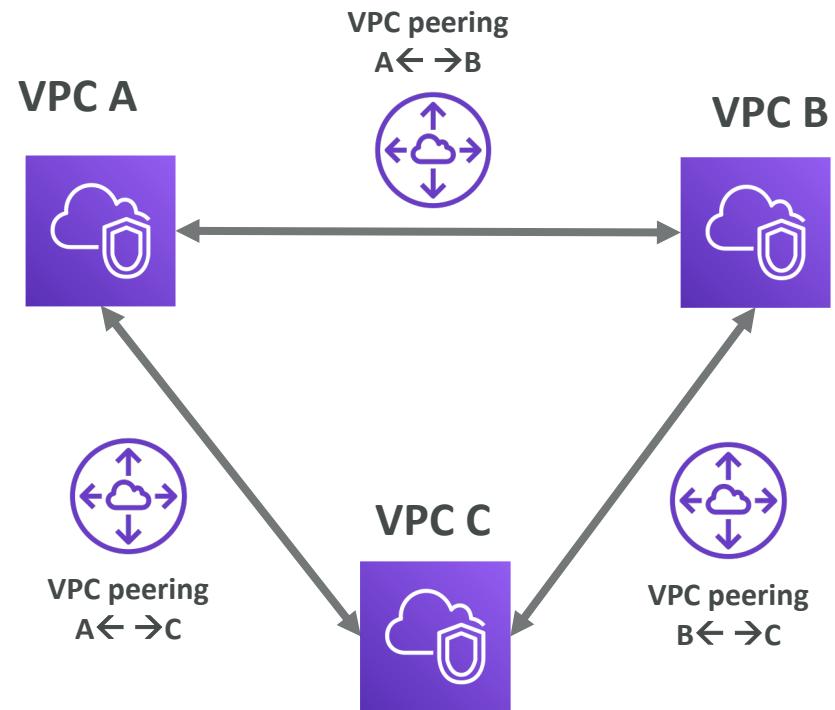


VPC Flow Logs

- Capture information about IP traffic going into your interfaces:
 - VPC Flow Logs
 - Subnet Flow Logs
 - Elastic Network Interface Flow Logs
- Helps to monitor & troubleshoot connectivity issues. Example:
 - Subnets to internet
 - Subnets to subnets
 - Internet to subnets
- Captures network information from AWS managed interfaces too: Elastic Load Balancers, ElastiCache, RDS, Aurora, etc...
- VPC Flow logs data can go to S3 / CloudWatch Logs

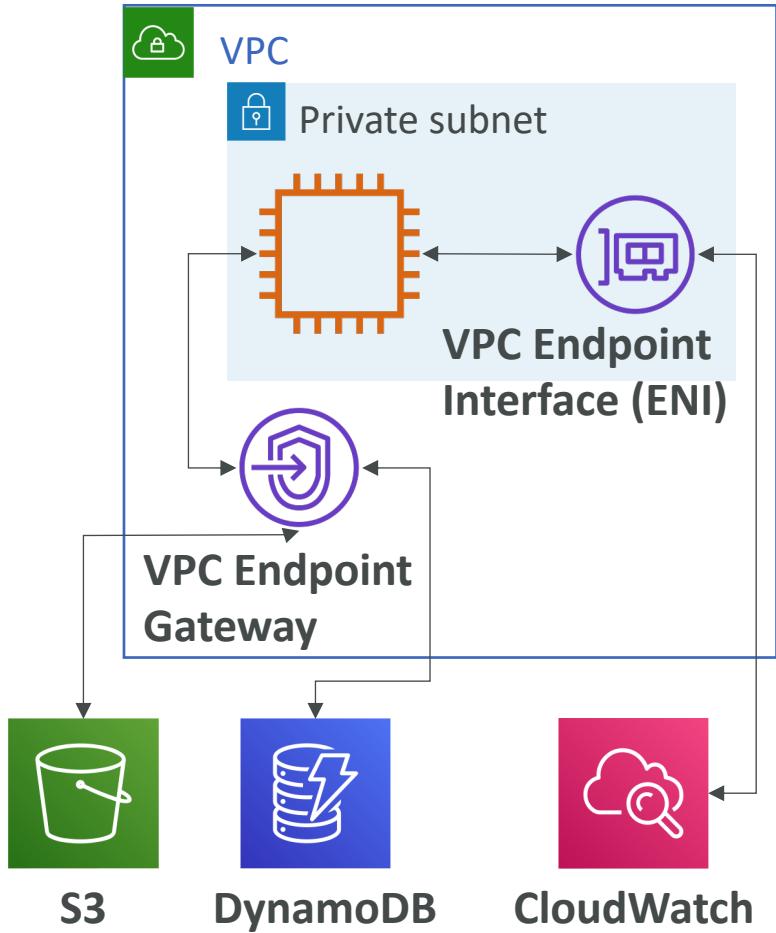
VPC Peering

- Connect two VPC, privately using AWS' network
- Make them behave as if they were in the same network
- Must not have overlapping CIDR (IP address range)
- VPC Peering connection is **not transitive** (must be established for each VPC that need to communicate with one another)



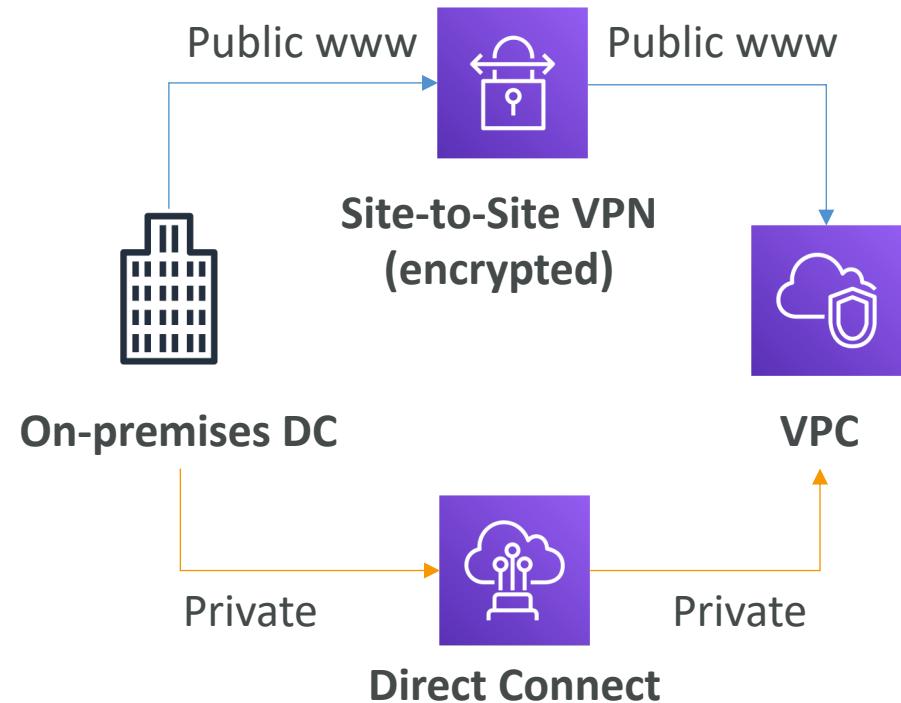
VPC Endpoints

- Endpoints allow you to connect to AWS Services **using a private network** instead of the public www network
- This gives you enhanced security and lower latency to access AWS services
- VPC Endpoint Gateway: S3 & DynamoDB
- VPC Endpoint Interface: the rest
- Only used within your VPC



Site to Site VPN & Direct Connect

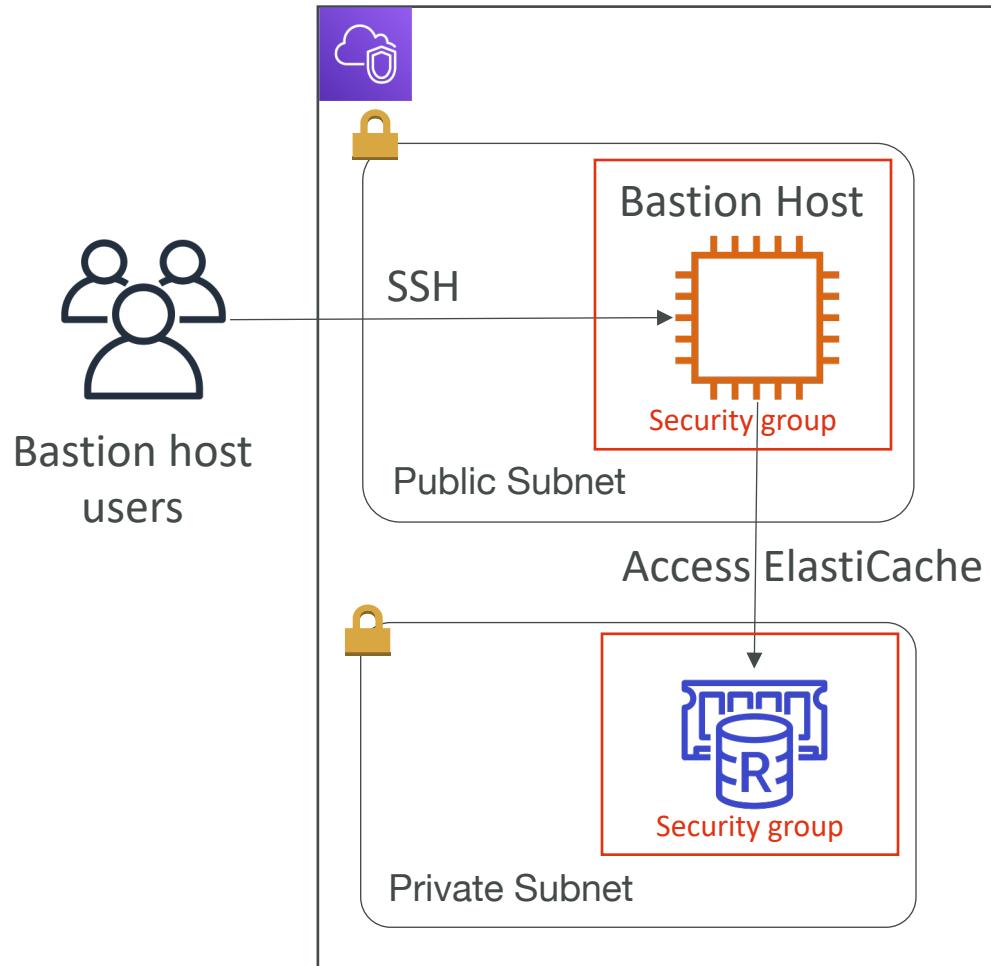
- Site to Site VPN
 - Connect an on-premises VPN to AWS
 - The connection is automatically encrypted
 - Goes over the public internet
- Direct Connect (DX)
 - Establish a physical connection between on-premises and AWS
 - The connection is private, secure and fast
 - Goes over a private network
 - Takes at least a month to establish
- Note: Site-to-site VPN and Direct Connect cannot access VPC endpoints



VPC Closing Comments

- **VPC:** Virtual Private Cloud
- **Subnets:** Tied to an AZ, network partition of the VPC
- **Internet Gateway:** at the VPC level, provide Internet Access
- **NAT Gateway / Instances:** give internet access to private subnets
- **NACL:** Stateless, subnet rules for inbound and outbound
- **Security Groups:** Stateful, operate at the EC2 instance level or ENI
- **VPC Peering:** Connect two VPC with non overlapping IP ranges, non transitive
- **VPC Endpoints:** Provide private access to AWS Services within VPC
- **VPC Flow Logs:** network traffic logs
- **Site to Site VPN:** VPN over public internet between on-premises DC and AWS
- **Direct Connect:** direct private connection to a AWS

Bastion Hosts



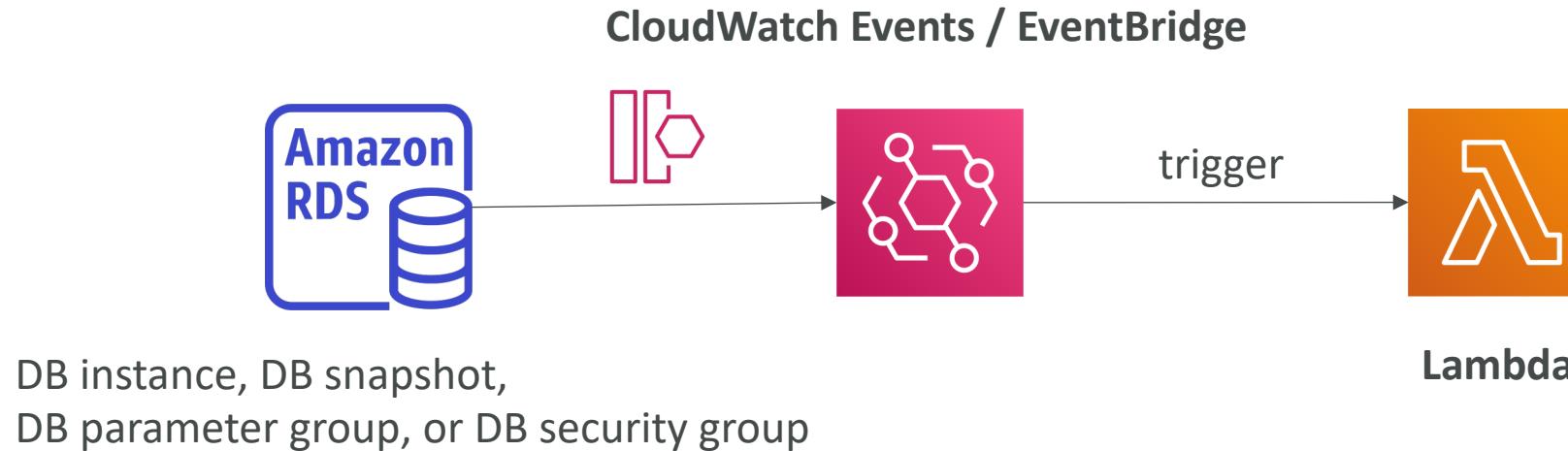
- We can use a Bastion Host to access our private RDS databases, ElastiCache clusters, etc....
- The bastion is in the public subnet which is then connected to all other private subnets
- Bastion Host security group must be tightened
- Make sure DB Security Groups allows the bastion security group in

Other Topics

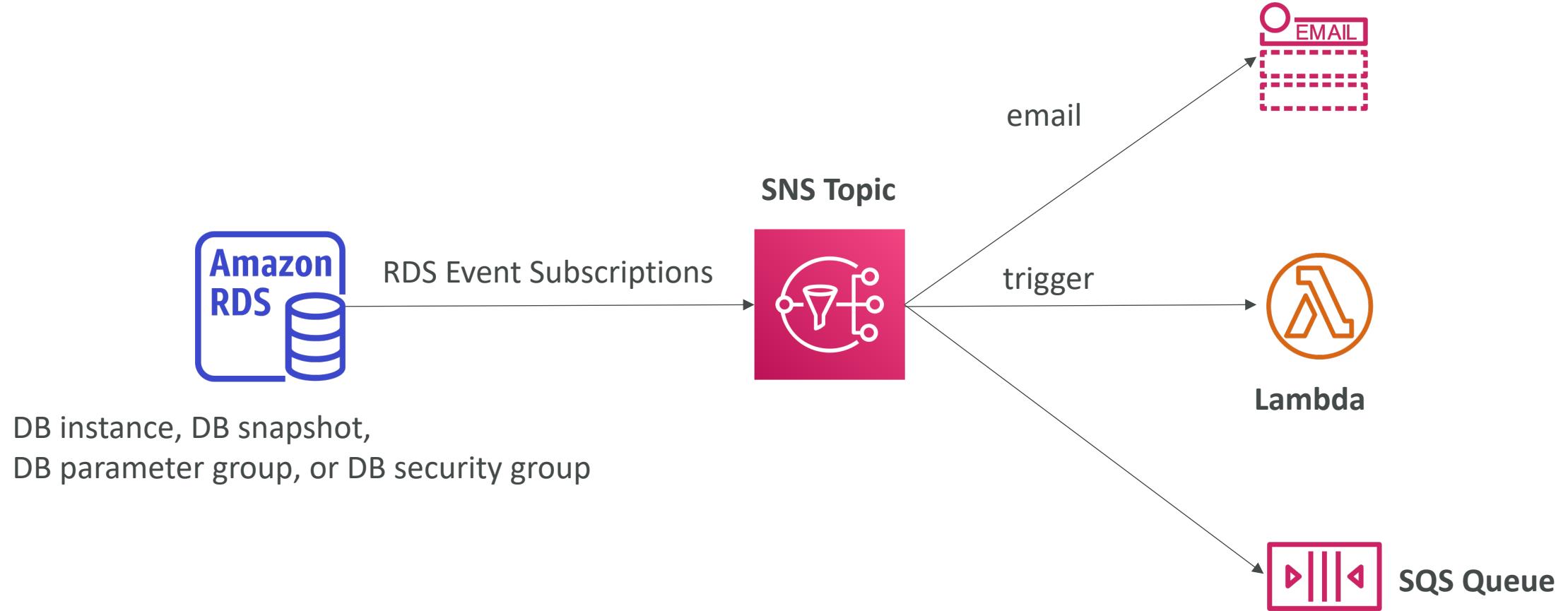
Services worth a look!

Lambda use cases

- React to events happening to your database using CloudWatch Events / EventBridge

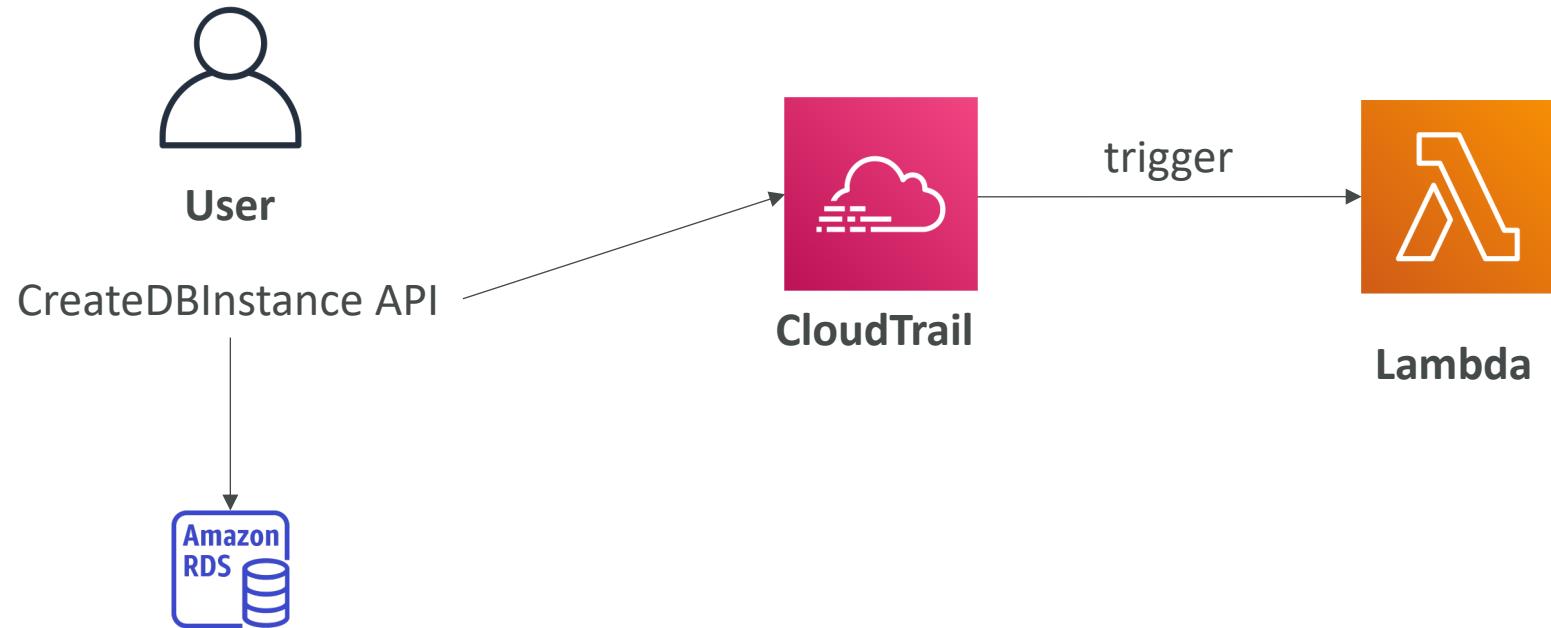


Alternative – SNS topics & RDS Event Subscription



Lambda use cases

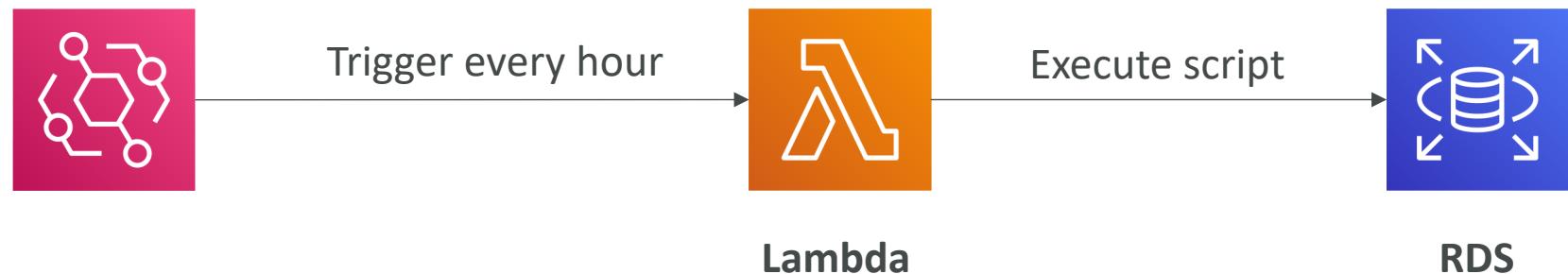
- React to API calls using CloudTrail



Lambda Cron jobs

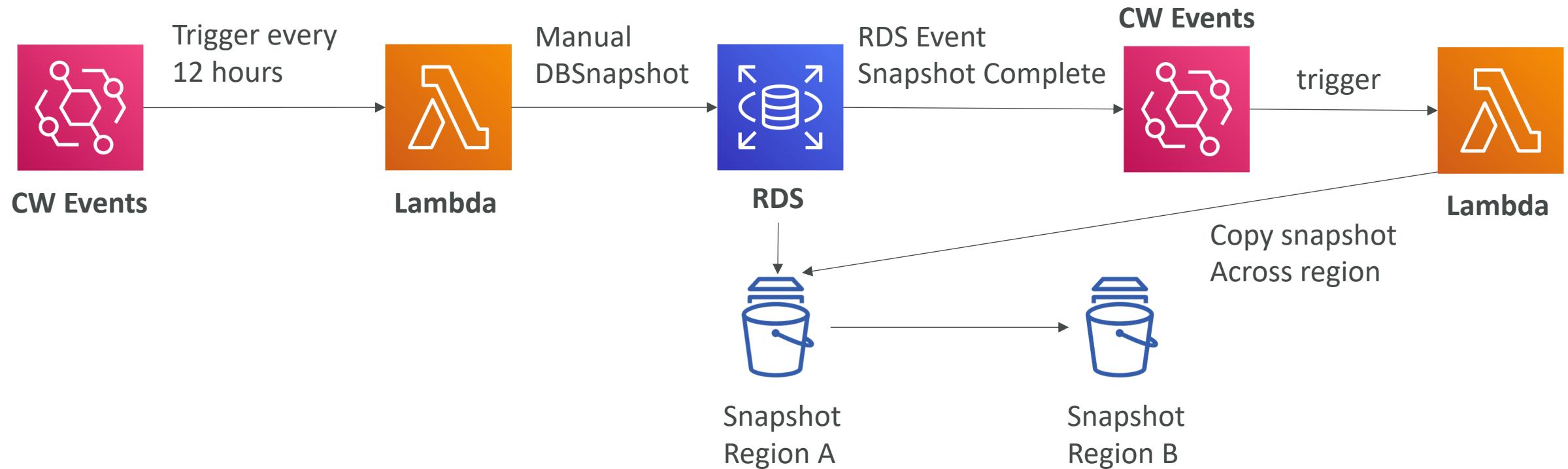
- CW Events / EventBridge can trigger Lambda on a cron schedule
- Max execution time of a Lambda function: 15 minutes
- Languages: Python, JavaScript, Java, etc... except Docker

CloudWatch Events / EventBridge



Lambda RDS backup automation – DR Strategy

- Automated RDS backups: once every day during maintenance window
- Manual backups: whenever you want (ex: every 12 hours)



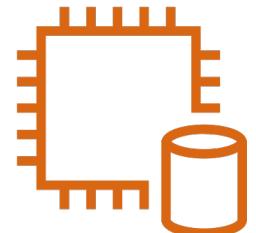
AWS Server Migration Service (SMS)



- Migrate entire VMs to AWS, improvement over EC2 VM Import/Export service
- Not meant for database migration (use DMS instead)
- That means the OS, the data, everything is kept intact
- After loading the VM onto EC2, then you can update the OS, the data, make an AMI
- Therefore SMS is used to Re-host
- Only works with VMware vSphere, Windows Hyper-V, and Azure VM
- Every replication creates an EBS snapshot / AMI ready for deployment on EC2
- Every replication is incremental
- “One time migrations”, or “replication every *interval*” option
- Not continuous, unlike DMS (Database Migration Service)

EBS-optimized instances

Instance size	Maximum bandwidth (Mbps)
i3.large	425
i3.xlarge	850
i3.2xlarge	1,700
i3.4xlarge	3,500
i3.8xlarge	7,000
i3.16xlarge	14,000
i3.metal	19,000



EC2 Instance
EBS-Optimized



Dedicated bandwidth



EBS Volume (io1)
Dedicated IOPS

Choose an EBS-optimized instance that provides more dedicated Amazon EBS throughput than your application needs; otherwise, the connection between Amazon EBS and Amazon EC2 can become a performance bottleneck.

EBS-optimized instances

- The same applies to RDS databases with io1 EBS volumes
- Increase the instance size to leverage all the PIOPS from your volume
- Getting maximum performance from EBS-optimized instances
 - Use EBSIOBalance% and EBSByteBalance% metrics
 - Instances with a consistently low balance percentage are candidates to size up
 - Instances where the balance percentage never drops below 100% are candidates for downsizing
- <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-optimized.html>

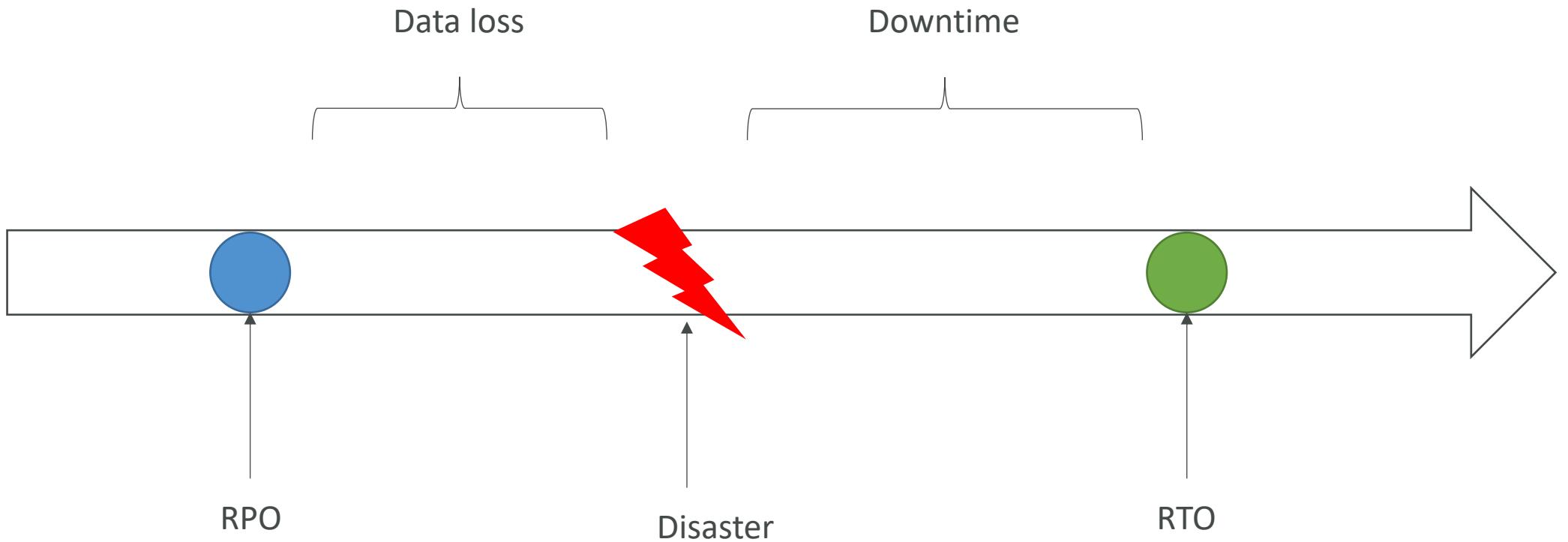
Transferring large amount of data into AWS

- Example: transfer 200 TB of data in the cloud. We have a 100 Mbps internet connection.
- Over the internet / Site-to-Site VPN:
 - Immediate to setup
 - Will take $200(\text{TB}) * 1000(\text{GB}) * 1000(\text{MB}) * 8(\text{Mb}) / 100 \text{ Mbps} = 16,000,000 \text{s} = 185\text{d}$
- Over direct connect 1 Gbps:
 - Long for the one-time setup (over a month)
 - Will take $200(\text{TB}) * 1000(\text{GB}) * 8(\text{Gb}) / 1 \text{ Gbps} = 1,600,000 \text{s} = 18.5\text{d}$
- Over Snowball:
 - Will take 2 to 3 snowballs in parallel
 - Takes about 1 week for the end-to-end transfer
 - Can be combined with DMS
- For on-going replication / transfers: Site-to-Site VPN or DX with DMS

Disaster Recovery Overview

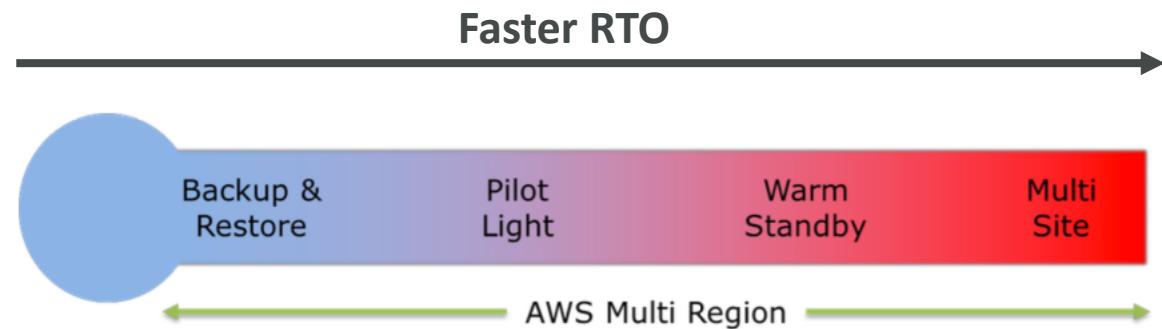
- Any event that has a negative impact on a company's business continuity or finances is a disaster
- Disaster recovery (DR) is about preparing for and recovering from a disaster
- What kind of disaster recovery?
 - On-premise => On-premise: traditional DR, and very expensive
 - On-premise => AWS Cloud: hybrid recovery
 - AWS Cloud Region A => AWS Cloud Region B
- Need to define two terms:
 - RPO: Recovery Point Objective
 - RTO: Recovery Time Objective

RPO and RTO

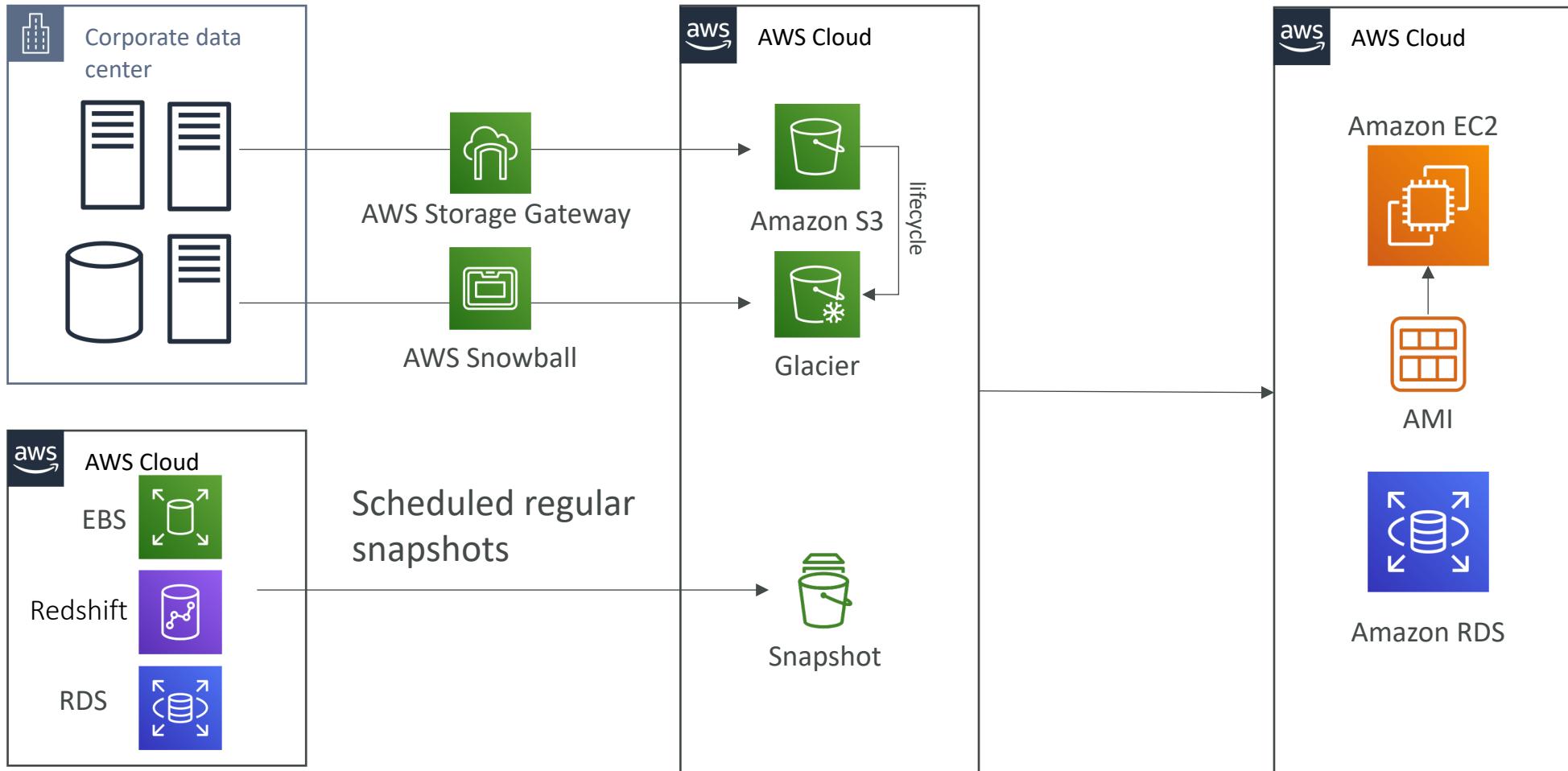


Disaster Recovery Strategies

- Backup and Restore
- Pilot Light
- Warm Standby
- Hot Site / Multi Site Approach

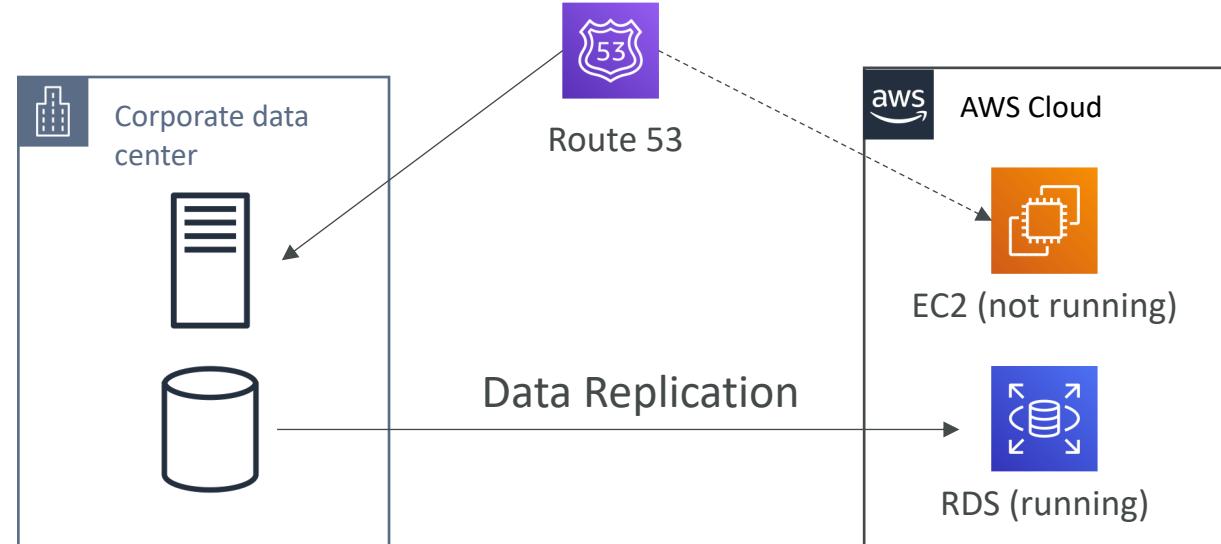


Backup and Restore (High RPO)



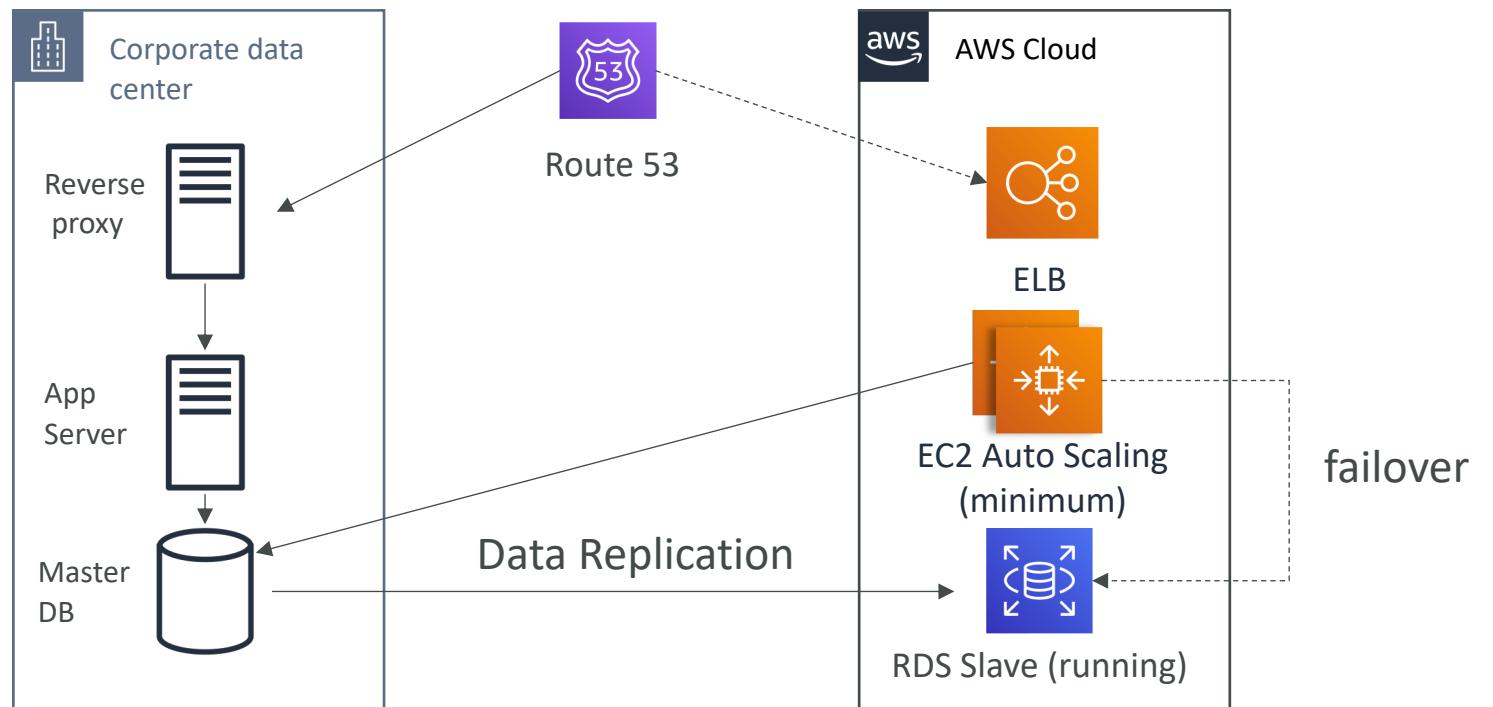
Disaster Recovery – Pilot Light

- A small version of the app is always running in the cloud
- Useful for the critical core (pilot light)
- Very similar to Backup and Restore
- Faster than Backup and Restore as critical systems are already up



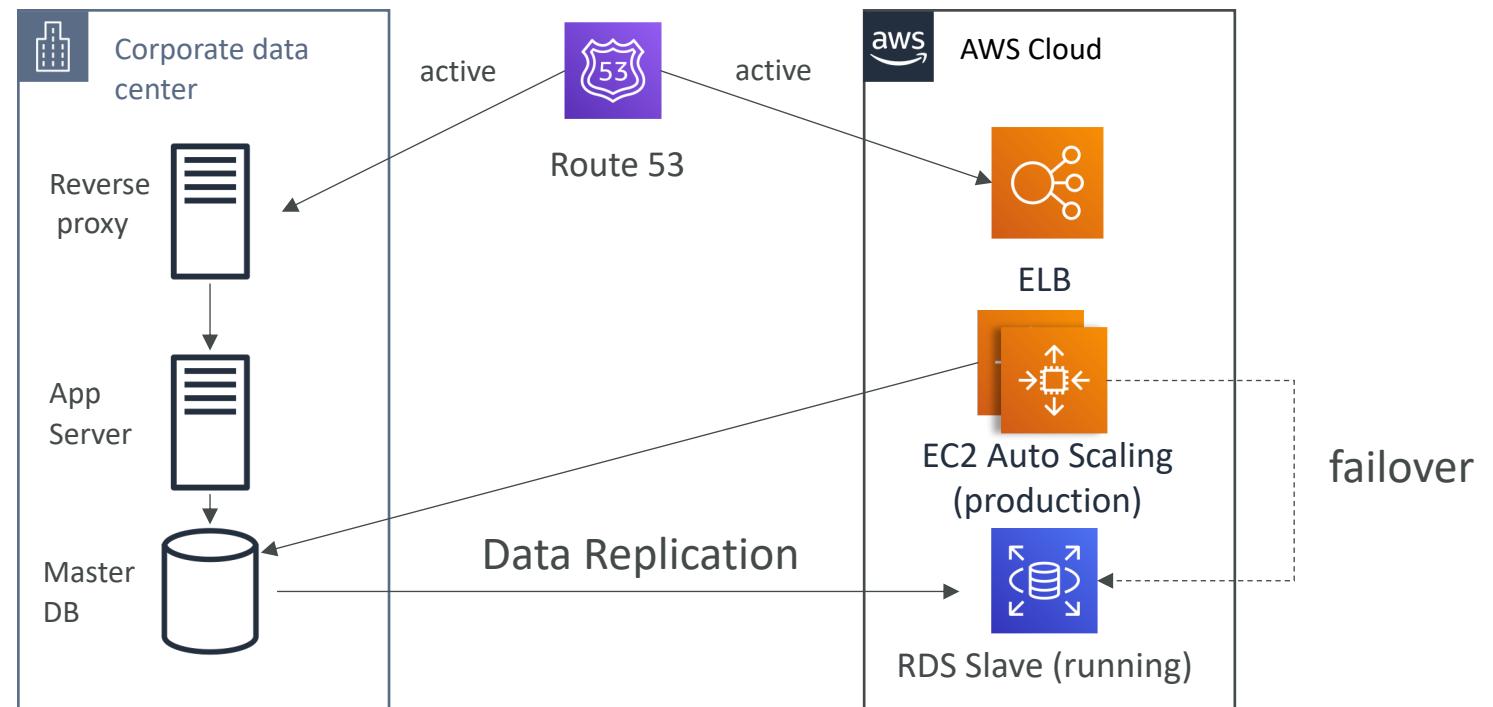
Warm Standby

- Full system is up and running, but at minimum size
- Upon disaster, we can scale to production load

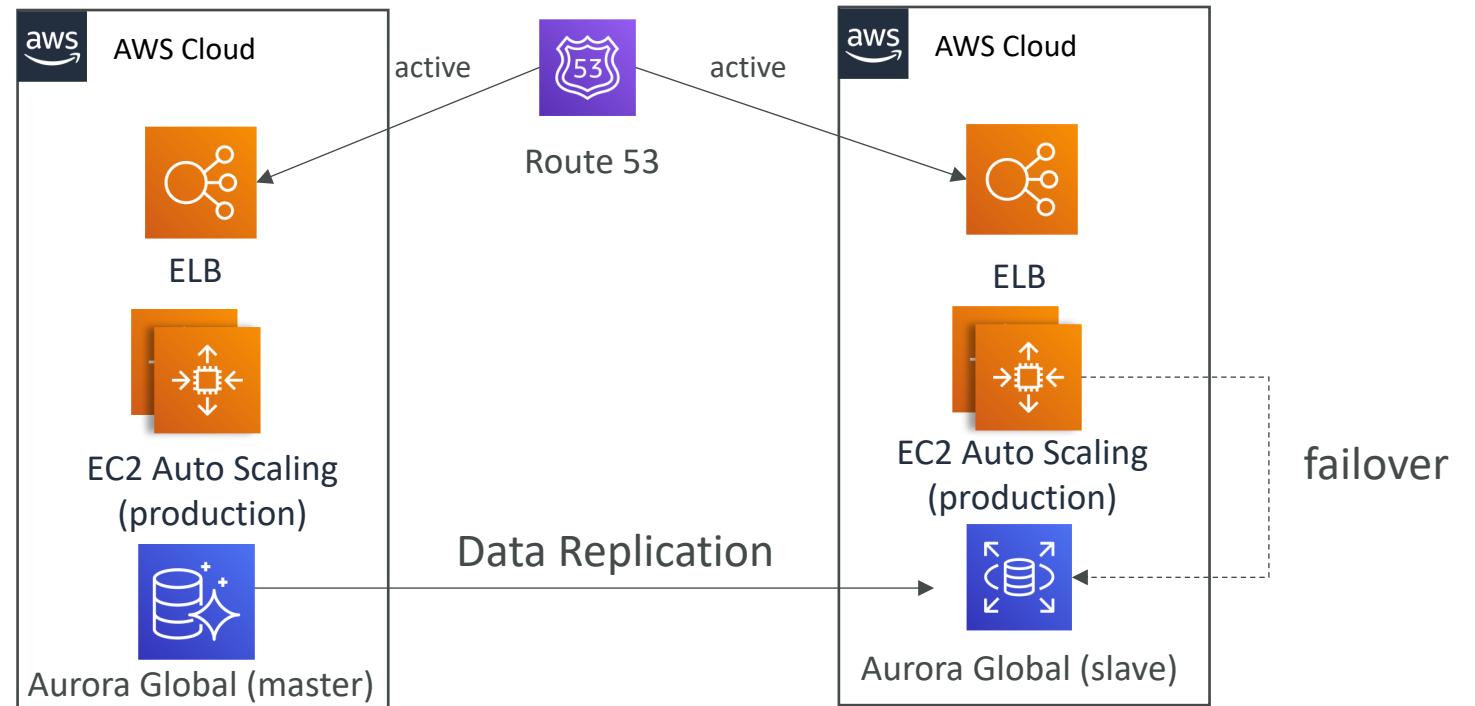


Multi Site / Hot Site Approach

- Very low RTO (minutes or seconds) – very expensive
- Full Production Scale is running AWS and On Premise



All AWS Multi Region



Disaster Recovery Tips

- **Backup**
 - EBS Snapshots, RDS automated backups / Snapshots, etc...
 - Regular pushes to S3 / S3 IA / Glacier, Lifecycle Policy, Cross Region Replication
 - From On-Premise: Snowball or Storage Gateway
- **High Availability**
 - Use Route53 to migrate DNS over from Region to Region
 - RDS Multi-AZ, ElastiCache Multi-AZ, EFS, S3
 - Site to Site VPN as a recovery from Direct Connect
- **Replication**
 - RDS Replication (Cross Region), AWS Aurora + Global Databases
 - Database replication from on-premise to RDS
 - Storage Gateway
- **Automation**
 - CloudFormation / Elastic Beanstalk to re-create a whole new environment
 - Recover / Reboot EC2 instances with CloudWatch if alarms fail
 - AWS Lambda functions for customized automations
- **Chaos**
 - Netflix has a “simian-army” randomly terminating EC2

Tackling the exam questions

“Perfect” practice makes perfect!

Question |

A media company is running a critical production application that uses Amazon RDS for PostgreSQL with Multi-AZ deployments. The database size is currently 25 TB. The IT director wants to migrate the database to Amazon Aurora PostgreSQL with minimal effort and minimal disruption to the business.

What is the best migration strategy to meet these requirements?

- A) Use the AWS Schema Conversion Tool (AWS SCT) to copy the database schema from RDS for PostgreSQL to an Aurora PostgreSQL DB cluster. Create an AWS DMS task to copy the data.
- B) Create a script to continuously back up the RDS for PostgreSQL instance using pg_dump, and restore the backup to an Aurora PostgreSQL DB cluster using pg_restore.
- C) Create a read replica from the existing production RDS for PostgreSQL instance. Check that the replication lag is zero and then promote the read replica as a standalone Aurora PostgreSQL DB cluster.
- D) Create an Aurora Replica from the existing production RDS for PostgreSQL instance. Stop the writes on the master; check that the replication lag is zero, and then promote the Aurora Replica as a standalone Aurora PostgreSQL DB cluster.

A media company is running a critical production application that uses Amazon RDS for PostgreSQL with Multi-AZ deployments. The database size is currently 25 TB. The IT director wants to migrate the database to Amazon Aurora PostgreSQL with minimal effort and minimal disruption to the business.

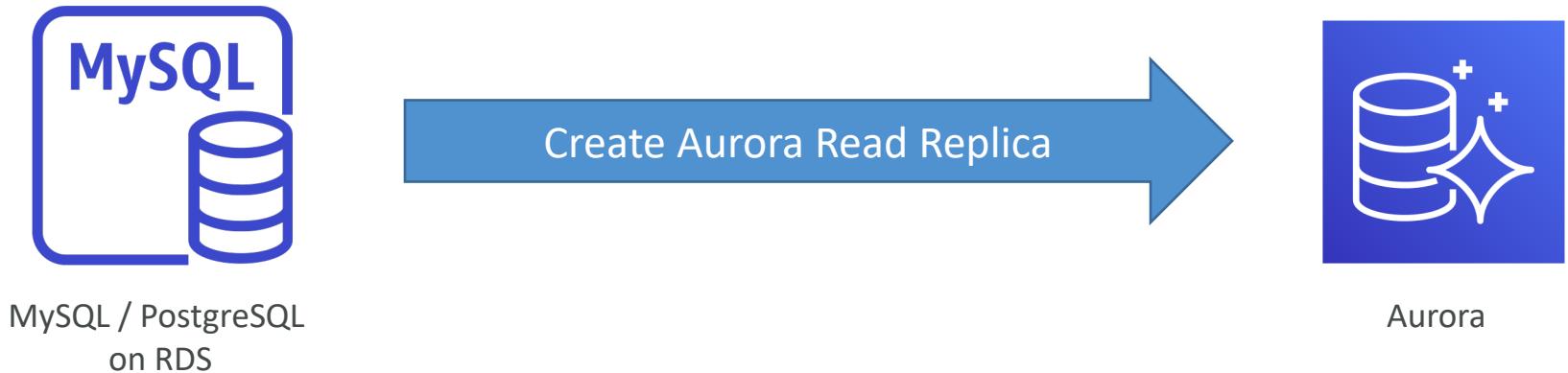
What is the best migration strategy to meet these requirements?

- A) Use the AWS Schema Conversion Tool (AWS SCT) to copy the database schema from RDS for PostgreSQL to an Aurora PostgreSQL DB cluster. Create an AWS DMS task to copy the data.
- B) Create a script to continuously back up the RDS for PostgreSQL instance using pg_dump, and restore the backup to an Aurora PostgreSQL DB cluster using pg_restore.
- C) Create a read replica from the existing production RDS for PostgreSQL instance. Check that the replication lag is zero and then promote the read replica as a standalone Aurora PostgreSQL DB cluster.
- D) Create an Aurora Replica from the existing production RDS for PostgreSQL instance. Stop the writes on the master; check that the replication lag is zero, and then promote the Aurora Replica as a standalone Aurora PostgreSQL DB cluster.

Cluster Replication Options for Aurora

Replication b/w RDS DB instance and an Aurora DB cluster

- By creating an Aurora read replica of RDS DB instance
- Typically used for migration to Aurora rather than ongoing replication
 - To migrate, stop the writes on master. After replication lag is zero, promote the Aurora replica as a standalone Aurora DB cluster



A media company is running a critical production application that uses Amazon RDS for PostgreSQL with Multi-AZ deployments. The database size is currently 25 TB. The IT director wants to migrate the database to Amazon Aurora PostgreSQL with minimal effort and minimal disruption to the business.

What is the best migration strategy to meet these requirements?

- A) Use the AWS Schema Conversion Tool (AWS SCT) to copy the database schema from RDS for PostgreSQL to an Aurora PostgreSQL DB cluster. Create an AWS DMS task to copy the data.
- B) Create a script to continuously back up the RDS for PostgreSQL instance using pg_dump, and restore the backup to an Aurora PostgreSQL DB cluster using pg_restore.
- C) Create a read replica from the existing production RDS for PostgreSQL instance. Check that the replication lag is zero and then promote the read replica as a standalone Aurora PostgreSQL DB cluster.
- D) Create an Aurora Replica from the existing production RDS for PostgreSQL instance. Stop the writes on the master; check that the replication lag is zero, and then promote the Aurora Replica as a standalone Aurora PostgreSQL DB cluster.

A media company is running a critical production application that uses Amazon RDS for PostgreSQL with Multi-AZ deployments. The database size is currently 25 TB. The IT director wants to migrate the database to Amazon Aurora PostgreSQL with minimal effort and minimal disruption to the business.

What is the best migration strategy to meet these requirements?

- A) Use the AWS Schema Conversion Tool (AWS SCT) to copy the database schema from RDS for PostgreSQL to an Aurora PostgreSQL DB cluster. Create an AWS DMS task to copy the data.
- B) Create a script to continuously back up the RDS for PostgreSQL instance using pg_dump, and restore the backup to an Aurora PostgreSQL DB cluster using pg_restore.
- C) Create a read replica from the existing production RDS for PostgreSQL instance. Check that the replication lag is zero and then promote the read replica as a standalone Aurora PostgreSQL DB cluster.
- D) Create an Aurora Replica from the existing production RDS for PostgreSQL instance. Stop the writes on the master; check that the replication lag is zero, and then promote the Aurora Replica as a standalone Aurora PostgreSQL DB cluster.

Question 2

A medical company is planning to migrate its on-premises PostgreSQL database, along with application and web servers, to AWS. Amazon RDS for PostgreSQL is being considered as the target database engine. Access to the database should be limited to application servers and a bastion host in a VPC.

Which solution meets the security requirements?

- A) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Modify the pg_hba.conf file on the DB instance to allow connections from only the application servers and bastion host.
- B) Launch the RDS for PostgreSQL database in a DB subnet group containing public subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- C) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- D) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a NACL attached to the VPC and private subnets. Modify the inbound and outbound rules to allow connections to and from the application servers and bastion host.

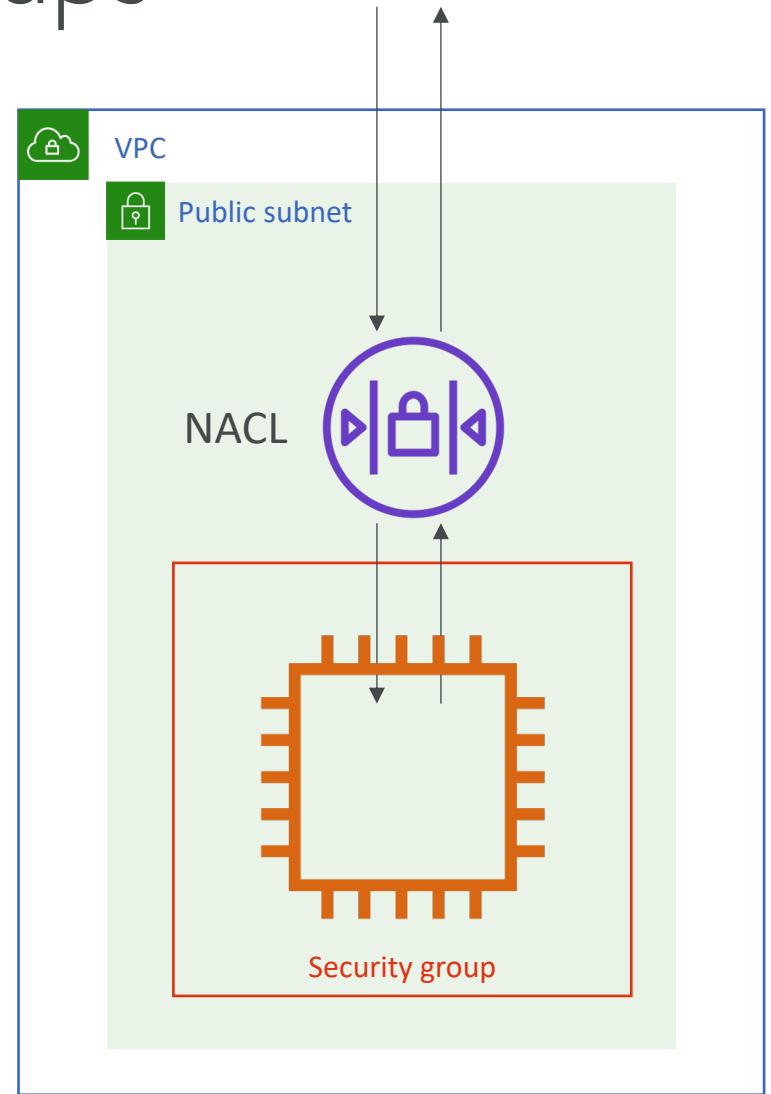
A medical company is planning to migrate its on-premises PostgreSQL database, along with application and web servers, to AWS. Amazon RDS for PostgreSQL is being considered as the target database engine. Access to the database should be limited to application servers and a bastion host in a VPC.

Which solution meets the security requirements?

- A) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Modify the pg_hba.conf file on the DB instance to allow connections from only the application servers and bastion host.
- B) Launch the RDS for PostgreSQL database in a DB subnet group containing public subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- C) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- D) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a NACL attached to the VPC and private subnets. Modify the inbound and outbound rules to allow connections to and from the application servers and bastion host.

Network ACL & Security Groups

- NACL (Network ACL)
 - A firewall which controls traffic from and to subnet
 - Can have ALLOW and DENY rules
 - Are attached at the **Subnet** level
 - Rules only include IP addresses
- Security Groups
 - A firewall that controls traffic to and from **an ENI / an EC2 Instance**
 - Can have only ALLOW rules
 - Rules include IP addresses and other security groups



A medical company is planning to migrate its on-premises PostgreSQL database, along with application and web servers, to AWS. Amazon RDS for PostgreSQL is being considered as the target database engine. Access to the database should be limited to application servers and a bastion host in a VPC.

Which solution meets the security requirements?

- A) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Modify the pg_hba.conf file on the DB instance to allow connections from only the application servers and bastion host.
- B) Launch the RDS for PostgreSQL database in a DB subnet group containing public subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- C) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- D) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a NACL attached to the VPC and private subnets. Modify the inbound and outbound rules to allow connections to and from the application servers and bastion host.

A medical company is planning to migrate its on-premises PostgreSQL database, along with application and web servers, to AWS. Amazon RDS for PostgreSQL is being considered as the target database engine. Access to the database should be limited to application servers and a bastion host in a VPC.

Which solution meets the security requirements?

- A) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Modify the pg_hba.conf file on the DB instance to allow connections from only the application servers and bastion host.
- B) Launch the RDS for PostgreSQL database in a DB subnet group containing public subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- C) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a new security group with inbound rules to allow connections from only the security groups of the application servers and bastion host. Attach the new security group to the DB instance.
- D) Launch the RDS for PostgreSQL database in a DB subnet group containing private subnets. Create a NACL attached to the VPC and private subnets. Modify the inbound and outbound rules to allow connections to and from the application servers and bastion host.

Question 3

A database specialist is troubleshooting complaints from an application's users who are experiencing performance issues when saving data in an Amazon ElastiCache for Redis cluster with cluster mode disabled. The database specialist finds that the performance issues are occurring during the cluster's backup window. The cluster runs in a replication group containing three nodes. Memory on the nodes is fully utilized. Organizational policies prohibit the database specialist from changing the backup window time.

How could the database specialist address the performance concern?
(Select TWO)

- A) Add an additional node to the cluster in the same Availability Zone as the primary.
- B) Configure the backup job to take a snapshot of a read replica.
- C) Increase the local instance storage size for the cluster nodes.
- D) Increase the reserved-memory-percent parameter value.
- E) Configure the backup process to flush the cache before taking the backup.

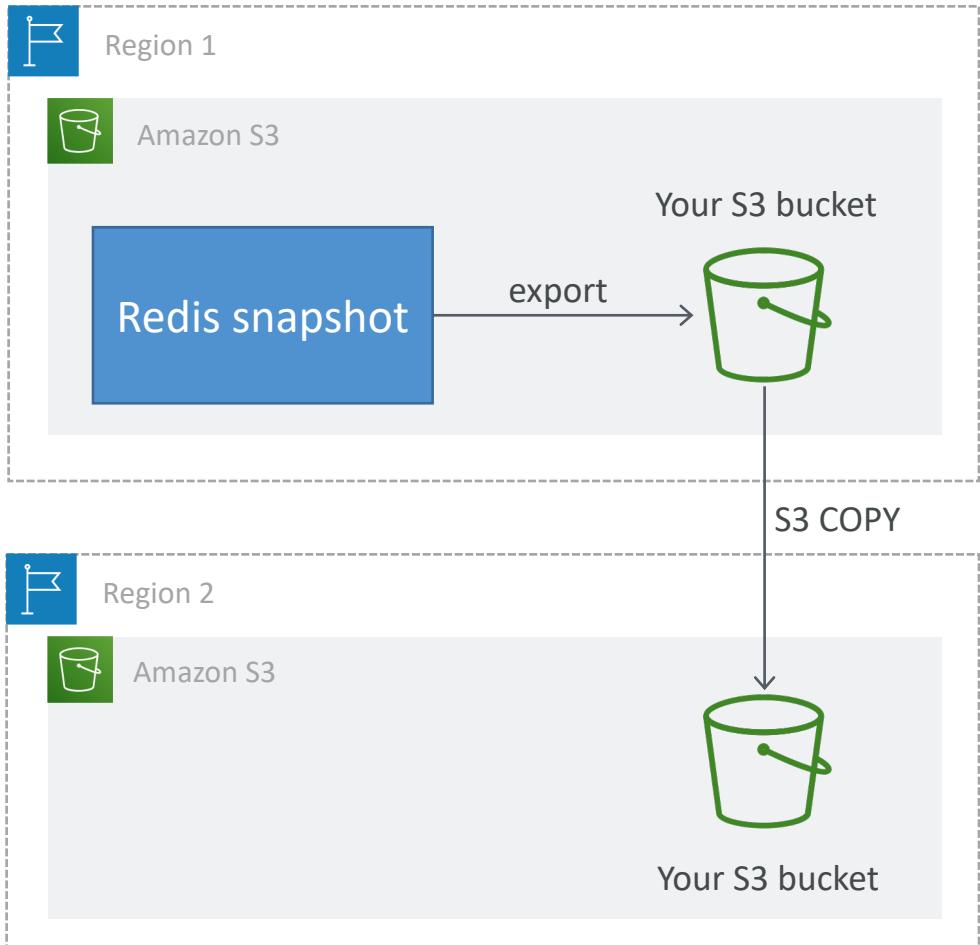
A database specialist is troubleshooting complaints from an application's users who are experiencing **performance issues** when saving data in an Amazon **ElastiCache for Redis** cluster with **cluster mode disabled**. The database specialist finds that the performance issues are **occurring during the cluster's backup window**. The cluster runs in a replication group containing three nodes. **Memory on the nodes is fully utilized**. Organizational policies prohibit the database specialist from changing the backup window time.

How could the database specialist address the performance concern?
(Select TWO)

- A) Add an additional node to the cluster in the same Availability Zone as the primary.
- B) Configure the backup job to take a snapshot of a read replica.
- C) Increase the local instance storage size for the cluster nodes.
- D) Increase the reserved-memory-percent parameter value.
- E) Configure the backup process to flush the cache before taking the backup.

Redis Backup and Restore

- Supports manual and automatic backups
- Backups are point-in-time copy of the entire Redis cluster; can't backup individual nodes
- Can be used to warm start a new cluster (=preloaded data)
- Can backup from primary node or from replica
- Recommended to backup from a replica (ensures primary node performance)
- Backups (also called snapshots) are stored in S3
- Can export snapshots to your S3 buckets in the same region
- Can then copy the exported snapshot to other region / account using S3 API



Redis best practices

- Cluster mode – connect using the configuration endpoint (allows for auto-discovery of shard and keyspace (slot) mappings)
- Cluster mode disabled – use primary endpoint for writes and reader endpoint for reads (always kept up to date with any cluster changes)
- Set the parameter reserved-memory-percent=25% (for background processes, non-data)
- Keep socket timeout = 1 second (at least)
 - Too low => numerous timeouts on high load
 - Too high => application might take longer to detect connection issues
- Keep DNS caching timeout low (TTL = 5–10 seconds recommended)
- Do not use the “cache forever” option for DNS caching



ElastiCache for
Redis

A database specialist is troubleshooting complaints from an application's users who are experiencing **performance issues** when saving data in an Amazon **ElastiCache for Redis** cluster with **cluster mode disabled**. The database specialist finds that the performance issues are **occurring during the cluster's backup window**. The cluster runs in a replication group containing three nodes. **Memory on the nodes is fully utilized**. Organizational policies prohibit the database specialist from changing the backup window time.

How could the database specialist address the performance concern?
(Select TWO)

- A) Add an additional node to the cluster in the same Availability Zone as the primary.
- B) Configure the backup job to take a snapshot of a read replica.
- C) Increase the local instance storage size for the cluster nodes.
- D) Increase the reserved-memory-percent parameter value.
- E) Configure the backup process to flush the cache before taking the backup.

A database specialist is troubleshooting complaints from an application's users who are experiencing **performance issues** when saving data in an Amazon **ElastiCache for Redis** cluster with **cluster mode disabled**. The database specialist finds that the performance issues are **occurring during the cluster's backup window**. The cluster runs in a replication group containing three nodes. **Memory on the nodes is fully utilized**. Organizational policies prohibit the database specialist from changing the backup window time.

How could the database specialist address the performance concern?
(Select TWO)

- A) Add an additional node to the cluster in the same Availability Zone as the primary.
- B) Configure the backup job to take a snapshot of a read replica.
- C) Increase the local instance storage size for the cluster nodes.
- D) Increase the reserved-memory-percent parameter value.
- E) Configure the backup process to flush the cache before taking the backup.

Question 4

A company's security department has mandated that their existing Amazon RDS for MySQL DB instance be encrypted at rest.

What should a database specialist do to meet this requirement?

- A) Modify the database to enable encryption. Apply this setting immediately without waiting for the next scheduled maintenance window.
- B) Export the database to an Amazon S3 bucket with encryption enabled. Create a new database and import the export file.
- C) Create a snapshot of the database. Create an encrypted copy of the snapshot. Create a new database from the encrypted snapshot.
- D) Create a snapshot of the database. Restore the snapshot into a new database with encryption enabled.

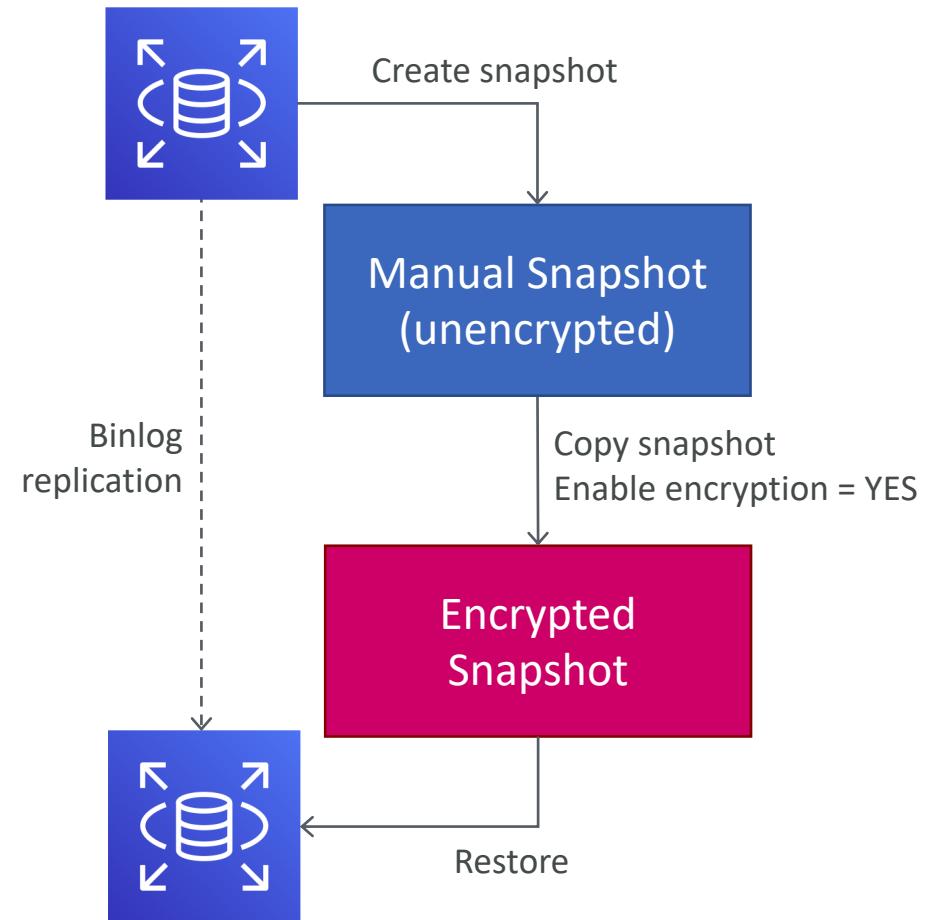
A company's security department has mandated that their existing Amazon RDS for MySQL DB instance be encrypted at rest.

What should a database specialist do to meet this requirement?

- A) Modify the database to enable encryption. Apply this setting immediately without waiting for the next scheduled maintenance window.
- B) Export the database to an Amazon S3 bucket with encryption enabled. Create a new database and import the export file.
- C) Create a snapshot of the database. Create an encrypted copy of the snapshot. Create a new database from the encrypted snapshot.
- D) Create a snapshot of the database. Restore the snapshot into a new database with encryption enabled.

How to encrypt an unencrypted RDS DB

- Can't encrypt an existing unencrypted RDS DB instance
- Can't create an encrypted read replica from an unencrypted instance
- Copy an unencrypted snapshot with encryption enabled
- Restore the encrypted snapshot to a new RDS DB instance
- Can use MySQL replication to synchronize changes (binlog replication)
- Sidenote – if it's an Aurora unencrypted snapshot, then you can directly restore it to an encrypted aurora DB by specifying the KMS key. No need to copy the snapshot.



A company's security department has mandated that their existing Amazon RDS for MySQL DB instance be encrypted at rest.

What should a database specialist do to meet this requirement?

- A) Modify the database to enable encryption. Apply this setting immediately without waiting for the next scheduled maintenance window.
- B) Export the database to an Amazon S3 bucket with encryption enabled. Create a new database and import the export file.
- C) Create a snapshot of the database. Create an encrypted copy of the snapshot. Create a new database from the encrypted snapshot.
- D) Create a snapshot of the database. Restore the snapshot into a new database with encryption enabled.

A company's security department has mandated that their existing Amazon RDS for MySQL DB instance be encrypted at rest.

What should a database specialist do to meet this requirement?

- A) Modify the database to enable encryption. Apply this setting immediately without waiting for the next scheduled maintenance window.
- B) Export the database to an Amazon S3 bucket with encryption enabled. Create a new database and import the export file.
- C) Create a snapshot of the database. Create an encrypted copy of the snapshot. Create a new database from the encrypted snapshot.
- D) Create a snapshot of the database. Restore the snapshot into a new database with encryption enabled.

Question 5

A company has a highly available production 10TB SQL Server relational database running on Amazon EC2. Users have recently been reporting performance and connectivity issues. A database specialist has been asked to configure a monitoring and alerting strategy that will provide metrics visibility and notifications to troubleshoot these issues.

Which solution will meet these requirements?

- A) Configure AWS CloudTrail logs to monitor and detect signs of potential problems. Create an AWS Lambda function that is triggered when specific API calls are made and send notifications to an Amazon SNS topic.
- B) Install an Amazon Inspector agent on the DB instance. Configure the agent to stream server and database activity to Amazon CloudWatch Logs. Configure metric filters and alarms to send notifications to an Amazon SNS topic.
- C) Migrate the database to Amazon RDS for SQL Server and use Performance Insights to monitor and detect signs of potential problems. Create a scheduled AWS Lambda function that retrieves metrics from the Performance Insights API and send notifications to an Amazon SNS topic.
- D) Configure Amazon CloudWatch Application Insights for .NET and SQL Server to monitor and detect signs of potential problems. Configure CloudWatch Events to send notifications to an Amazon SNS topic.

A company has a highly available production 10TB **SQL Server** relational database running on **Amazon EC2**. Users have recently been reporting **performance and connectivity issues**. A database specialist has been asked to configure a **monitoring and alerting strategy** that will provide metrics visibility and notifications to troubleshoot these issues.

Which solution will meet these requirements?

- A) Configure AWS CloudTrail logs to monitor and detect signs of potential problems. Create an AWS Lambda function that is triggered when specific API calls are made and send notifications to an Amazon SNS topic.
- B) Install an Amazon Inspector agent on the DB instance. Configure the agent to stream server and database activity to Amazon CloudWatch Logs. Configure metric filters and alarms to send notifications to an Amazon SNS topic.
- C) Migrate the database to Amazon RDS for SQL Server and use Performance Insights to monitor and detect signs of potential problems. Create a scheduled AWS Lambda function that retrieves metrics from the Performance Insights API and send notifications to an Amazon SNS topic.
- D) Configure Amazon CloudWatch Application Insights for .NET and SQL Server to monitor and detect signs of potential problems. Configure CloudWatch Events to send notifications to an Amazon SNS topic.

CloudWatch Application Insights

- For .NET and SQL Server
- Also supports DynamoDB tables
- Identifies and sets up key metrics, logs, and alarms for SQL Server workloads
- Uses CloudWatch events and alarms
- Useful for problem detection, notification and troubleshooting



CloudWatch

A company has a highly available production 10TB **SQL Server** relational database running on **Amazon EC2**. Users have recently been reporting **performance and connectivity issues**. A database specialist has been asked to configure a **monitoring and alerting strategy** that will provide metrics visibility and notifications to troubleshoot these issues.

Which solution will meet these requirements?

- A) Configure AWS CloudTrail logs to monitor and detect signs of potential problems. Create an AWS Lambda function that is triggered when specific API calls are made and send notifications to an Amazon SNS topic.
- B) Install an Amazon Inspector agent on the DB instance. Configure the agent to stream server and database activity to Amazon CloudWatch Logs. Configure metric filters and alarms to send notifications to an Amazon SNS topic.
- C) Migrate the database to Amazon RDS for SQL Server and use Performance Insights to monitor and detect signs of potential problems. Create a scheduled AWS Lambda function that retrieves metrics from the Performance Insights API and send notifications to an Amazon SNS topic.
- D) Configure Amazon CloudWatch Application Insights for .NET and SQL Server to monitor and detect signs of potential problems. Configure CloudWatch Events to send notifications to an Amazon SNS topic.

A company has a highly available production 10TB **SQL Server** relational database running on **Amazon EC2**. Users have recently been reporting **performance and connectivity issues**. A database specialist has been asked to configure a **monitoring and alerting strategy** that will provide metrics visibility and notifications to troubleshoot these issues.

Which solution will meet these requirements?

- A) Configure AWS CloudTrail logs to monitor and detect signs of potential problems. Create an AWS Lambda function that is triggered when specific API calls are made and send notifications to an Amazon SNS topic.
- B) Install an Amazon Inspector agent on the DB instance. Configure the agent to stream server and database activity to Amazon CloudWatch Logs. Configure metric filters and alarms to send notifications to an Amazon SNS topic.
- C) Migrate the database to Amazon RDS for SQL Server and use Performance Insights to monitor and detect signs of potential problems. Create a scheduled AWS Lambda function that retrieves metrics from the Performance Insights API and send notifications to an Amazon SNS topic.
- D) Configure Amazon CloudWatch Application Insights for .NET and SQL Server to monitor and detect signs of potential problems. Configure CloudWatch Events to send notifications to an Amazon SNS topic.

Question 6

A company's ecommerce application stores order transactions in an Amazon RDS for MySQL database. The database has run out of available storage and the application is currently unable to take orders.

Which action should a database specialist take to resolve the issue in the shortest amount of time?

- A) Add more storage space to the DB instance using the ModifyDBInstance action.
- B) Create a new DB instance with more storage space from the latest backup.
- C) Change the DB instance status from STORAGE_FULL to AVAILABLE.
- D) Configure a read replica with more storage space.

A company's ecommerce application stores order transactions in an Amazon RDS for MySQL database. The database has run out of available storage and the application is currently unable to take orders.

Which action should a database specialist take to resolve the issue in the shortest amount of time?

- A) Add more storage space to the DB instance using the ModifyDBInstance action.
- B) Create a new DB instance with more storage space from the latest backup.
- C) Change the DB instance status from STORAGE_FULL to AVAILABLE.
- D) Configure a read replica with more storage space.

Choosing Storage Type

- General Purpose Storage (=cost-effective SSD storage)
 - You choose the storage size
 - You get a baseline of 3 IOPS/GB
 - Volumes below 1 TiB can burst to 3,000 IOPS (uses I/O credits)
 - Use with variable workloads
 - Typically used for small to medium sized DBs and in DEV/TEST environments
- Provisioned IOPS (=high-performance storage, recommended for production)
 - You choose storage size and required IOPS
 - Fast and predictable performance
 - Up to 32,000 IOPS max per DB instance
 - Use when consistent high IOPS are required (I/O-intensive workloads)
 - Well-suited for write-heavy workloads
- If instance runs out of storage, it may no longer be available until you allocate more storage (=> use storage autoscaling)



Amazon RDS

A company's ecommerce application stores order transactions in an Amazon RDS for MySQL database. The database has run out of available storage and the application is currently unable to take orders.

Which action should a database specialist take to resolve the issue in the shortest amount of time?

- A) Add more storage space to the DB instance using the ModifyDBInstance action.
- B) Create a new DB instance with more storage space from the latest backup.
- C) Change the DB instance status from STORAGE_FULL to AVAILABLE.
- D) Configure a read replica with more storage space.

A company's ecommerce application stores order transactions in an Amazon RDS for MySQL database. The database has run out of available storage and the application is currently unable to take orders.

Which action should a database specialist take to resolve the issue in the shortest amount of time?

- A) Add more storage space to the DB instance using the ModifyDBInstance action.
- B) Create a new DB instance with more storage space from the latest backup.
- C) Change the DB instance status from STORAGE_FULL to AVAILABLE.
- D) Configure a read replica with more storage space.

Question 7

A company undergoing a security audit has determined that its database administrators are presently sharing an administrative database user account for the company's Amazon Aurora deployment. To support proper traceability, governance, and compliance, each database administration team member must start using individual, named accounts. Furthermore, long-term database user credentials should not be used.

Which solution should a database specialist implement to meet these requirements?

- A) Use the AWS CLI to fetch the AWS IAM users and passwords for all team members. For each IAM user, create an Aurora user with the same password as the IAM user.
- B) Enable IAM database authentication on the Aurora cluster. Create a database user for each team member without a password. Attach an IAM policy to each administrator's IAM user account that grants the connect privilege using their database user account.
- C) Create a database user for each team member. Share the new database user credentials with the team members. Have users change the password on the first login to the same password as their IAM user.
- D) Create an IAM role and associate an IAM policy that grants the connect privilege using the shared account. Configure a trust policy that allows the administrator's IAM user account to assume the role.

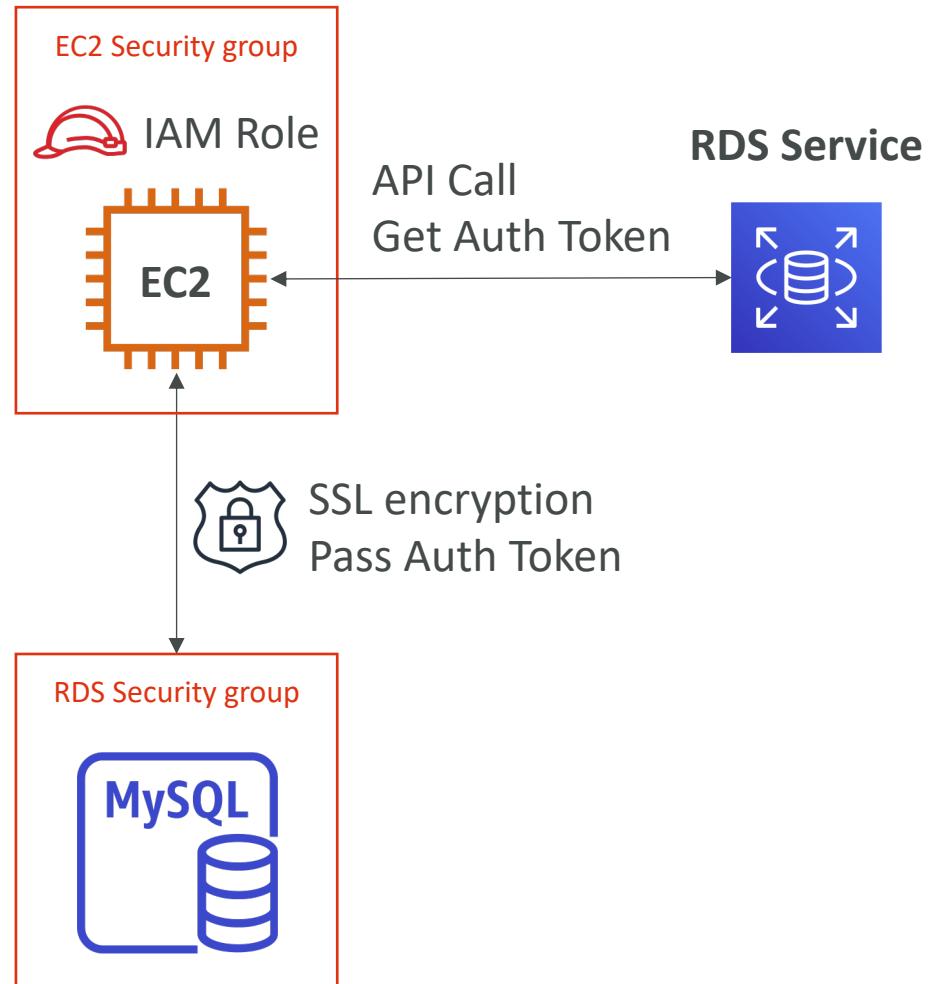
A company undergoing a security audit has determined that its database administrators are presently **sharing an administrative database user account** for the company's Amazon Aurora deployment. To support proper traceability, governance, and compliance, each database administration team member must start using **individual, named accounts**. Furthermore, **long-term database user credentials should not be used**.

Which solution should a database specialist implement to meet these requirements?

- A) Use the AWS CLI to fetch the AWS IAM users and passwords for all team members. For each IAM user, create an Aurora user with the same password as the IAM user.
- B) Enable IAM database authentication on the Aurora cluster. Create a database user for each team member without a password. Attach an IAM policy to each administrator's IAM user account that grants the connect privilege using their database user account.
- C) Create a database user for each team member. Share the new database user credentials with the team members. Have users change the password on the first login to the same password as their IAM user.
- D) Create an IAM role and associate an IAM policy that grants the connect privilege using the shared account. Configure a trust policy that allows the administrator's IAM user account to assume the role.

RDS - IAM Authentication (IAM DB Auth)

- IAM database authentication works with MySQL and PostgreSQL
- You don't need a password, just an authentication token obtained through IAM & RDS API calls
- Auth token has a lifetime of 15 minutes
- Benefits:
 - Network in/out must be encrypted using SSL
 - IAM to centrally manage users instead of DB
 - Can leverage IAM Roles and EC2 Instance profiles for easy integration



Using IAM DB Auth

- Enable IAM DB authentication on the DB cluster
- Create a DB user (without a password)
- Attach an IAM policy to map the DB user to the IAM role
- Attach the IAM role to an IAM user (or to EC2 instance)
- Now you can connect to the DB using IAM token over SSL

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "rds-db:connect"  
      ],  
      "Resource": [  
        "arn:aws:rds-db:us-east-2:1234567890:dbuser:db-ABCDEFGHIJKLM01234/db_user"  
      ]  
    }  
  ]  
}
```

Using IAM DB Auth (MySQL)

- Enable IAM DB authentication on the DB cluster
- Create a DB user (without a password)
- Attach an IAM policy to map the DB user to the IAM role
- Attach the IAM role to an IAM user (or to EC2 instance)
- Now you can connect to the DB using IAM token over SSL
- Note: use native GRANT / REVOKE for DB specific privileges

```
CREATE USER {db_username} IDENTIFIED WITH
AWSAuthenticationPlugin as 'RDS';

GRANT USAGE ON *.* TO '{db_username}'@'%' REQUIRE
SSL;

wget https://s3.amazonaws.com/rds-downloads/rds-
ca-2019-root.pem

TOKEN=$(aws rds generate-db-auth-token
--hostname {db_or_cluster_endpoint}
--port 3306 --username {db_username})"

mysql --host={db_or_cluster_endpoint} --port=3306
--ssl-ca=/home/ec2-user/rds-combined-ca-
bundle.pem --enable-cleartext-plugin
--user={db_username} --password=$TOKEN
```

Using IAM DB Auth (PostgreSQL)

- Enable IAM DB authentication on the DB cluster
- Create a DB user (without a password)
- Attach an IAM policy to map the DB user to the IAM role
- Attach the IAM role to an IAM user (or to EC2 instance)
- Now you can connect to the DB using IAM token over SSL
- Note: use native GRANT / REVOKE for DB specific privileges



You don't need to modify pg_hba.conf file for DB access. With RDS, you use DB grants.

```
CREATE USER {db_username};  
  
GRANT rds_iam to {db_username};  
  
wget https://s3.amazonaws.com/rds-downloads/rds-ca-2019-root.pem  
  
export PGPASSWORD=$(aws rds generate-db-auth-token --hostname={db_endpoint} --port=5432 --username={db_username} --region us-west-2)"  
  
psql -h {db_endpoint} -p 5432 "dbname={db_name} user={db_username} password=$PGPASSWORD sslrootcert=/home/ec2-user/rds-combined-ca-bundle.pem sslmode=verify-ca"
```

A company undergoing a security audit has determined that its database administrators are presently **sharing an administrative database user account** for the company's Amazon Aurora deployment. To support proper traceability, governance, and compliance, each database administration team member must start using **individual, named accounts**. Furthermore, **long-term database user credentials should not be used**.

Which solution should a database specialist implement to meet these requirements?

- A) Use the AWS CLI to fetch the AWS IAM users and passwords for all team members. For each IAM user, create an Aurora user with the same password as the IAM user.
- B) Enable IAM database authentication on the Aurora cluster. Create a database user for each team member without a password. Attach an IAM policy to each administrator's IAM user account that grants the connect privilege using their database user account.
- C) Create a database user for each team member. Share the new database user credentials with the team members. Have users change the password on the first login to the same password as their IAM user.
- D) Create an IAM role and associate an IAM policy that grants the connect privilege using the shared account. Configure a trust policy that allows the administrator's IAM user account to assume the role.

A company undergoing a security audit has determined that its database administrators are presently **sharing an administrative database user account** for the company's Amazon Aurora deployment. To support proper traceability, governance, and compliance, each database administration team member must start using **individual, named accounts**. Furthermore, **long-term database user credentials should not be used**.

Which solution should a database specialist implement to meet these requirements?

- A) Use the AWS CLI to fetch the AWS IAM users and passwords for all team members. For each IAM user, create an Aurora user with the same password as the IAM user.
- B) Enable IAM database authentication on the Aurora cluster. Create a database user for each team member without a password. Attach an IAM policy to each administrator's IAM user account that grants the connect privilege using their database user account.
- C) Create a database user for each team member. Share the new database user credentials with the team members. Have users change the password on the first login to the same password as their IAM user.
- D) Create an IAM role and associate an IAM policy that grants the connect privilege using the shared account. Configure a trust policy that allows the administrator's IAM user account to assume the role.

Question 8

A global company wants to run an application in several AWS Regions to support a global user base. The application will need a database that can support a high volume of low-latency reads and writes that is expected to vary over time. The data must be shared across all of the Regions to support dynamic company-wide reports.

Which database meets these requirements?

- A) Use Amazon Aurora Serverless and configure endpoints in each Region.
- B) Use Amazon RDS for MySQL and deploy read replicas in an auto scaling group in each Region.
- C) Use Amazon DocumentDB (with MongoDB compatibility) and configure read replicas in an auto scaling group in each Region.
- D) Use Amazon DynamoDB global tables and configure DynamoDB auto scaling for the tables.

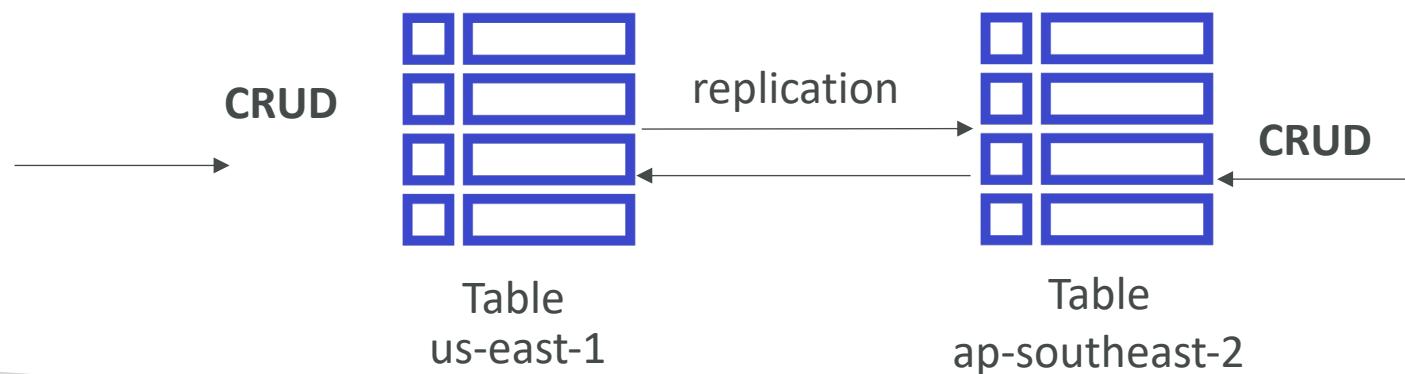
A global company wants to run an application in **several AWS Regions** to support a global user base. The application will need a database that can support a **high volume of low-latency reads and writes** that is expected to vary over time. The data must be shared across all of the Regions to support dynamic company-wide reports.

Which database meets these requirements?

- A) Use Amazon Aurora Serverless and configure endpoints in each Region.
- B) Use Amazon RDS for MySQL and deploy read replicas in an auto scaling group in each Region.
- C) Use Amazon DocumentDB (with MongoDB compatibility) and configure read replicas in an auto scaling group in each Region.
- D) Use Amazon DynamoDB global tables and configure DynamoDB auto scaling for the tables.

DynamoDB Global Tables

- Automatic, **Multi-Master, Active-Active, Cross-region** replication
- Useful for **low latency**, DR purposes
- Near real-time replication (< 1 second replication lag)
- Eventual consistency for cross-region reads
- Strong consistency for same region reads
- “Last Writer Wins” approach for conflict resolution
- Transactions are ACID-compliant only in the region where write occurs originally



A global company wants to run an application in **several AWS Regions** to support a global user base. The application will need a database that can support a **high volume of low-latency reads and writes** that is expected to vary over time. The data must be shared across all of the Regions to support dynamic company-wide reports.

Which database meets these requirements?

- A) Use Amazon Aurora Serverless and configure endpoints in each Region.
- B) Use Amazon RDS for MySQL and deploy read replicas in an auto scaling group in each Region.
- C) Use Amazon DocumentDB (with MongoDB compatibility) and configure read replicas in an auto scaling group in each Region.
- D) Use Amazon DynamoDB global tables and configure DynamoDB auto scaling for the tables.

A global company wants to run an application in **several AWS Regions** to support a global user base. The application will need a database that can support a **high volume of low-latency reads and writes** that is expected to vary over time. The data must be shared across all of the Regions to support dynamic company-wide reports.

Which database meets these requirements?

- A) Use Amazon Aurora Serverless and configure endpoints in each Region.
- B) Use Amazon RDS for MySQL and deploy read replicas in an auto scaling group in each Region.
- C) Use Amazon DocumentDB (with MongoDB compatibility) and configure read replicas in an auto scaling group in each Region.
- D) Use Amazon DynamoDB global tables and configure DynamoDB auto scaling for the tables.

Question 9

A company's customer relationship management application uses an Amazon RDS for PostgreSQL Multi-AZ database. The database size is approximately 100 GB. A database specialist has been tasked with developing a cost-effective disaster recovery plan that will restore the database in a different Region within 2 hours. The restored database should not be missing more than 8 hours of transactions.

What is the MOST cost-effective solution that meets the availability requirements?

- A) Create an RDS read replica in the second Region. For disaster recovery, promote the read replica to a standalone instance.
- B) Create an RDS read replica in the second Region using a smaller instance size. For disaster recovery, scale the read replica and promote it to a standalone instance.
- C) Schedule an AWS Lambda function to create an hourly snapshot of the DB instance and another Lambda function to copy the snapshot to the second Region. For disaster recovery, create a new RDS Multi-AZ DB instance from the last snapshot.
- D) Create a new RDS Multi-AZ DB instance in the second Region. Configure an AWS DMS task for ongoing replication.

A company's customer relationship management application uses an Amazon RDS for PostgreSQL Multi-AZ database. The database size is approximately 100 GB. A database specialist has been tasked with developing a cost-effective disaster recovery plan that will restore the database in a different Region within 2 hours. The restored database should not be missing more than 8 hours of transactions.

What is the MOST cost-effective solution that meets the availability requirements?

- A) Create an RDS read replica in the second Region. For disaster recovery, promote the read replica to a standalone instance.
- B) Create an RDS read replica in the second Region using a smaller instance size. For disaster recovery, scale the read replica and promote it to a standalone instance.
- C) Schedule an AWS Lambda function to create an hourly snapshot of the DB instance and another Lambda function to copy the snapshot to the second Region. For disaster recovery, create a new RDS Multi-AZ DB instance from the last snapshot.
- D) Create a new RDS Multi-AZ DB instance in the second Region. Configure an AWS DMS task for ongoing replication.

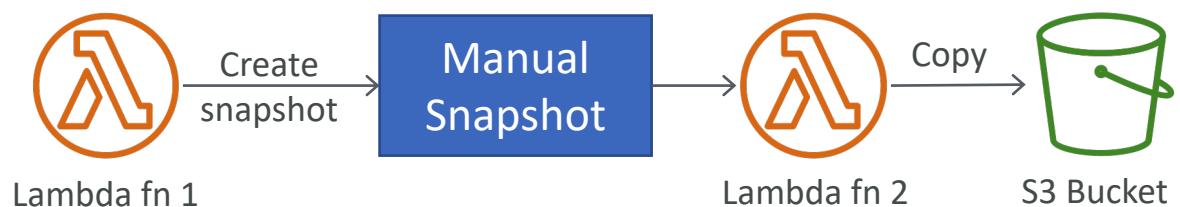
Backups vs Snapshots

Backups

- Are automated
- Are incremental
- Retention period up to 35 days
- **Support PITR** within retention period
- Great for unexpected failures
- A non-zero backup retention period in RDS also enables a snapshot before and after DB engine upgrades

Snapshots

- Are manually triggered
- Are full backups
- Retained as long as you want
- Does not support PITR
- Great for known events like DB upgrades etc.
- **Can use Lambda functions to take periodic backups and move them to S3 (say for compliance purposes)**



A company's customer relationship management application uses an Amazon RDS for PostgreSQL Multi-AZ database. The database size is approximately 100 GB. A database specialist has been tasked with developing a cost-effective disaster recovery plan that will restore the database in a different Region within 2 hours. The restored database should not be missing more than 8 hours of transactions.

What is the MOST cost-effective solution that meets the availability requirements?

- A) Create an RDS read replica in the second Region. For disaster recovery, promote the read replica to a standalone instance.
- B) Create an RDS read replica in the second Region using a smaller instance size. For disaster recovery, scale the read replica and promote it to a standalone instance.
- C) Schedule an AWS Lambda function to create an hourly snapshot of the DB instance and another Lambda function to copy the snapshot to the second Region. For disaster recovery, create a new RDS Multi-AZ DB instance from the last snapshot.
- D) Create a new RDS Multi-AZ DB instance in the second Region. Configure an AWS DMS task for ongoing replication.

A company's customer relationship management application uses an Amazon RDS for PostgreSQL Multi-AZ database. The database size is approximately 100 GB. A database specialist has been tasked with developing a cost-effective disaster recovery plan that will restore the database in a different Region within 2 hours. The restored database should not be missing more than 8 hours of transactions.

What is the MOST cost-effective solution that meets the availability requirements?

- A) Create an RDS read replica in the second Region. For disaster recovery, promote the read replica to a standalone instance.
- B) Create an RDS read replica in the second Region using a smaller instance size. For disaster recovery, scale the read replica and promote it to a standalone instance.
- C) Schedule an AWS Lambda function to create an hourly snapshot of the DB instance and another Lambda function to copy the snapshot to the second Region. For disaster recovery, create a new RDS Multi-AZ DB instance from the last snapshot.
- D) Create a new RDS Multi-AZ DB instance in the second Region. Configure an AWS DMS task for ongoing replication.

Question 10

An operations team in a large company wants to centrally manage resource provisioning for its development teams across multiple accounts. When a new AWS account is created, the developers require full privileges for a database environment that uses the same configuration, data schema, and source data as the company's production Amazon RDS for MySQL DB instance.

How can the operations team achieve this?

- A) Enable the source DB instance to be shared with the new account so the development team may take a snapshot. Create an AWS CloudFormation template to launch the new DB instance from the snapshot.
- B) Create an AWS CLI script to launch the approved DB instance configuration in the new account. Create an AWS DMS task to copy the data from the source DB instance to the new DB instance.
- C) Take a manual snapshot of the source DB instance and share the snapshot privately with the new account. Specify the snapshot ARN in an RDS resource in an AWS CloudFormation template and use StackSets to deploy to the new account.
- D) Create a DB instance read replica of the source DB instance. Share the read replica with the new AWS account.

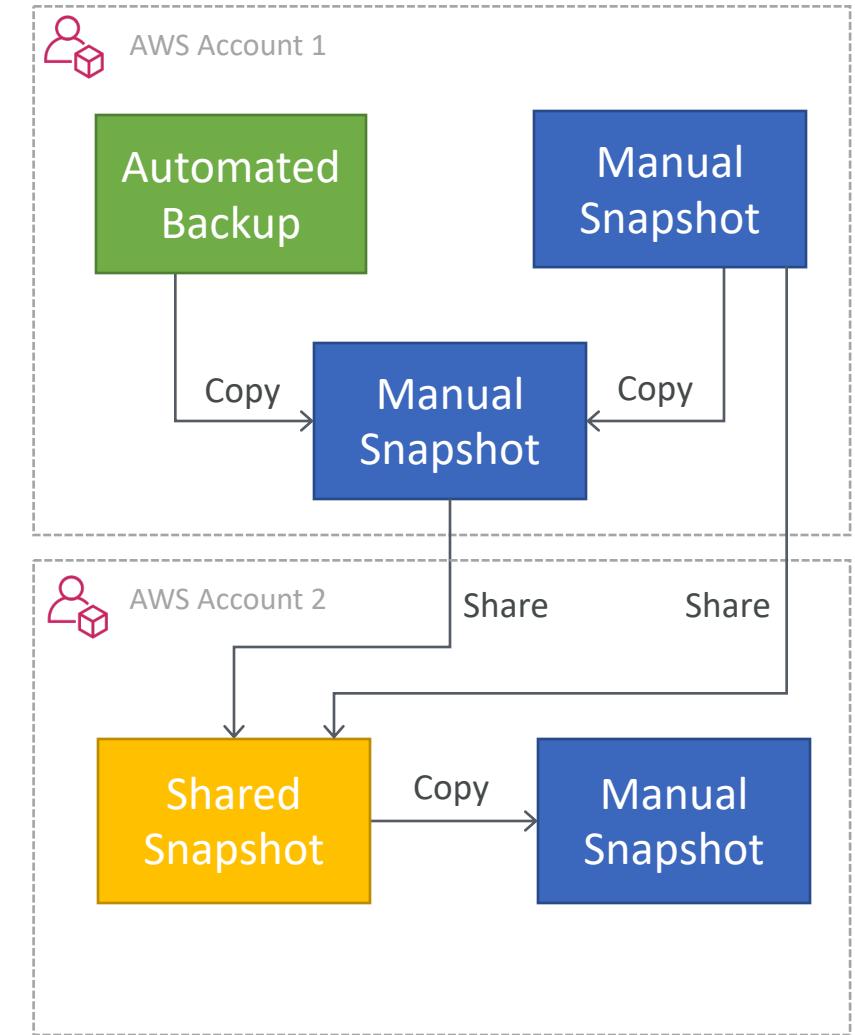
An operations team in a large company wants to **centrally manage resource provisioning** for its development teams **across multiple accounts**. When a new AWS account is created, the developers require full privileges for a database environment that uses the same configuration, data schema, and source data as the company's production Amazon **RDS for MySQL DB** instance.

How can the operations team achieve this?

- A) Enable the source DB instance to be shared with the new account so the development team may take a snapshot. Create an AWS CloudFormation template to launch the new DB instance from the snapshot.
- B) Create an AWS CLI script to launch the approved DB instance configuration in the new account. Create an AWS DMS task to copy the data from the source DB instance to the new DB instance.
- C) Take a manual snapshot of the source DB instance and share the snapshot privately with the new account. Specify the snapshot ARN in an RDS resource in an AWS CloudFormation template and use StackSets to deploy to the new account.
- D) Create a DB instance read replica of the source DB instance. Share the read replica with the new AWS account.

Copying and sharing RDS snapshots

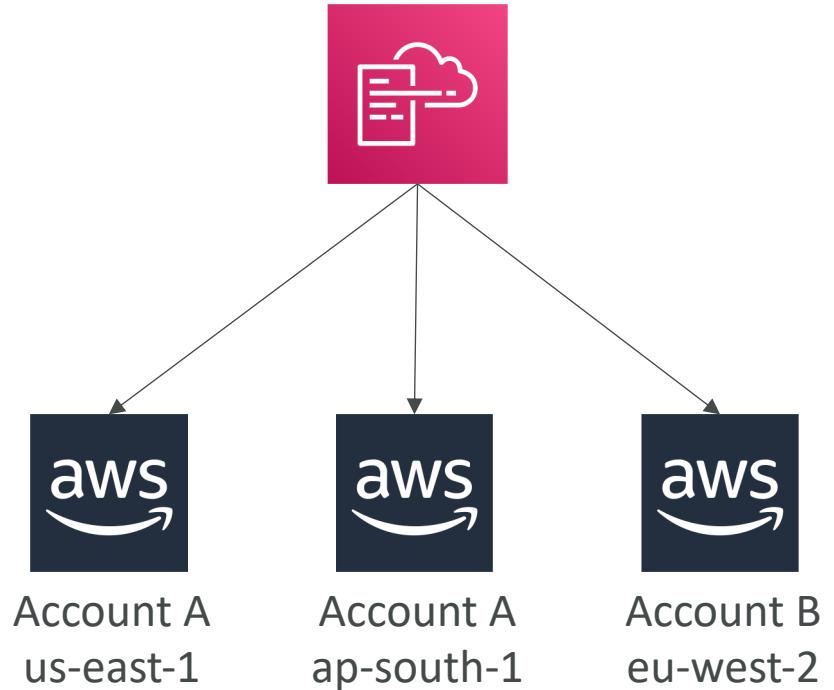
- You can copy automated backups or manual snapshots
- The copy becomes a manual snapshot
- You can copy snapshots within region, across regions or across accounts
- For copying snapshot across account, you must share the snapshot first, and then copy it in the target account
- Automated backups cannot be shared directly. Must snapshot first.
- Copying across regions/accounts = data transfer costs



CloudFormation - StackSets

- Create, update, or delete stacks across **multiple accounts and regions** with a single operation
- Administrator account to create StackSets
- Trusted accounts to create, update, delete stack instances from StackSets
- When you update a stack set, *all* associated stack instances are updated throughout all accounts and regions.

CloudFormation **StackSet**
Admin Account



An operations team in a large company wants to **centrally manage resource provisioning** for its development teams **across multiple accounts**. When a new AWS account is created, the developers require full privileges for a database environment that uses the same configuration, data schema, and source data as the company's production Amazon **RDS for MySQL DB** instance.

How can the operations team achieve this?

- A) Enable the source DB instance to be shared with the new account so the development team may take a snapshot. Create an AWS CloudFormation template to launch the new DB instance from the snapshot.
- B) Create an AWS CLI script to launch the approved DB instance configuration in the new account. Create an AWS DMS task to copy the data from the source DB instance to the new DB instance.
- C) Take a manual snapshot of the source DB instance and share the snapshot privately with the new account. Specify the snapshot ARN in an RDS resource in an AWS CloudFormation template and use StackSets to deploy to the new account.
- D) Create a DB instance read replica of the source DB instance. Share the read replica with the new AWS account.

An operations team in a large company wants to **centrally manage resource provisioning** for its development teams **across multiple accounts**. When a new AWS account is created, the developers require full privileges for a database environment that uses the same configuration, data schema, and source data as the company's production Amazon **RDS for MySQL DB** instance.

How can the operations team achieve this?

- A) Enable the source DB instance to be shared with the new account so the development team may take a snapshot. Create an AWS CloudFormation template to launch the new DB instance from the snapshot.
- B) Create an AWS CLI script to launch the approved DB instance configuration in the new account. Create an AWS DMS task to copy the data from the source DB instance to the new DB instance.
- C) Take a manual snapshot of the source DB instance and share the snapshot privately with the new account. Specify the snapshot ARN in an RDS resource in an AWS CloudFormation template and use StackSets to deploy to the new account.
- D) Create a DB instance read replica of the source DB instance. Share the read replica with the new AWS account.

Exam Tips

Cues you don't wanna miss!

Extra Resources

- <https://www.youtube.com/watch?v=hwnNbLXN4vA>
- Exam readiness course (free): <https://www.aws.training/Details/eLearning?id=47245>
- Reference architectures:
 - <https://github.com/aws-samples/aws-dbs-refarch-graph>
 - <https://github.com/aws-samples/aws-dbs-refarch-rdbms>
 - <https://github.com/aws-samples/aws-dbs-refarch-datalake>
 - <https://github.com/aws-samples/aws-dbs-refarch-edw>

Extra Resources

- Read service FAQS
 - FAQ = Frequently asked questions
 - Example: <https://aws.amazon.com/rds/faqs/>
 - FAQ cover a lot of the questions asked at the exam
- Read service troubleshooting
 - Example:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Troubleshooting.html

Congratulations & Next Steps

And then some more...

Next steps

- Congratulations, you have covered all the domains!
- Make sure you revisit the lectures as much as possible
- A good extra resource to do is the AWS Exam Readiness course at:
 - <https://www.aws.training/Details/eLearning?id=47245>
- The AWS Certified Database Specialty exam is hard, and tests experience...
- Make sure you master every single concept outlined in this course

Thank You!

Muchas gracias | Merci beaucoup | Danke Schon | දැන්යවාද