Database
RDS, Aurora, Elasticache

AWS Certified Solutions Architect– Professional (SAP-C01)
Study notes - Sep’2019

RDS

RDS Multi-AZ Deployments and Read Replicas

https://www.percona.com/blog/2018/08/02/amazon-rds-multi-az-deployments-read-replicas/
https://aws.amazon.com/about-aws/whats-new/2018/01/amazon-rds-read-replicas-now-support-multi-az-deployments/

Multi-AZ deploment

RDS provides **HA and failover support** for DB instances using Multi-AZ deployments. It automatically provisions and maintains a **synchronous standby replica** of the master DB in a different AZ to provide data redundancy, failover support and to minimize latency during system backups.

Benefits of Multi-AZ deployment:

- Replication to a standby replica is synchronous
- AUTO FAILOVER in case:
    i. The primary DB instance fails
    ii. An Availability Zone outage
    iii. The DB instance server type is changed
    iv. OS patching.
    v. Manual failover initiated
- The endpoint of the DB instance remains the same after a failover, the application can resume database operations without manual intervention.
- If a failure occurs, your availability impact is limited to the time that the automatic failover takes to complete. This helps to achieve increased availability.
- It reduces the impact of maintenance.

Can I use an RDS standby replica for read scaling?

The Multi-AZ deployments are not a read scaling solution, you cannot use a standby replica to serve read traffic. Multi-AZ maintains a standby replica for HA/failover. It is available for use only when RDS promotes the standby instance as the primary. To service read-only traffic, you should use a **Read Replica** instead.

Read Replicas

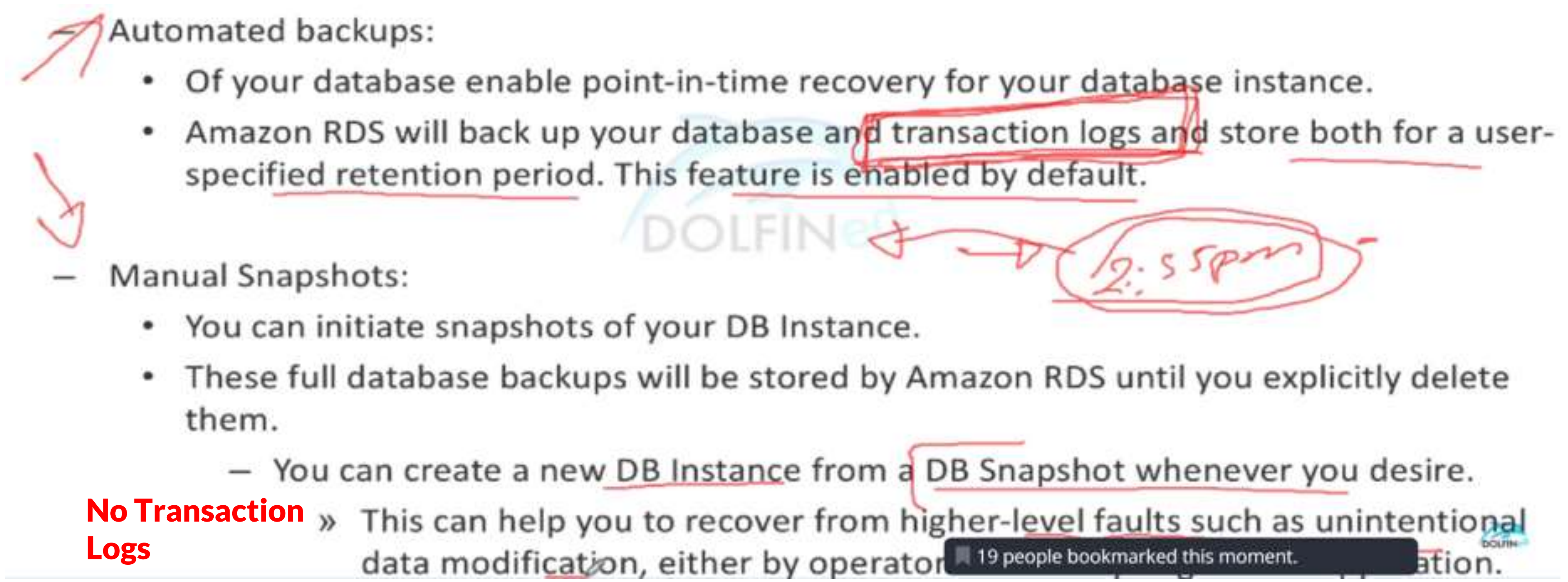
Read replicas allow you to have a **read-only copy** of your DB. When you create a Read Replica, you first specify an existing DB instance as the source. Then RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot. You can use MySQL native asynchronous replication to keep Read Replica up-to-date with the changes. The source DB must have automatic backups enabled for setting up read replica.

Benefits of Read Replicas:

- Read Replica helps in decreasing load on the primary DB by serving read-only traffic.
- A Read Replica can be manually promoted as a standalone DB instance.
- You can create Read Replicas within AZ, Cross-AZ or Cross-Region.
- You can have up to five Read Replicas per master, each with own DNS endpoint. Unlike a Multi-AZ standby replica, you can connect to each Read Replica and use them for read scaling.
- You can have Read Replicas of Read Replicas.
- Read Replicas can be Multi-AZ enabled.
- You can use Read Replicas to take logical backups (mysqldump/mydumper) if you want to store the backups externally.
- Read Replica helps to maintain a copy of databases in a different region for disaster recovery.

Conclusion
While both (Multi-AZ and Read replica) maintain a copy of database but they are different in nature. Use Multi-AZ deployments for High Availability and Read Replica for read scalability. You can further set up a cross-region read replica for disaster recovery.

Automated vs Manual backups



– Automated backups:

- Of your database enable point-in-time recovery for your database instance.
- Amazon RDS will back up your database and transaction logs and store both for a user-specified retention period. This feature is enabled by default.

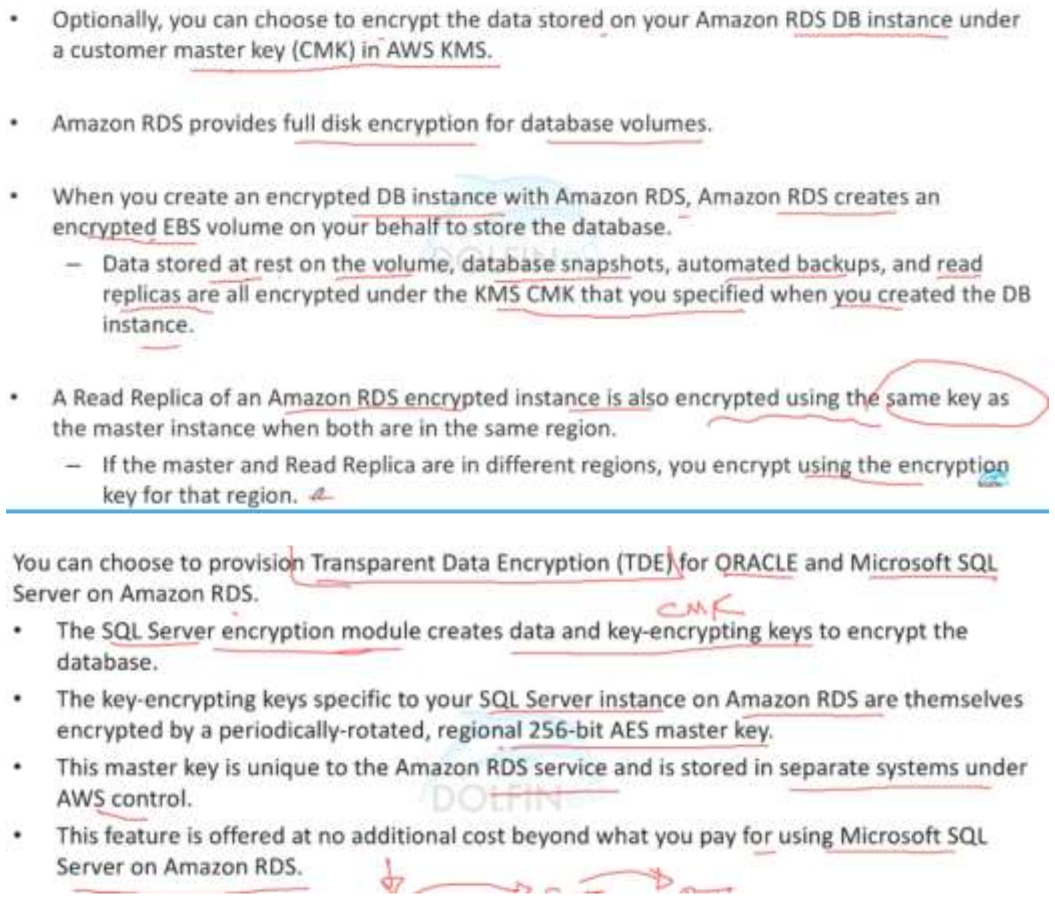
2:55pm

– Manual Snapshots:

- You can initiate snapshots of your DB Instance.
- These full database backups will be stored by Amazon RDS until you explicitly delete them.
  – You can create a new DB Instance from a DB Snapshot whenever you desire.
    » This can help you to recover from higher-level faults such as unintentional data modification, either by operator 19 people bookmarked this moment. ation.

No Transaction Logs

RDS Encryption



- Optionally, you can choose to encrypt the data stored on your Amazon RDS DB instance under a customer master key (CMK) in AWS KMS.
- Amazon RDS provides full disk encryption for database volumes.
- When you create an encrypted DB instance with Amazon RDS, Amazon RDS creates an encrypted EBS volume on your behalf to store the database.
  - Data stored at rest on the volume, database snapshots, automated backups, and read replicas are all encrypted under the KMS CMK that you specified when you created the DB instance.
- A Read Replica of an Amazon RDS encrypted instance is also encrypted using the same key as the master instance when both are in the same region.
  - If the master and Read Replica are in different regions, you encrypt using the encryption key for that region.

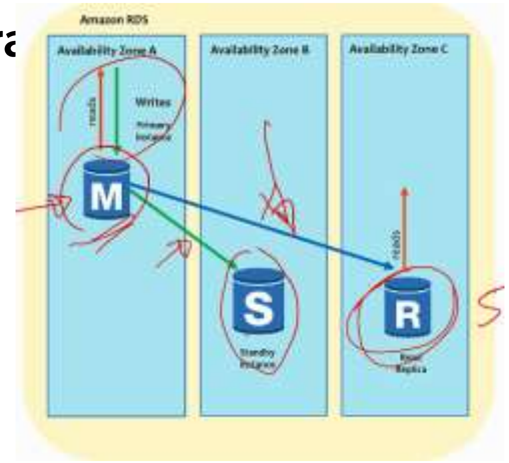
You can choose to provision Transparent Data Encryption (TDE) for ORACLE and Microsoft SQL Server on Amazon RDS.

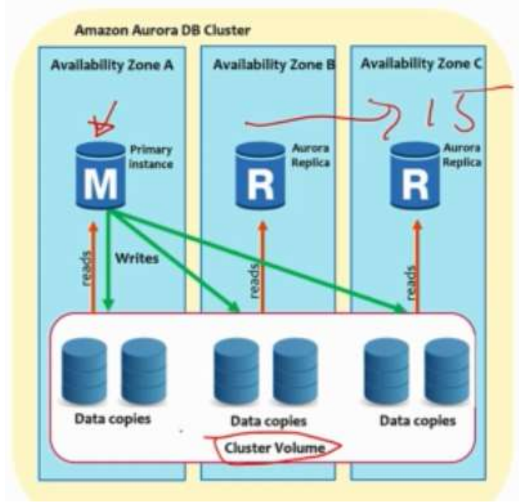
CMK

- The SQL Server encryption module creates data and key-encrypting keys to encrypt the database.
- The key-encrypting keys specific to your SQL Server instance on Amazon RDS are themselves encrypted by a periodically-rotated, regional 256-bit AES master key.
- This master key is unique to the Amazon RDS service and is stored in separate systems under AWS control.
- This feature is offered at no additional cost beyond what you pay for using Microsoft SQL Server on Amazon RDS.

Difference between RDS and Aurora

	RDS	Aurora
Cluster Architecture	Compute & Volume per instance	Compute per instance, Volume cluster across 3 AZs
Service Type	Regional	Regional
HA	Multi-AZ and/or Read replicas (upto 5) RR Auto promoted on failure	only Read replicas (upto 15) RR Auto promoted on failure
RR sync	ASynchronous	Synchronous
Advantages		Cost effective, higher throughput, simple to set up, operate and scale, supports MySQL & PostgresQL

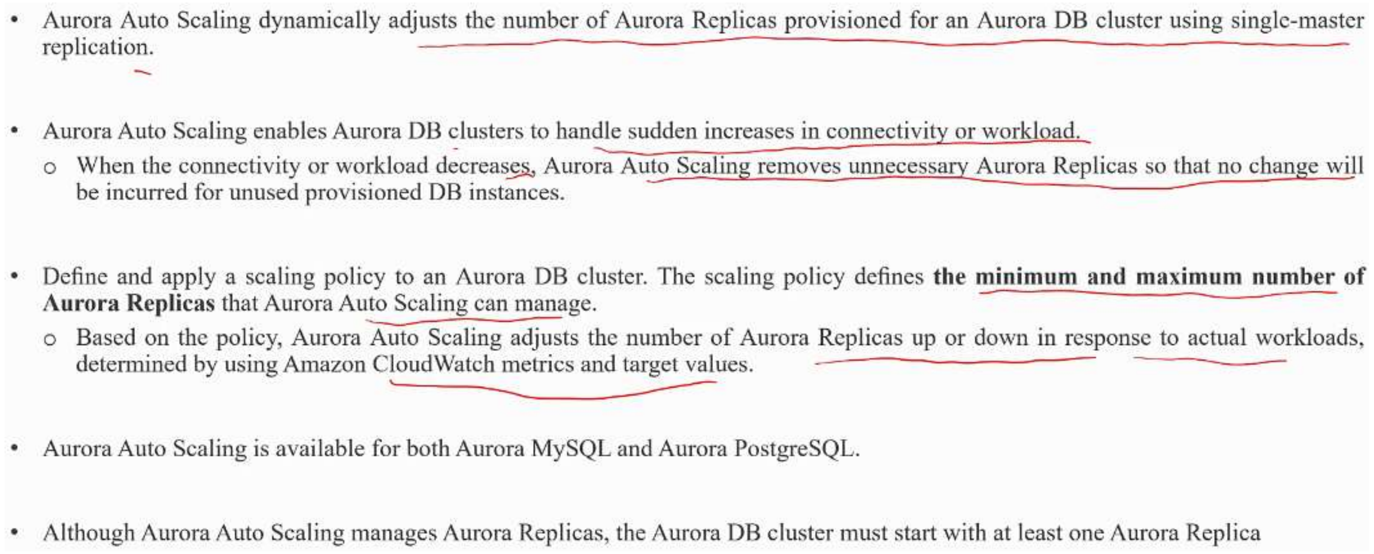




Aurora Connection endpoints

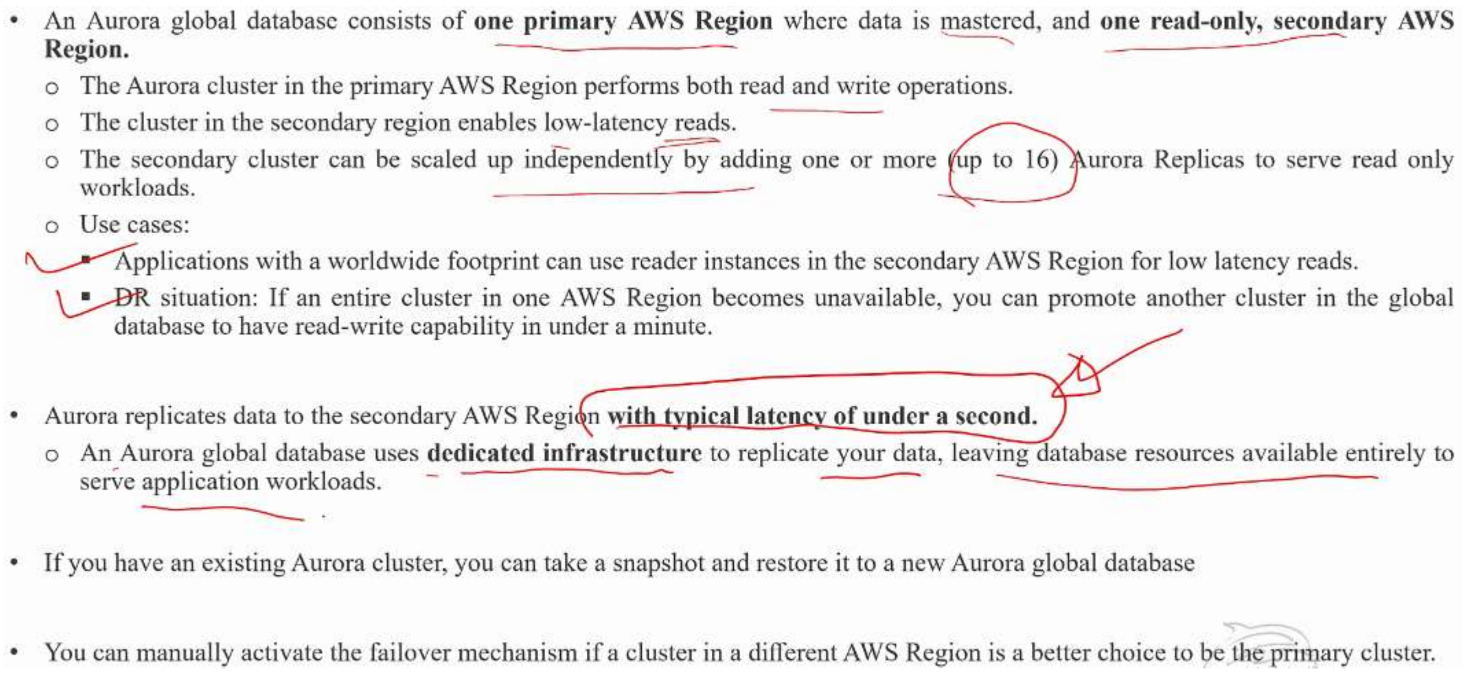
Cluster endpoint
A *cluster endpoint* for an Aurora DB cluster that connects to the current primary DB instance for that DB cluster. ONLY write operations. Each Aurora DB cluster has one cluster endpoint and one primary DB instance.
Reader endpoint
Connects to one of the available Aurora Replicas for that DB cluster. Each Aurora DB cluster has one reader endpoint. If there is more than one Aurora Replica, the reader endpoint directs each connection request to one of the Aurora Replicas.
Custom endpoint
Represents a set of DB instances that you choose. When you connect to the endpoint, Aurora performs load balancing and chooses one of the instances in the group to handle the connection. You define which instances this endpoint refers to, and you decide what purpose the endpoint serves.
Instance endpoint
An instance endpoint connects to a specific DB instance within an Aurora cluster. Each DB instance in a DB cluster has its own unique instance endpoint. So there is one instance endpoint for the current primary DB instance of the DB cluster, and there is one instance endpoint for each of the Aurora Replicas in the DB cluster.

Aurora Autoscaling



- Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas provisioned for an Aurora DB cluster using single-master replication.
- Aurora Auto Scaling enables Aurora DB clusters to handle sudden increases in connectivity or workload.
    - When the connectivity or workload decreases, Aurora Auto Scaling removes unnecessary Aurora Replicas so that no change will be incurred for unused provisioned DB instances.
- Define and apply a scaling policy to an Aurora DB cluster. The scaling policy defines **the minimum and maximum number of Aurora Replicas** that Aurora Auto Scaling can manage.
    - Based on the policy, Aurora Auto Scaling adjusts the number of Aurora Replicas up or down in response to actual workloads, determined by using Amazon CloudWatch metrics and target values.
- Aurora Auto Scaling is available for both Aurora MySQL and Aurora PostgreSQL.
- Although Aurora Auto Scaling manages Aurora Replicas, the Aurora DB cluster must start with at least one Aurora Replica

Aurora Global Database



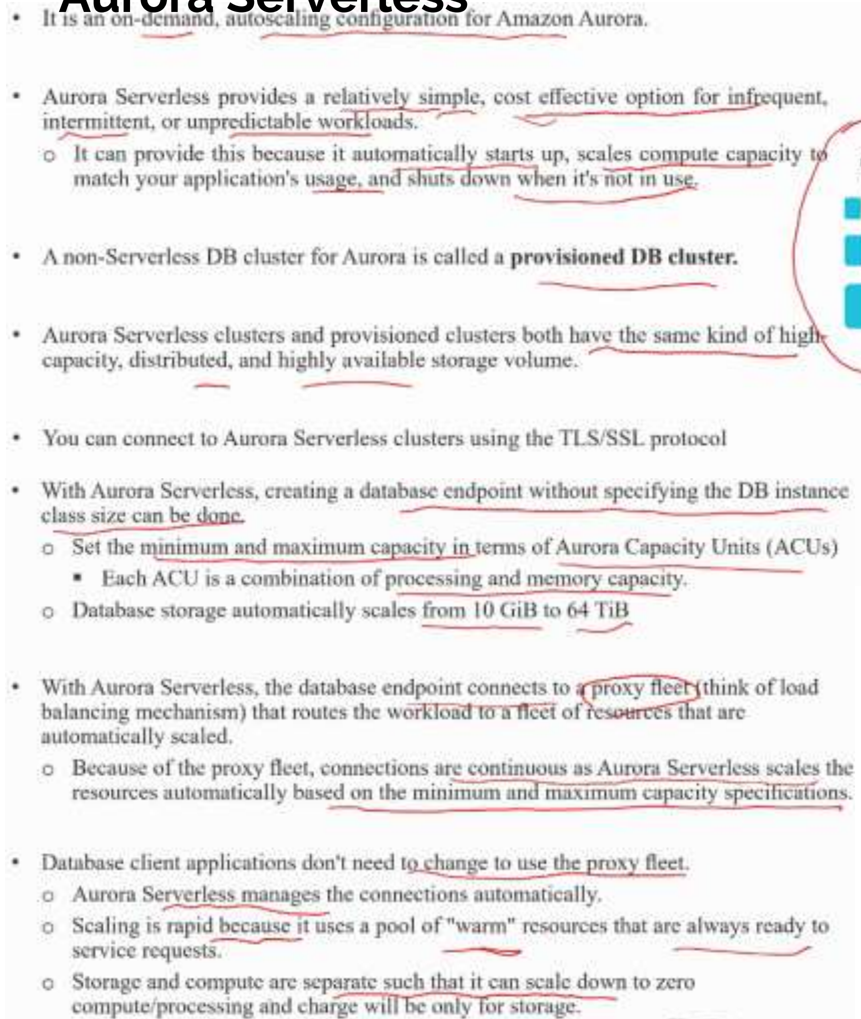
- An Aurora global database consists of **one primary AWS Region** where data is mastered, and **one read-only, secondary AWS Region.**
  - The Aurora cluster in the primary AWS Region performs both read and write operations.
  - The cluster in the secondary region enables low-latency reads.
  - The secondary cluster can be scaled up independently by adding one or more (up to 16) Aurora Replicas to serve read only workloads.
  - Use cases:
    - Applications with a worldwide footprint can use reader instances in the secondary AWS Region for low latency reads.
    - DR situation: If an entire cluster in one AWS Region becomes unavailable, you can promote another cluster in the global database to have read-write capability in under a minute.

- Aurora replicates data to the secondary AWS Region **with typical latency of under a second.**
  - An Aurora global database uses **dedicated infrastructure** to replicate your data, leaving database resources available entirely to serve application workloads.

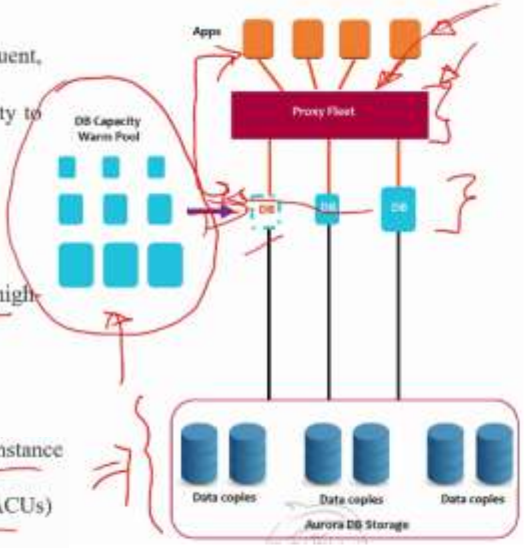
- If you have an existing Aurora cluster, you can take a snapshot and restore it to a new Aurora global database

- You can manually activate the failover mechanism if a cluster in a different AWS Region is a better choice to be the primary cluster.

Aurora Serverless



- It is an on-demand, autoscaling configuration for Amazon Aurora.
- Aurora Serverless provides a relatively simple, cost effective option for infrequent, intermittent, or unpredictable workloads.
  - It can provide this because it automatically starts up, scales compute capacity to match your application's usage, and shuts down when it's not in use.
- A non-Serverless DB cluster for Aurora is called a **provisioned DB cluster.**
- Aurora Serverless clusters and provisioned clusters both have the same kind of high-capacity, distributed, and highly available storage volume.
- You can connect to Aurora Serverless clusters using the TLS/SSL protocol
- With Aurora Serverless, creating a database endpoint without specifying the DB instance class size can be done.
  - Set the minimum and maximum capacity in terms of Aurora Capacity Units (ACUs)
    - Each ACU is a combination of processing and memory capacity.
  - Database storage automatically scales from 10 GiB to 64 TiB
- With Aurora Serverless, the database endpoint connects to a proxy fleet (think of load balancing mechanism) that routes the workload to a fleet of resources that are automatically scaled.
  - Because of the proxy fleet, connections are continuous as Aurora Serverless scales the resources automatically based on the minimum and maximum capacity specifications.
- Database client applications don't need to change to use the proxy fleet.
  - Aurora Serverless manages the connections automatically.
  - Scaling is rapid because it uses a pool of "warm" resources that are always ready to service requests.
  - Storage and compute are separate such that it can scale down to zero compute/processing and charge will be only for storage.



Serverless use cases:
Infrequently used / New applications
Variable workload
Unpredictable workload

Difference b/w RDS MySQL and Aurora MySQL

TL;DR

- If you are looking for a native HA solution then you should use Aurora
- For a read-intensive workload within an HA environment, Aurora is a perfect match.
- To use MySQL plugins you should use RDS MySQL
- Aurora only supports InnoDB. If you need other engines i.e. MyISAM, RDS MySQL is the only option
- With RDS MySQL you can use specific MySQL releases
- AUR has more efficient thread pooling that allows using 5000 concurrent connections, no need of using connection pooling
- AUR supports Online & transactional DDL which has much better performance
- AUR has better query cache which used to stall MYSQL

Replication:

With Aurora, you can provision up to fifteen replicas compared to just five in RDS MySQL. All Aurora replicas share the same underlying volume with the primary instance and this means that replication can be performed in milliseconds as updates made by the primary instance are instantly available to all Aurora replicas.

HA:

Adding a reader automatically makes Aurora multi-AZ Failover is automatic with no data loss on Amazon Aurora whereas the replicas failover priority can be set. *RDS failover within region (Multi-AZ) is also automatic.*

No need of Capacity planning for Aurora as storage increases automatically from 10G to 64TB

DynamoDB

DynamoDB

Fully managed, Unlike RDS, DynamoDB is not launched in your VPC
Provides fully managed Auto scaling & Auto healing
Partition is selected by running a hash function on the specified partition key
Supports Eventual (default) & Strong Read consistency, Application can specify strong
Indexes ALSO has a partition and sort key. Always prefer GSI against LCI unless u need strong consistency

Table -> Items (aka rows) -> Attributes (aka columns). DynamoDB uses ***primary keys to uniquely identify each item*** in a table and ***secondary indexes to provide more querying flexibility***. Items are schemaless except for the primary key

Primary Key

1. Simple primary key (partition key) - unique identifier
2. ***Composite primary key*** (partition & sort key) - partition key + an attribute to be stored in sorted order

Secondary Indexes also has a partition & sort key. lets you query the data in the table using an alternate key, in addition to queries against the primary key. Has 2 types -

1. ***Local secondary index*** - partition key MUST BE SAME as the partition key specified for the table, sort key can be any attribute
2. Global secondary index - partition key & sort key CAN BE ANY 2 ATTRIBUTES
    a. They can be added on to existing tables (local indexes needs to be created at Create time only!)
    b. They have their own provisioned throughput
    c. ONLY Supports Eventual Read consistency

DynamoDB Streams is an optional feature that captures data modification events in DynamoDB tables. The data about these events appear in the stream in near real time, and in the order that the events occurred. Events - Item Added, Updated, Deleted

RCU and WRU calculation for Provisioned mode ***-> with*** a provisioned table with 6 RCUs and 6 WRUs, your application could do the following:

★ Perform strongly consistent reads of up to 24 KB per second (4 KB × 6 read capacity units).
★ Perform eventually consistent reads of up to 48 KB per second (twice as much read throughput).
★ Perform transactional read requests of up to 3 KB per second.
★ Write up to 6 KB per second (1 KB × 6 write capacity units).
★ Perform transactional write requests of up to 3 KB per second.

Best practices:
In RDBMS, NORMALIZATION is important, whereas in NoSQL, search and access patterns are most important. Use SORT order. Distribute queries. Use GSI. Use Global Tables for Cross-region data replication., Use partition key that can ensure large # of distinct values compared to # of rows

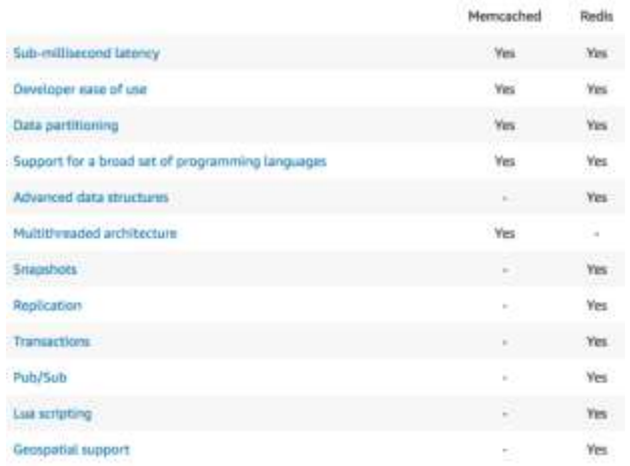
Using Amazon Aurora Auto Scaling with Aurora Replicas

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Integrating.AutoScaling.html

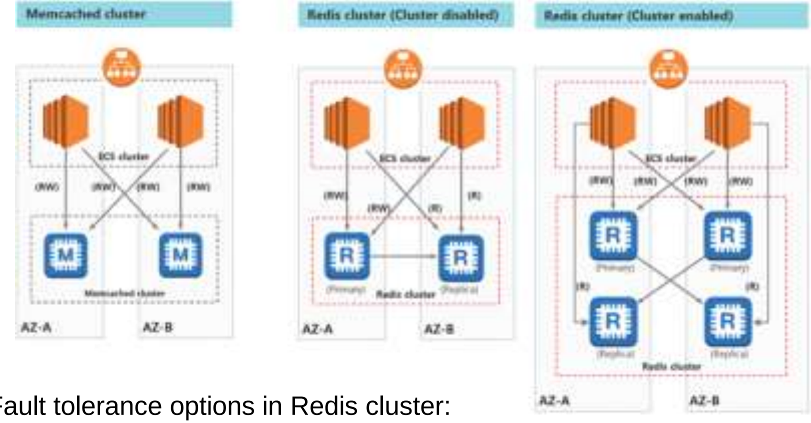
ElastiCache

ElastiCache

	Memcached	Redis (Cluster disabled)	Redis (Cluster enabled)	
Data types	Simple	Complex	Complex	
Encryption	No	Yes	Yes	
Scaling	Yes	Yes	No	Add replica nodes if cluster disabled Cluster mode supports max 15 nodes
Data partitioning	Yes	No	Yes	Cluster supports sharding
High Availability	No	Yes	Yes	Redis non-cluster can auto failover by promoting read node to primary node, clients can be unaware of this. Same for cluster mode
Automatic failover	No	Yes	Yes	
Persistence	No	Yes	Yes	
Backup & Restore	No	Yes	Yes	
Sorted list	No	Yes	Yes	
Geospatial indexing	No	Yes	Yes	



	Memcached	Redis
Sub-millisecond latency	Yes	Yes
Developer ease of use	Yes	Yes
Data partitioning	Yes	Yes
Support for a broad set of programming languages	Yes	Yes
Advanced data structures	-	Yes
Multithreaded architecture	Yes	-
Snapshots	-	Yes
Replication	-	Yes
Transactions	-	Yes
Pub/Sub	-	Yes
Lua scripting	-	Yes
Geospatial support	-	Yes



Fault tolerance options in Redis cluster:

1. Multi-AZ with Automatic Failover: best option when data retention, minimal downtime, and application performance are a priority.
   - Data loss potential - Low. Multi-AZ provides fault tolerance for every scenario, including hardware-related issues.
   - Performance impact - Low. Of the available options, Multi-AZ provides the fastest time to recovery, because there is no manual procedure to follow after the process is implemented.
   - Cost - Low to high. Multi-AZ is the lowest-cost option. Use Multi-AZ when you can't risk losing data because of hardware failure or you can't afford the downtime required by other options in your response to an outage.

1. Daily automatic backups
2. Manual backups using Redis append-only file (AOF)

choose Memcached over Redis if you have the following requirements:

- You need the simplest model possible. - You need to run large nodes with multiple cores or threads. - You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases. - You need to cache objects, such as a database.