AWS RDS Aurora

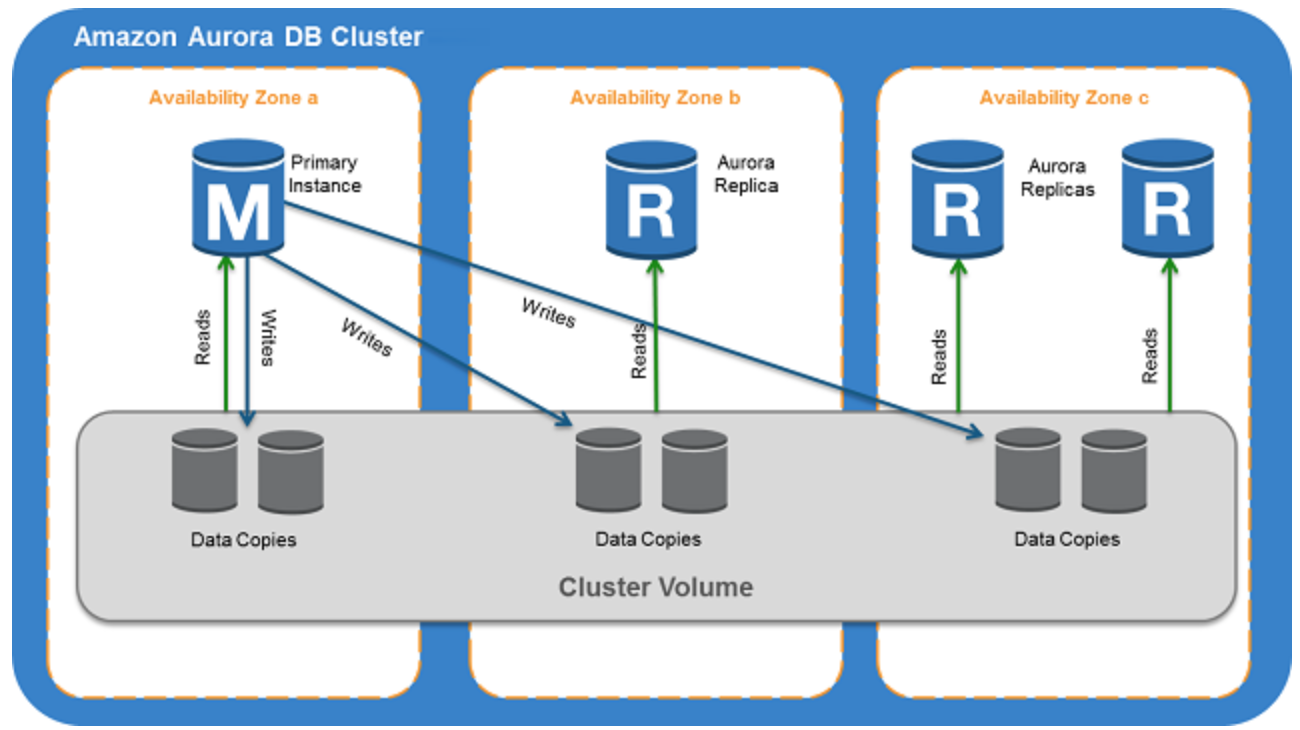
# AWS RDS Aurora

Graphical user interface

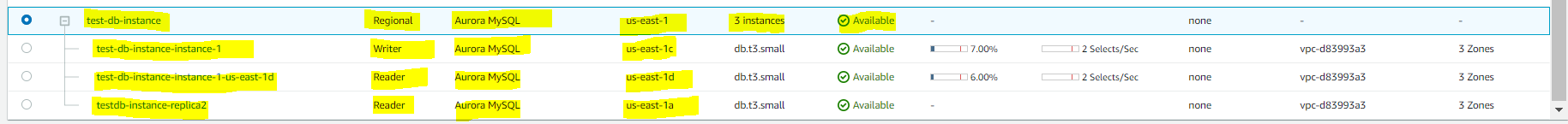
Description automatically generated with low confidence

* AWS Aurora is a relational database engine that combines the speed and reliability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases.
* Aurora is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine i.e. applications developed with MySQL can switch to Aurora with little or no changes
* Aurora delivers up to 5x performance of MySQL without requiring any changes to most MySQL applications
* Aurora PostgreSQL delivers up to 3x performance of PostgreSQL.
* RDS manages the Aurora databases, handling time-consuming tasks such as provisioning, patching, backup, recovery, failure detection and repair.
* **Based on the database usage, Aurora storage will automatically grow, from 10GB to 64TiB in 10GB increments with no impact to database performance**

## Aurora DB Clusters



* Aurora DB cluster consists of one or more DB instances and a cluster volume that manages the data for those DB instances.
* An Aurora cluster volume is a virtual database storage volume that spans multiple AZs, with each AZ having a copy of the DB cluster data
* Two types of DB instances make up an Aurora DB cluster:
  + **Primary DB instance**
    - Supports read and write operations, and performs all of the data modifications to the cluster volume.
    - Each Aurora DB cluster has one primary DB instance.
  + **Aurora Replica**
    - Connects to the same storage volume as the primary DB instance and supports only read operations.
    - Each Aurora DB cluster can have up to **15 Aurora Replicas** in addition to the primary DB instance.
    - Provides high availability by locating Replicas in separate AZs
    - Aurora automatically fails over to an Aurora Replica in case the primary DB instance becomes unavailable.
    - Failover priority for Aurora Replicas can be specified.
    - Aurora Replicas can also offload read workloads from the primary DB instance

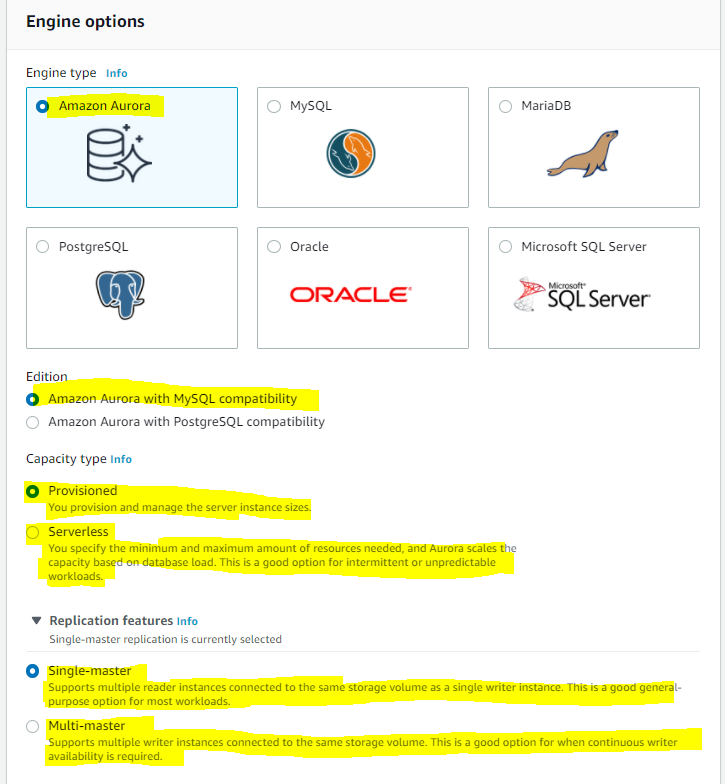


* **For Aurora multi-master clusters (only supported on MySQL engine)**

A screenshot of a computer

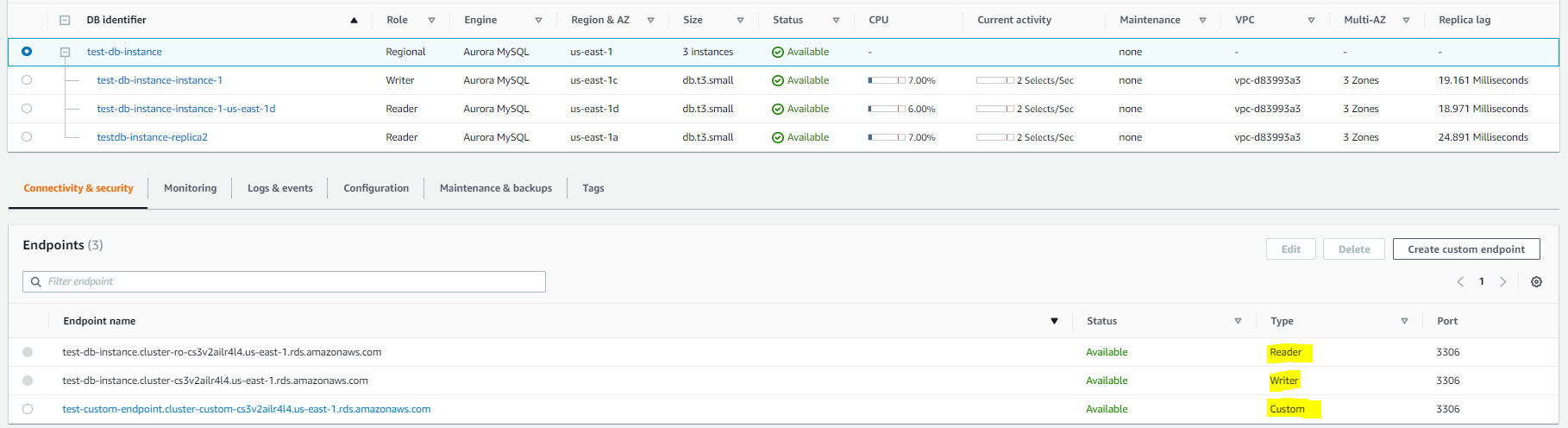
Description automatically generated with medium confidence

* + all DB instances have read/write capability, with no difference between primary and replica.



## Aurora Connection Endpoints

* Aurora involves a cluster of DB instances instead of a single instance
* Endpoint refers to an intermediate handler with the host name and port specified to connect to the cluster
* Aurora uses the endpoint mechanism to abstract these connections



### Cluster endpoint

* **Cluster endpoint (or writer endpoint)** for an Aurora DB cluster connects to the current primary DB instance for that DB cluster.
* Cluster endpoint is the only one that can perform write operations such as DDL statements as well as read operations
* Each Aurora DB cluster has one cluster endpoint and one primary DB instance
* Cluster endpoint provides failover support for read/write connections to the DB cluster. If the current primary DB instance of a DB cluster fails, Aurora automatically fails over to a new primary DB instance. During a failover, the DB cluster continues to serve connection requests to the cluster endpoint from the new primary DB instance, with minimal interruption of service.

Example:

**Cluster-endpoint (writer-endpoint)-> Only one -> test-db-instance.cluster-cs3v2ailr4l4.us-east-1.rds.amazonaws.com**

### Reader endpoint

* Reader endpoint for an Aurora DB cluster provides load-balancing support for read-only connections to the DB cluster.
* Use the reader endpoint for read operations, such as queries.
* Reader endpoint reduces the overhead on the primary instance by processing the statements on the read-only Aurora Replicas.
* Each Aurora DB cluster has one reader endpoint.
* If the cluster contains one or more Aurora Replicas, the reader endpoint load-balances each connection request among the Aurora Replicas.

Example:

**Reader-endpoint -> Only one which load balances all read replicas -> test-db-instance.cluster-ro-cs3v2ailr4l4.us-east-1.rds.amazonaws.com**

### Custom endpoint

* Custom endpoint for an Aurora cluster represents a set of DB instances that you choose.
* Aurora performs load balancing and chooses one of the instances in the group to handle the connection.
* An Aurora DB cluster has no custom endpoints until one created and upto five custom endpoints can be created for each provisioned Aurora cluster.
* Aurora Serverless clusters does not support custom endpoints.

Example:

**Custom-endpoint -> Can create upto 5 which can load balances selected instances->** [**test-custom-endpoint.cluster-custom-cs3v2ailr4l4.us-east-1.rds.amazonaws.com**](https://console.aws.amazon.com/rds/home?region=us-east-1#custom-endpoint:id=test-custom-endpoint)

### Instance endpoint

* An instance endpoint connects to a specific DB instance within an Aurora cluster and provides direct control over connections to the DB 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.

Example:

**Instance endpoint -> One for each instance -> testdb-instance-replica2.cs3v2ailr4l4.us-east-1.rds.amazonaws.com**

## High Availability and Replication

* Aurora is designed to offer greater than 99.99% availability
* Aurora provides data durability and reliability
  + by **replicating the database volume six ways across three Availability Zones in a single region**
  + backing up the data continuously to  S3.
* Aurora transparently recovers from physical storage failures; instance failover typically takes less than 30 seconds.
* If the primary DB instance fails, Aurora automatically fails over to a new primary DB instance, by either promoting an existing Aurora Replica to a new primary DB instance or creating a new primary DB instance
* **Aurora automatically divides the database volume into 10GB segments spread across many disks. Each 10GB chunk of the database volume is replicated six ways, across three Availability Zones**
* **RDS databases for e.g. MySQL, Oracle etc. have the data in a single AZ**
* Aurora is designed to transparently handle
  + the loss of up to two copies of data without affecting database write availability and
  + up to three copies without affecting read availability.
* Aurora storage is also self-healing. Data blocks and disks are continuously scanned for errors and repaired automatically.
* **Aurora Replicas share the same underlying volume as the primary instance. Updates made by the primary are visible to all Aurora Replicas**
* **As Aurora Replicas share the same data volume as the primary instance, there is virtually no replication lag**
* **Any Aurora Replica can be promoted to become primary without any data loss and therefore can be used for enhancing fault tolerance in the event of a primary DB Instance failure.**
* **To increase database availability, 1 to 15 replicas can be created in any of 3 AZs, and RDS will automatically include them in failover primary selection in the event of a database outage.**

Graphical user interface

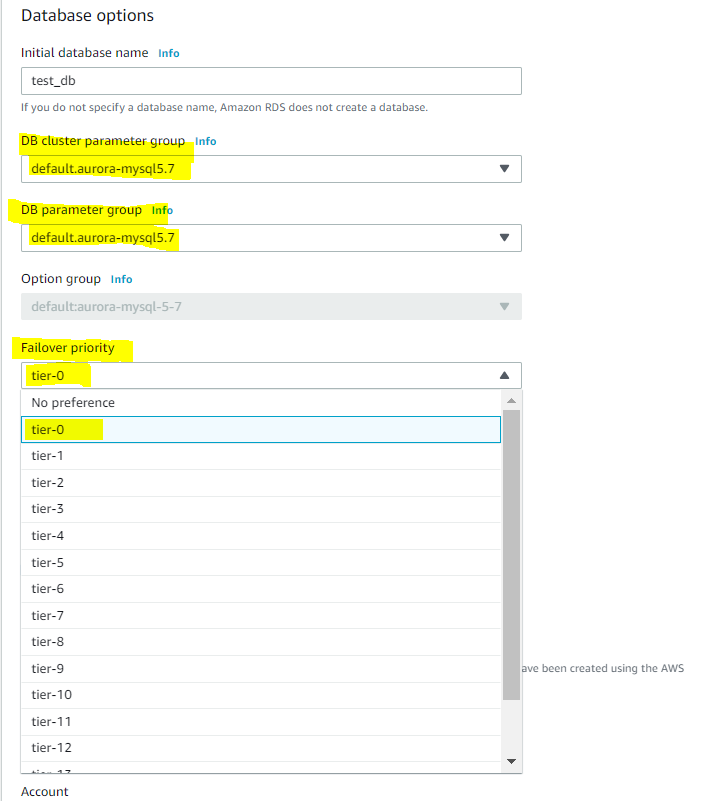
Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

## Aurora Failovers

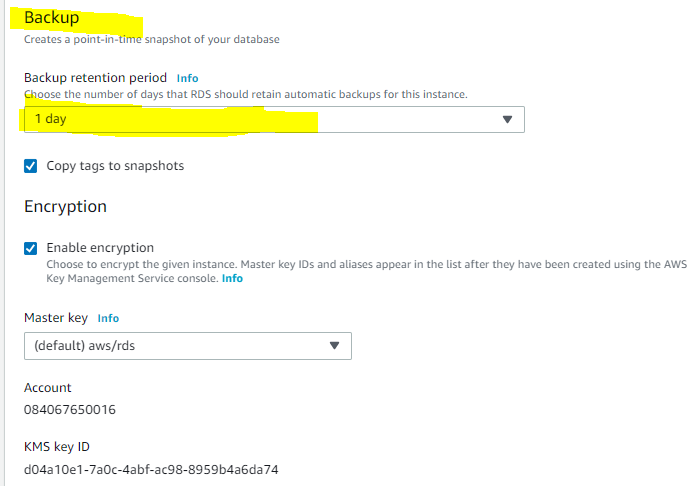
* Aurora automatically fails over, if the primary instance in a DB cluster fails, in the following order:
  + If Aurora Read Replicas are available, promote an existing Read Replica to the new primary instance.
  + If no Read Replicas are available, then create a new primary instance.
* If there are multiple Aurora Read Replicas, the criteria for promotion is based on the **priority** that is defined for Read Replicas.
  + Priority number can vary from 0 to 15 and can be modified any time.
  + Aurora PostgreSQL promotes the Aurora Replica with the highest priority to the new primary instance.
  + For Read Replicas with same priority, Aurora PostgreSQL promotes the replica that is largest in size or in an arbitrary manner.
* **During the failover, AWS modifies the cluster endpoint to point to the newly created/promoted DB instance.**
* Applications experience minimal interruption of service if they connect using the cluster endpoint and implement connection retry logic.



## Security

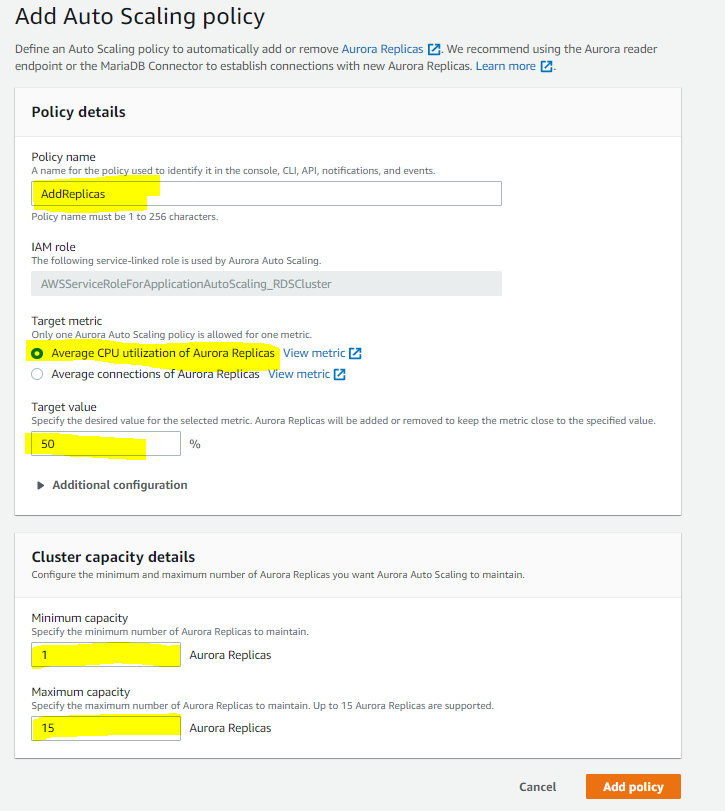
* Aurora uses SSL (AES-256) to secure the connection between the database instance and the application
* Aurora allows database encryption using keys managed through AWS Key Management Service (KMS).
* Encryption and decryption are handled seamlessly.
* With Aurora encryption, data stored at rest in the underlying storage is encrypted, as are its automated backups, snapshots, and replicas in the same cluster.
* Encryption of existing unencrypted Aurora instance is not supported. Create a new encrypted Aurora instance and migrate the data

## Backup and Restore

* Automated backups are always enabled on Aurora DB Instances.
* Aurora automatically maintains 6 copies of the data across 3 AZs and will automatically attempt to recover the database in a healthy AZ with no data loss
* Backups do not impact database performance.
* Back retention period can be minimum 1 day & maximum 35 days.
* Aurora also allows creation of manual snapshots
* Restoring a snapshot creates a new Aurora DB instance.
* If in any case the data is unavailable within Aurora storage, DB Snapshot can be restored, or point-in-time restore operation can be performed to a new instance.
* Latest restorable time for a point-in-time restore operation can be up to 5 minutes in the past.
* **Deleting Aurora database deletes all the automated backups (with an option to create a final snapshot), but would not remove the manual snapshots.**
* Snapshots (including encrypted ones) can be shared with another AWS accounts
* 

## Aurora Scaling

* **Aurora storage scaling in built-in and will automatically grow, up to 64 TB (soft limit), in 10GB increments with no impact to database performance. There is no need to provision storage in advance**
* **Compute Scaling**
  + **Instance scaling**
    - Vertical scaling the master the instance. Memory and CPU resources are modified by changing the DB Instance class.
    - scaling the read replica and promoting it to master using forced failover which provides a minimal downtime
  + Read scaling
    - provides horizontal scaling with up to 15 read replicas
* **Auto Scaling**
  + Scaling policies to add read replicas with min and max replica count based on scaling CloudWatch CPU or connections metrics condition.

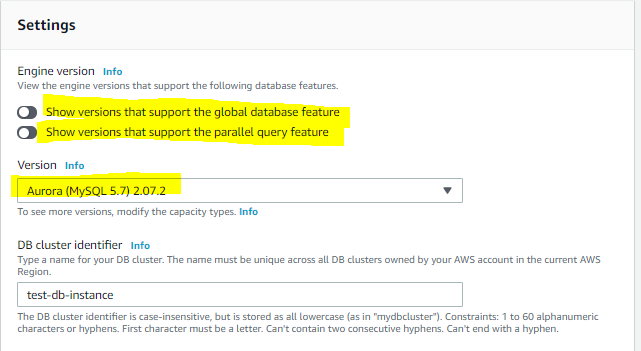


## Aurora Backtrack

* Backtracking “rewinds” the DB cluster to the specified time
* Backtracking performs in place restore and does not create a new instance. There is a minimal downtime associated with it.
* Backtracking is available for Aurora with MySQL compatibility
* Backtracking is not a replacement for backing up the DB cluster so that you can restore it to a point in time.
* With backtracking, there is a target backtrack window and an actual backtrack window:
  + **Target backtrack window** is the amount of time you **WANT** the DB cluster can be backtracked for e.g 24 hours. Limit for a backtrack window is 72 hours.
  + **Actual backtrack window** is the actual amount of time you **CAN** backtrack the DB cluster, which can be smaller than the target backtrack window. The actual backtrack window is based on the workload and the storage available for storing information about database changes, called change records
* Aurora DB cluster with backtracking enabled, generates change records.
* Aurora retains change records for the target backtrack window, and charges an hourly rate for storing them.
* Both the target backtrack window and the workload on the DB cluster determine the number of change records stored.
* Workload is the number of changes made to the DB cluster in a given amount of time. If the workload is heavy, you store more change records in the backtrack window than you do if your workload is light.
* Backtracking affects the entire DB cluster and can’t selectively backtrack a single table or a single data update.
* Backtracking provides the following advantages over traditional backup and restore:
  + Undo mistakes – revert destructive action, such as a DELETE without a WHERE clause
  + Backtrack DB cluster quickly – Restoring a DB cluster to a point in time launches a new DB cluster and restores it from backup data or a DB cluster snapshot, which can take hours. Backtracking a DB cluster doesn’t require a new DB cluster and rewinds the DB cluster in minutes.
  + Explore earlier data changes – repeatedly backtrack a DB cluster back and forth in time to help determine when a particular data change occurred

## Aurora Parallel Query

* **Aurora Parallel Query refers to the ability to push down and distribute the computational load of a single query across thousands of CPUs in Aurora’s storage layer.**
* Without Parallel Query, a query issued against an Aurora database would be executed wholly within one instance of the database cluster; this would be similar to how most databases operate.
* **Parallel Query is a good fit for analytical workloads requiring fresh data and good query performance, even on large tables.**
* Parallel Query provides following benefits
  + Faster performance: Parallel Query can speed up analytical queries by up to 2 orders of magnitude.
  + Operational simplicity and data freshness: you can issue a query directly over the current transactional data in your Aurora cluster.
  + Transactional and analytical workloads on the same database: Parallel Query allows Aurora to maintain high transaction throughput alongside concurrent analytical queries.
* **Parallel Query can be enabled and disabled dynamically at both the global and session level using the aurora\_pq parameter.**
* **Parallel Query is available for the MySQL 5.6-compatible version of Aurora**



## Aurora Global Database

Graphical user interface

Description automatically generated

* Aurora global database consists of **one primary AWS Region** (Will host primary cluster) where the data is mastered, and **up** **to five read-only, secondary AWS Regions**(Will host induvial secondry cluster)**.**
* Aurora cluster in the primary AWS Region where your data is mastered performs both read and write operations. The clusters in the secondary Regions enable low-latency reads.
* Aurora replicates data to the secondary AWS Regions with typical latency of under a second.
* Secondary clusters can be scaled independently by adding one of more DB instances (Aurora Replicas) to serve read-only workloads.
* Aurora global database uses dedicated infrastructure to replicate the data, leaving database resources available entirely to serve applications.
* Applications with a worldwide footprint can use reader instances in the secondary AWS Regions for low-latency reads.
* In case of a disaster or an outage, one of the cluster in a secondary AWS Regions can be promoted to take full read/write workloads in under a min.

## Aurora Clone

* Aurora cloning feature helps create Aurora cluster duplicates quickly and cost-effectively
* Creating a clone is faster and more space-efficient than physically copying the data using a different technique such as restoring a snapshot.
* Aurora cloning uses a copy-on-write protocol.
* Aurora clone requires only minimal additional space when first created. In the beginning, Aurora maintains a single copy of the data, which is used by both the original and new DB clusters.
* Aurora allocates new storage only when data changes, either on the source cluster or the cloned cluster.

## sGraphical user interface Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated

## AWS Certification Exam Practice Questions

1. Company wants to use MySQL compatible relational database with greater performance. Which AWS service can be used?
   1. **Aurora**
   2. RDS
   3. SimpleDB
   4. DynamoDB
2. An application requires a highly available relational database with an initial storage capacity of 8 TB. The database will grow by 8 GB every day. To support expected traffic, at least eight read replicas will be required to handle database reads. Which option will meet these requirements?
   1. DynamoDB
   2. Amazon S3
   3. **Amazon Aurora**
   4. Amazon Redshift
3. A company is migrating their on-premise 10TB MySQL database to AWS. As a compliance requirement, the company wants to have the data replicated across three availability zones. Which Amazon RDS engine meets the above business requirement?
   1. Use Multi-AZ RDS
   2. Use RDS
   3. **Use Aurora**
   4. Use DynamoDB