AWS SysOps Administrator – Associate Level

High Availability

Learning Objectives

By the end of this lesson, you will be able to:

- Analyse and plot the differences between elasticity and scalability on various AWS services
- Implement different ways of handling RDS failover to avoid database intervention
- Encrypt and share the RDS snapshot on the database
- Create Aurora DB cluster and connect it to a DB instance to work with its functionality and Replicas

Elasticity and Scalability

Elasticity

- Scales up when demand is high and shrinks when the demand is low
- Pay only for the service you need and when you need it
- Matches current resources with the actual amount required at any given point of time
- Used for a short period of time in case of EC2 instances

Scalability

- Adds resources to accommodate large loads of cloud volume
- Scales up the hardware
- Increases volume capacity
- Adds more nodes

Scalability and Elasticity of Key AWS Services

RDS: Elasticity

- Has limited elasticity

EC2: Elasticity

- Has autoscaling which grows with demand and shrinks back when demand is low

DynamoDB: Elasticity

- Increases or decreases read or write throughput capacity on demand

Scalability and Elasticity of Key AWS Services

EC2: Scalability

- Increases the size of an instance
- Provides multiple instance types
- Launches multiple instances

DynamoDB: Scalability

- Stores more data without provisioning any hardware

RDS: Scalability

- Increases size of an instance
- Launches read replicas
- Has multiple instance types available

TECHNOLOGY

RDS and Multi-AZ Failovers

What Is RDS?

- Amazon RDS makes it easy to set up, operate, and scale a relational database in the cloud.
- It provides cost-efficient and resizable capacity while automating time-consuming administration tasks.
- It frees you to focus on your applications to provide fast performance, high availability, security, and compatibility.

Benefits of RDS

1. Easy to administer
2. Highly scalable
3. Available and durable
4. Fast
5. Secure
6. Inexpensive

RDS Failover with Multi-AZ

- Loss of availability in the primary availability zone
- Loss of network connectivity to the primary instance
- Resource failure with the underlying virtualized resources
- Storage failure on the primary instance
- Change in service type of the DB instance and maintenance

Handling Failovers

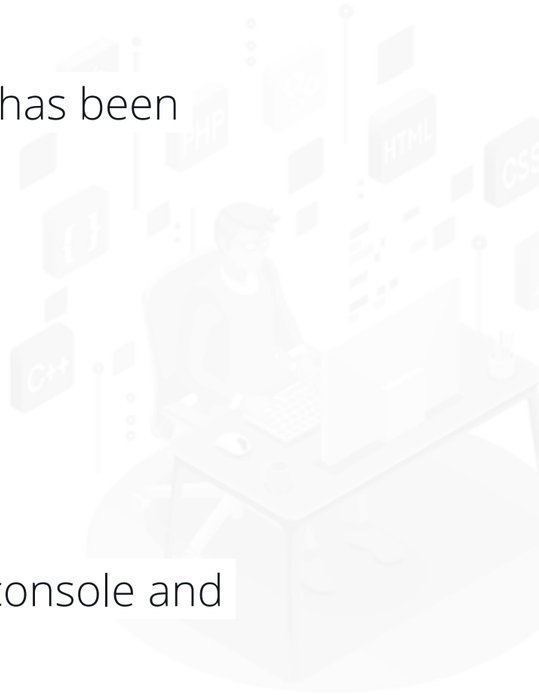
- Amazon RDS handles failovers automatically to resume database operations without any intervention.

- The primary DB instance switches over automatically to the standby replica if any of the following conditions occur:

    - An availability zone outage
    - Primary DB instance failure
    - DB instance server type is changed
    - OS is under software patching
    - Manual failover with Reboot

Ways to Determine RDS Failovers

- DB event subscriptions can be set up to notify by an email or SMS that a failover has been initiated
- Viewing the DB events by using the Amazon RDS console or API operations
- Viewing the current state of the Multi-AZ deployment by using the Amazon RDS console and API operations



RDS and Read Replicas

What Are Read Replicas?

RDS uses the built-in replication functionality of the DB engines to create a special type of DB instance called the Read Replica.

Updates made to the primary DB instance are asynchronously copied to the read replica.

It lets you elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

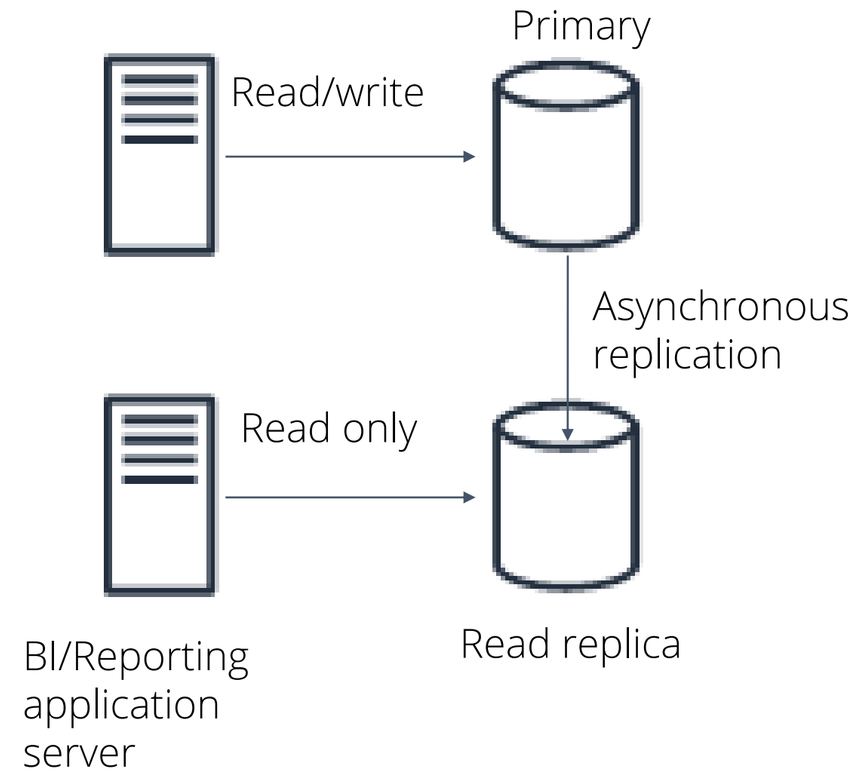
Load on the primary DB instance can be reduced by routing read queries from the application to the read replica.

Overview of RDS Read Replicas

Deploying one or more read replicas for a given source DB instance makes sense in a variety of scenarios, including the following:

1. Excess read traffic of the compute capacity can be directed to one or more read replicas
2. Serving read traffic while source DB instance is unavailable
3. Data warehousing where business reporting queries have to be run against read replicas
4. Implementing disaster recovery

Overview of RDS Read Replicas



Creating a Read Replica

- Replicas can be created with the existing database or server instance with AWS console.
- RDS takes the snapshot of the created replica and begins replication.
- During replication, I/O suspension will occur on the source DB instance.
- Automatic backups on the source DB instance should be enabled by setting the backup retention period to any value other than 0.
- RDS captures only one snapshot out of multiple snapshots.

Benefits of RDS Read Replicas

1. Enhanced Performance
2. Increased Availability
3. Designed for Security

Creating a Read Replica

Duration: 10 Min.

Problem Statement:

Create a read replica on an existing database.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create a read replica are as follows:

1. Login to your AWS lab and open **Amazon RDS console**

1. Choose **Databases** in the navigation pane

1. Choose the database that you want to use as a source in the read replica

1. In the **Actions** tab, choose **Create read replica**

1. Choose your instance specification

1. Choose **other settings** and hit **Create read replica**

Creating a Read Replica in Multiple Regions

Duration: 10 Min.

Problem Statement:

Create a read replica in multiple AWS regions.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create a read replica in multiple AWS regions are as follows:

1. Login to your AWS lab and open **Amazon RDS console**

1. Choose **Databases** in the navigation pane

1. Choose the database that you want to use as a source in the read replica

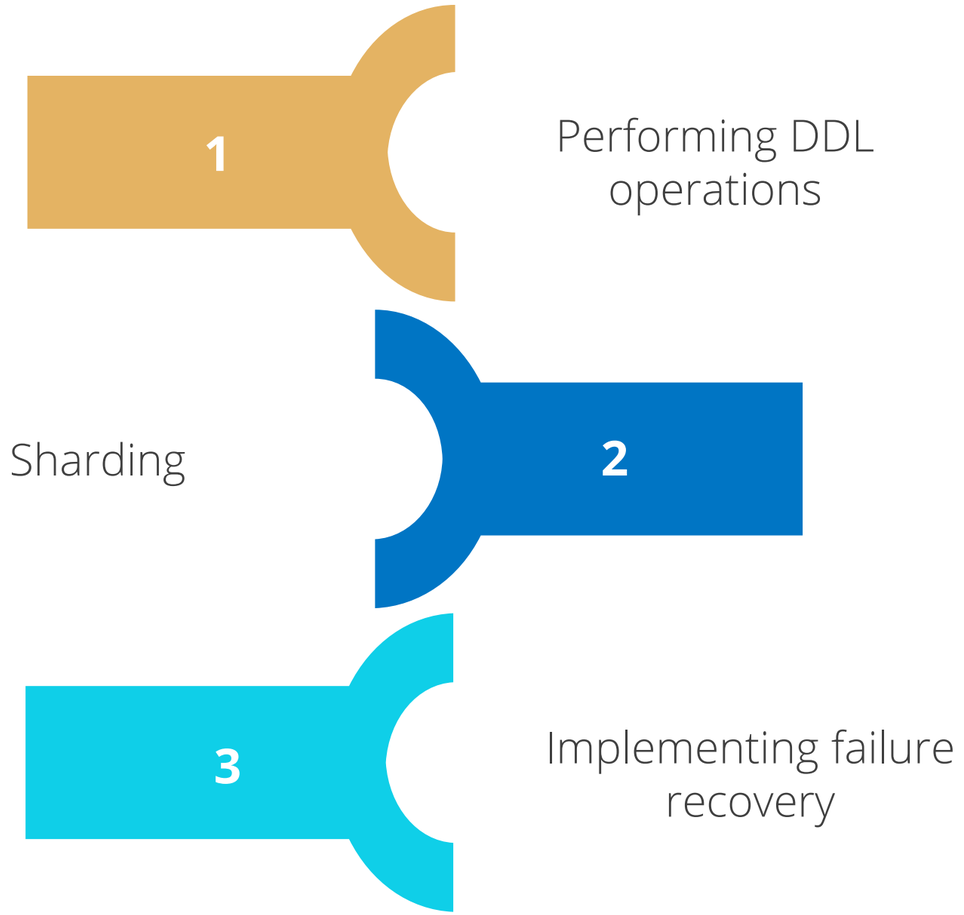
1. In the other settings, under **Network and Security region**, choose a value for **Destination region** and one for **Destination DB subnet group**

1. Enable encryption through master key in **AWS KMS**

1. Finally, choose **Create read replica**

Promoting a Read Replica to a Standalone DB Instance

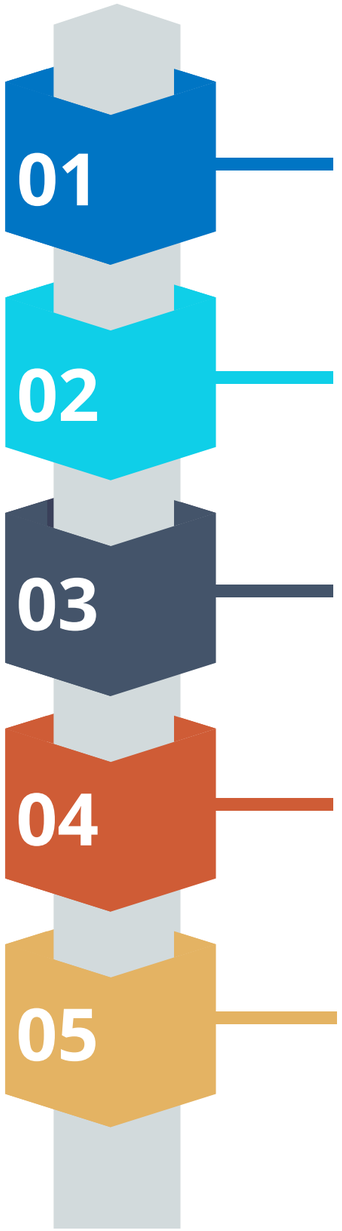
Reasons to promote a read replica to a standalone DB instance are:



TECHNOLOGY

Amazon RDS and Cross-Region Replication

How It Works?



Begins configuring the source DB instance as a replication source and sets the status to *modifying*

Sets up the specified read replica in the destination AWS Region and sets the status to *creating*

Creates an automated DB snapshot of the source DB instance in the source AWS Region

Begins a cross-region snapshot copy for the initial data transfer

When the read replica reaches the available status, Amazon RDS starts by replicating the changes made to the source instance when the *Create read replica* operation had started

Sharing Snapshots

Share an Encrypted RDS Snapshot

Duration: 10 Min.

Problem Statement:

Share an encrypted Amazon RDS snapshot to another account.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to share an encrypted Amazon RDS snapshot are as follows:

1. Login to your AWS lab and open **Amazon RDS console**

1. Add the target account to a custom (non-default) KMS key

1. Copy the snapshot using the customer managed key, and then share the snapshot with the target account

1. Copy the shared DB snapshot from the target account

AWS Aurora

What is AWS Aurora?

01 A MySQL- and PostgreSQL- compatible relational database built for the cloud, that combines the performance and availability of traditional enterprise databases

02 It’s five times faster than the standard MySQL databases and three times faster than the standard PostgreSQL databases

03 Provides security, availability, and reliability of commercial databases at ten percent of the cost

What is AWS Aurora?

04 Managed fully by Amazon RDS, which automates the time-consuming administration tasks

05 Features a distributed, fault-tolerant, self-healing storage system that auto scales up to 64TB per database instance

06 Delivers high performance and availability with up to fifteen low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three Availability Zones (AZs)

Create an Aurora MySQL Database Cluster

Duration: 10 Min.

Problem Statement:

Create an Aurora MySQL database cluster with *Easy Create* enabled on the AWS Console.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to create an Aurora MySQL database cluster are as follows:

1. Login to your AWS lab and open **Amazon RDS console**

1. Choose the AWS region in which you want to create a database

1. In the navigation pane, choose **Databases**

1. Choose **Create database** with **Easy Create** option enabled

1. Configure all the necessary settings and create the database

Connect to an Instance in a DB Cluster

Duration: 10 Min.

Problem Statement:

Connect Aurora MySQL database to an instance in the DB cluster.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to connect to a database instance are as follows:

1. Login to your AWS lab and open **Amazon RDS console**

1. In the navigation pane, choose **Database** with the cluster name which is already created

1. Connect to MySQL using the commands

1. Configure all the necessary settings and connect to a instance

Aurora Scaling

Used to automatically adjust the number of Aurora replicas provisioned for an Aurora DB cluster

Enables your Aurora DB cluster to handle a sudden increase in connectivity or workload

Removes the unnecessary Aurora replicas to avoid cost of unused provision instances

Defines the minimum and maximum number of replicas that can be managed

Aurora Scaling Policies

The various components of the policies are listed below:

1. A service-linked role
2. A target metric
3. Minimum and maximum capacity
4. A cooldown period

Adding the Aurora Autoscaling Policy Using AWS Console

Steps to add the Aurora autoscaling policy using AWS console are as follows:

1. Login to your AWS lab and open **Amazon RDS console.**

1. In the navigation pane, choose **Databases.**

1. Choose the Aurora DB cluster to which you want to add the policy.

1. Choose the **Logs & events** tab.

1. In the **Auto scaling policies** section, choose **Add**. The **Add Auto Scaling policy** dialog box appears.

Adding the Aurora Autoscaling Policy Using AWS Console

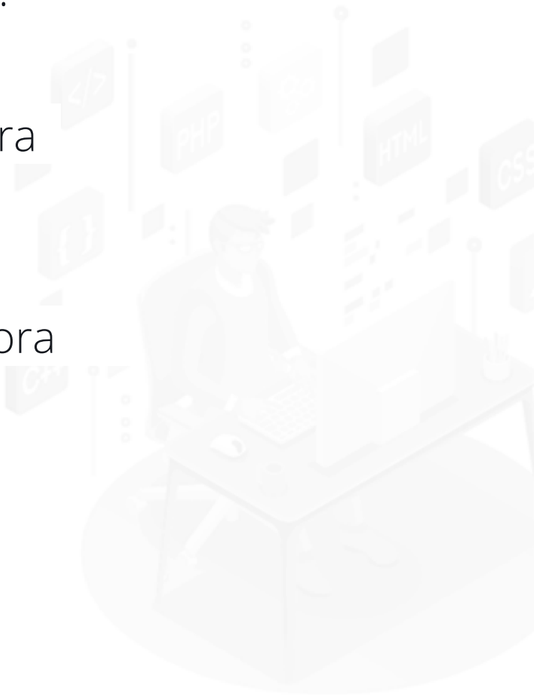
6. For **Policy Name**, type the policy name.

7. Open **Additional Configuration** to create a scale-in or a scale-out cooldown period.

8. For **Minimum capacity**, type the minimum number of Aurora replicas that the Aurora autoscaling policy is required to maintain.

9. For **Maximum capacity**, type the maximum number of Aurora replicas that the Aurora autoscaling policy is required to maintain.

10. Choose **Add policy**.



Amazon Aurora Database Clusters

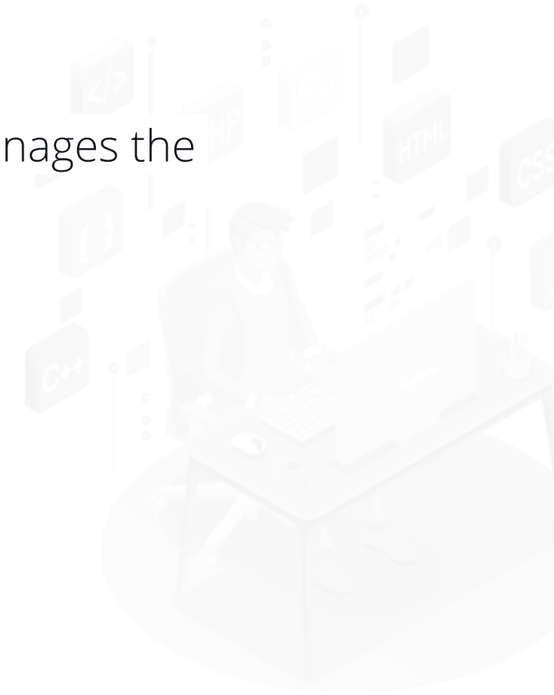
Amazon Aurora DB Clusters

Each cluster consists of one or more DB instances and a cluster volume that manages the data for those DB instances.

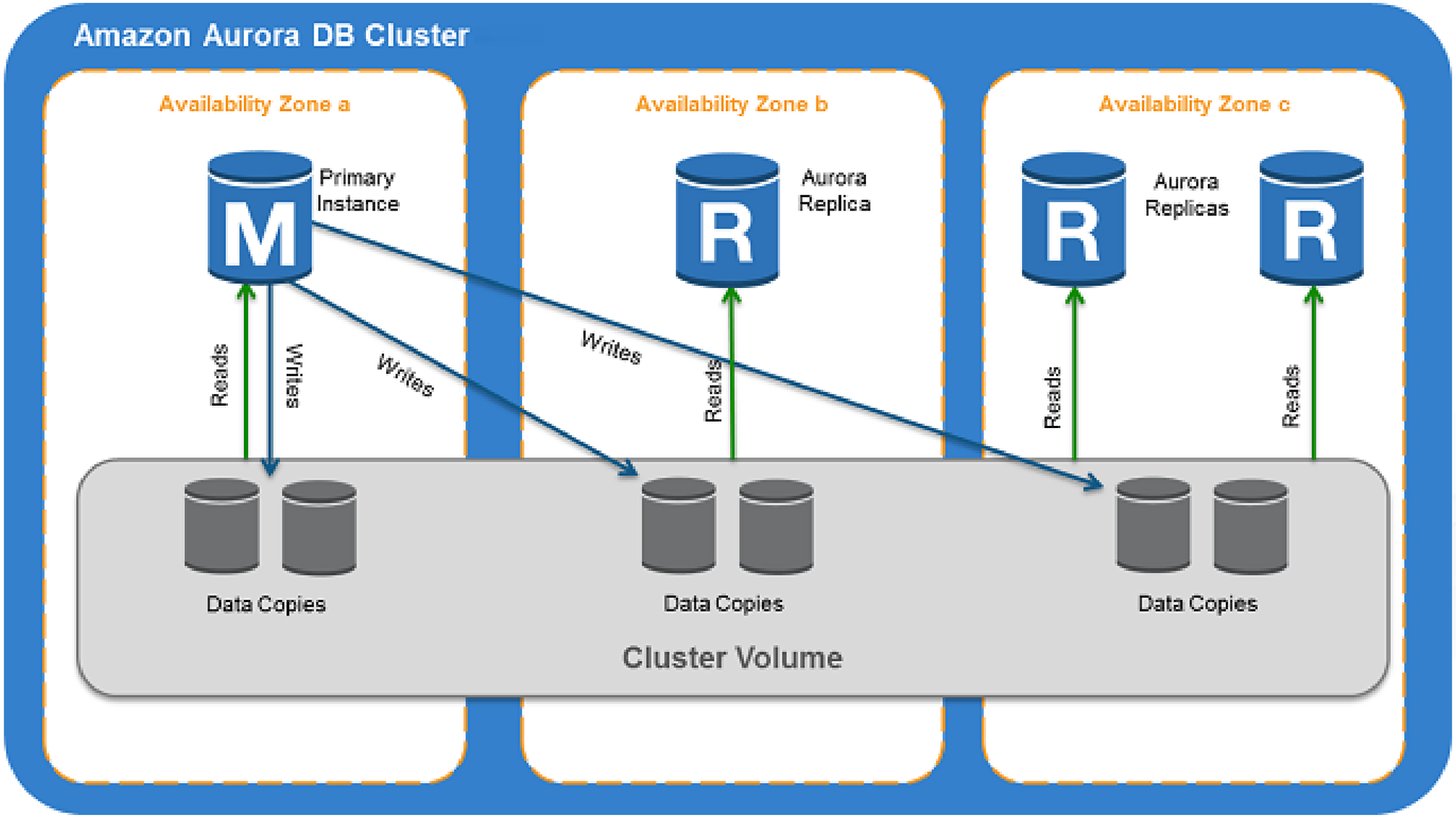
Two types of DB instances make an Aurora DB cluster:

1. Primary DB instance

1. Aurora Replica



Amazon Aurora DB Clusters



src: aws.amazon.com

Aurora Replicas

Aurora Replicas

1. Independent endpoints in an Aurora DB cluster used for scaling read operations
2. Good for read scaling as they are dedicated to read operations
3. Replica lag varies depending on the rate of database change
4. Replicas are used as failover targets to increase availability

Adding Aurora Replicas to a DB cluster

Duration: 10 Min.

Problem Statement:

Add Amazon Aurora replicas to a database cluster.

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to add Aurora replicas to a database cluster are as follows:

1. Login to your AWS lab and open **Amazon RDS console.**

1. In the navigation pane, choose **Databases,** and then select a DB cluster to create a DB instance.

1. For **Actions**, choose **Reader** and specify options for **Read Replica.**

1. Choose **Add Reader** to create the Aurora replica.

Aurora Serverless

Aurora Serverless

01 An on-demand autoscaling configuration for Amazon Aurora with MySQL-compatible editions

02 It helps to run a database in the cloud without managing any database instances

03 It's a simple and cost-effective option for infrequent, intermittent, or unpredictable workloads

Benefits of Serverless Aurora

1. Simple
2. Scalable
3. Cost-Effective
4. Highly Available

Application Areas

1. Infrequently used applications
2. Variable and unpredictable workloads
3. Developing and testing databases
4. Multi-tenant applications

Troubleshooting Autoscaling Issues

Instances Not Launching Autoscaling

If an instance is not launching into an autoscaling group, look for the following issues:

1. Associated key pair does not exist
2. Security group does not exist
3. Autoscaling configuration is not working correctly

Instances Not Launching Autoscaling

4. Autoscaling group is not present

5. Instance type specified is not supported in the AZ

6. AZ is no longer supported

Instances Not Launching Autoscaling

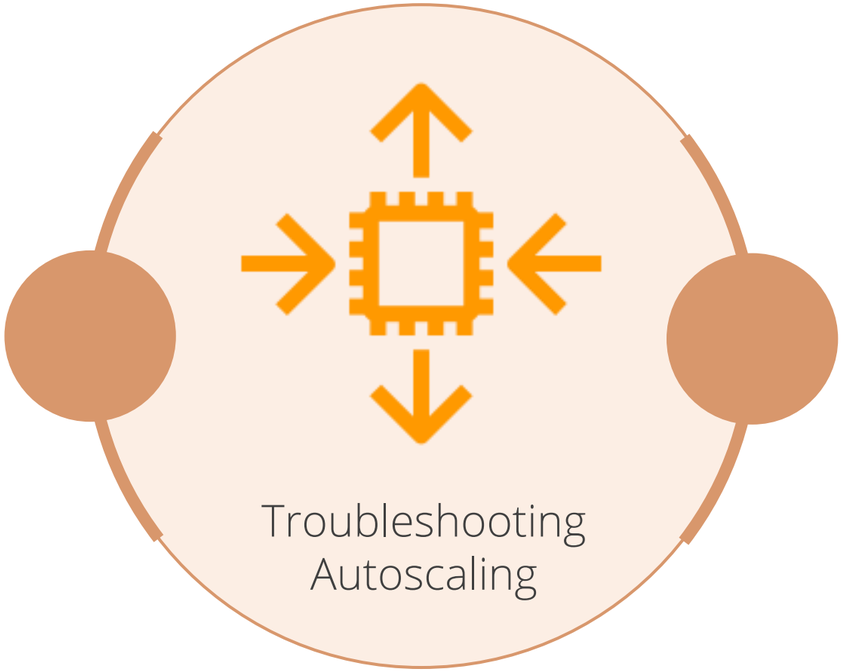
7. Invalid EBS device mapping

8. Autoscaling service is not enabled on your account

9. Attempt to attach an EBS block device to an instance-store AMI

Troubleshooting Autoscaling

Use various auto scaling actions, such as HealthCheck, Terminate, and AZRebalance for various auto scaling issues.



Ensure that the testing client is not behind a proxying cache and sends requests all the way to the ELB.

Content Delivery Network

Amazon CloudFront

- Fast content delivery network service that securely delivers data and applications globally with high-transfer speed
- Integrated with AWS (both physical locations) connected to the AWS global infrastructure
- It works seamlessly with services (DDos and Shield) to run custom codes and customize user experience

Benefits of Amazon CloudFront

1. Fast and Global
2. Security at the Edge
3. Highly Programmable
4. Deep Integration with AWS

Set up CloudFront to Deliver Network

1. Specify the origin servers from which CloudFront will get files to distribute over locations
2. Upload the files to the origin servers
3. Create CloudFront distribution to inform servers to get the user request files
4. Assigns a domain name to the new distribution that is returned as a response
5. Sends distribution configuration to all of its edge locations and points of presence

Cache Hit Ratios

The ratio of requests served from edge locations is known as cache hit ratio

More requests from edge locations imply better performance

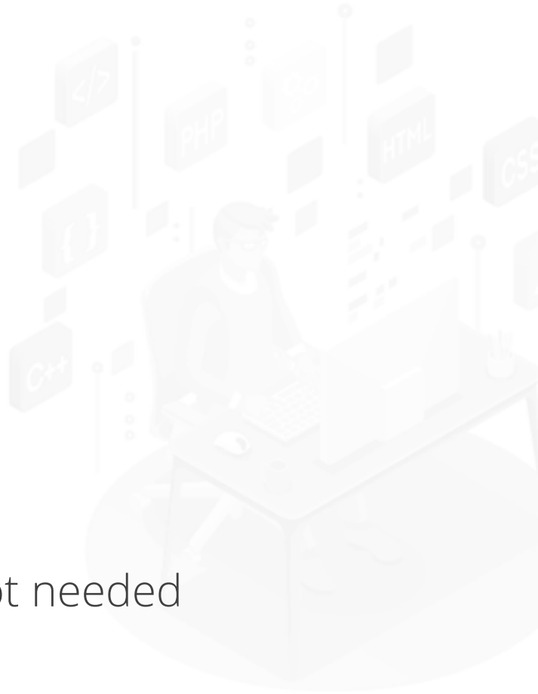
Reduced load on origin server and latency

Improving Cache Hit Ratios

Improve cache hit ratios by:

- Specifying how long CloudFront caches your objects
- Caching based on query string parameters
- Caching based on cookie values and cookie requests
- Removing accept-encoding header when compression is not needed
- Serving media content by using HTTP

Improving Cache Hit Ratios



Key Takeaways

- RDS provides cost-efficient and resizable capacity while automating time-consuming administrative tasks.
- Read replicas let you elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- Aurora is five times faster than the standard MySQL databases and three times faster than the standard PostgreSQL databases.
- Amazon CloudFront is a fast content delivery network service that securely delivers data and applications globally with high-transfer speed.

Launch an RDS Instance with Multi-AZ

Problem Statement:

Launch an RDS instance with the multi-AZ option in a private and a public subnet.

Background of Problem Statement:

As a SysOps administrator, create and launch an RDS instance to restore a database with the multi-AZ option.

LESSON-END PROJECT