

Cloud



Caltech | Center for Technology & Management Education

Post Graduate Program in Cloud

Cloud



Caltech

**Center for Technology &
Management Education**

AWS Certified SysOps Administrator – Associate Level



High Availability and Disaster Recovery

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



A Day in the Life of an AWS Administrator

You work as an architect for a company that is seeking a variety of high-availability solutions for its application and data. You must consider the following requirements when providing solutions to the organizations:

- Ensure that they have the correct number of Amazon EC2 instances available to handle the application's load.
- A database solution that manages common database management duties while providing cost-effective, resizable capacity for an industry-standard relational database.
- For high availability, a solution that allows your company to have one or more read-only copies of the database instance in the same or separate AWS Region.

A Day in the Life of an AWS Administrator

- They are looking for a method to distribute incoming traffic to multiple targets in one or more Availability Zones.
- They want a solution that can provide a scalable, high-performance in-memory cache at a low cost.
- The company would also benefit from having a few disaster recovery solutions on hand to aid in the rapid recovery of critical systems in the event of a calamity (system failure).

To achieve all the above along with some additional features, you will be learning a few concepts in this lesson that will help you find solutions for the above-given scenario.

Elasticity and Scalability

Scalability

Scalability refers to the ability to scale up or down IT resources as needed to meet the changing demands in cloud computing.

It's the ability to handle more traffic by scaling up the hardware (scale up) or adding nodes (scale out).

There are two types of scaling:

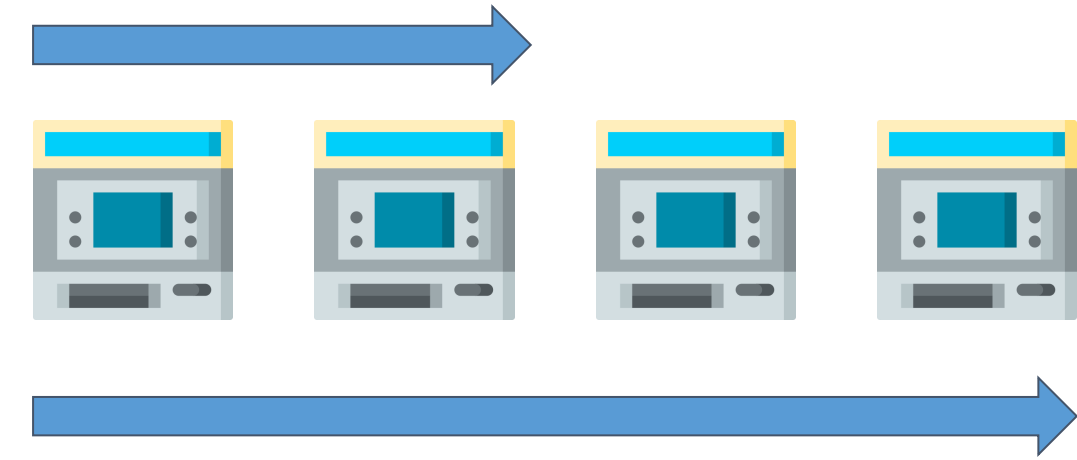
- Vertical Scaling
- Horizontal Scaling



Horizontal Scaling

Horizontal Scalability means increasing the number of instances/systems for an application.

- Example: Increasing the number of ec2 instances from 2 to 4 to handle the increasing workload.



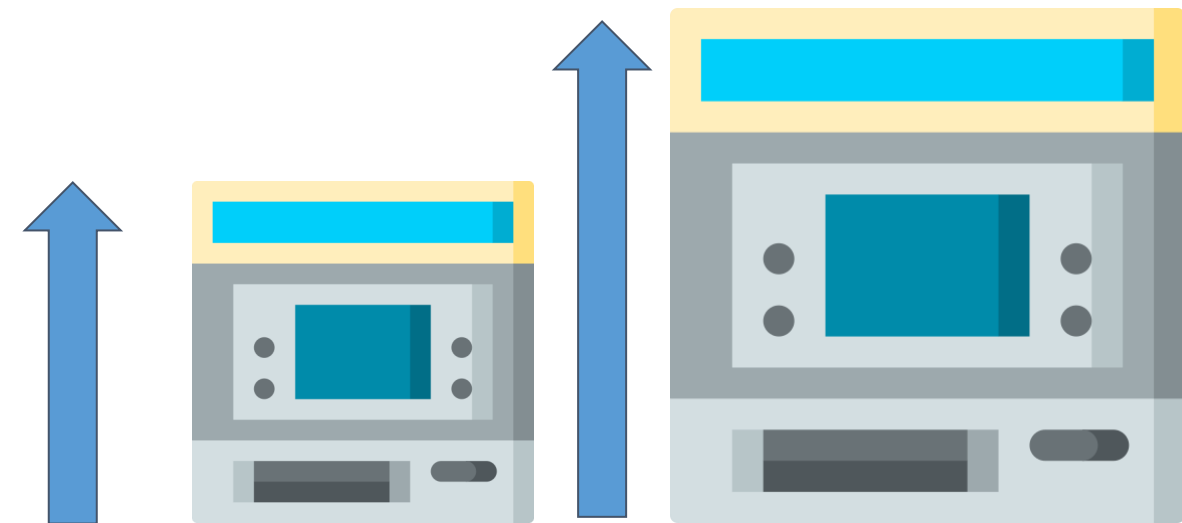
Note

Horizontal scaling implies distributed systems.

Vertical Scaling

Vertical Scaling means increasing the size of an instance for your application.

- Example: Changing the size of an ec2 instance from t2.small to t2.large.



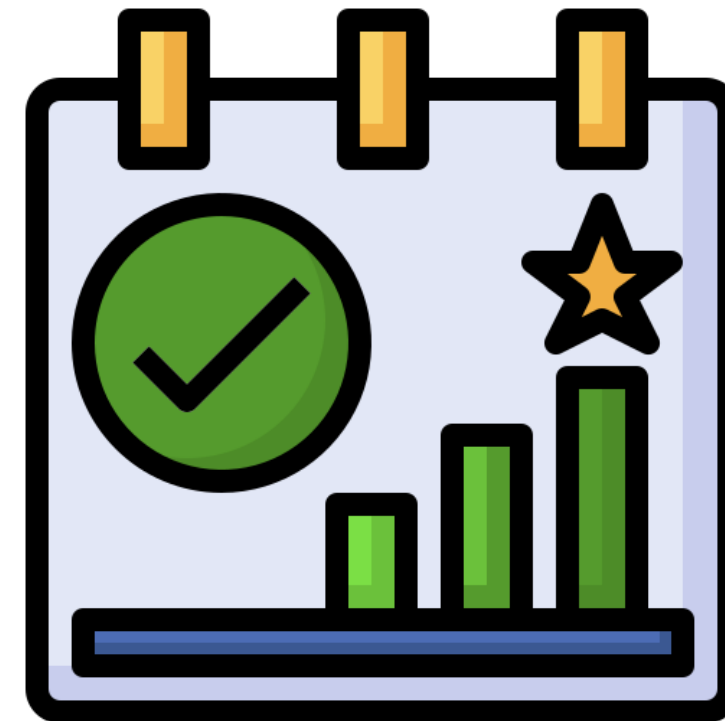
Note

Vertical scalability is very common for non distributed systems, such as a database.

High Availability

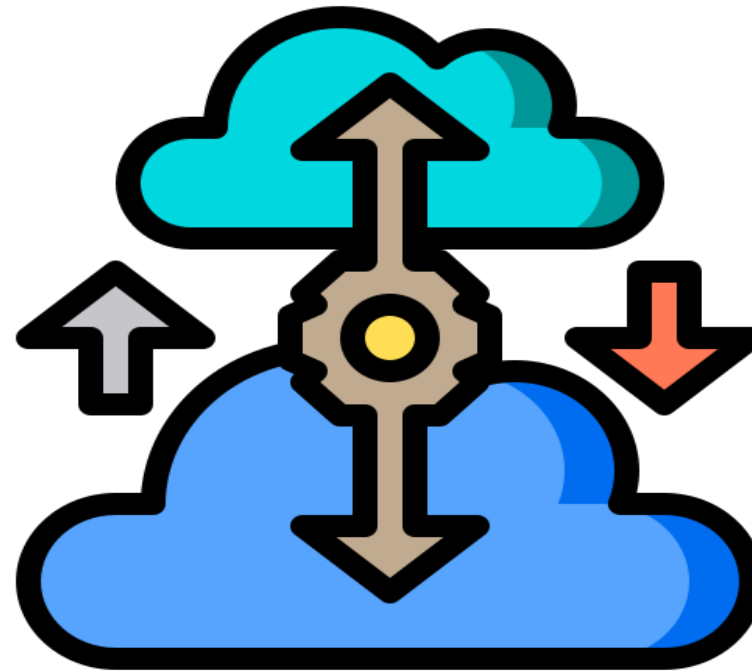
High availability refers to the capacity of a computing infrastructure to continue operating even if some of its components fail.

- In AWS, ensuring high availability means running an application/system in at least 2 Availability Zones.
- The goal of high availability is to survive a data center loss (disaster).
- High Availability usually goes hand in hand with horizontal scaling.



Elasticity

Elasticity refers to the cloud's ability to automatically expand or compress infrastructural resources in response to a sudden increase or decrease in demand, allowing the workload to be managed efficiently.



It's most typically found in public cloud services that charge by the minute, where the IT managers willingly pay for the time, they spend using the resources.

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

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

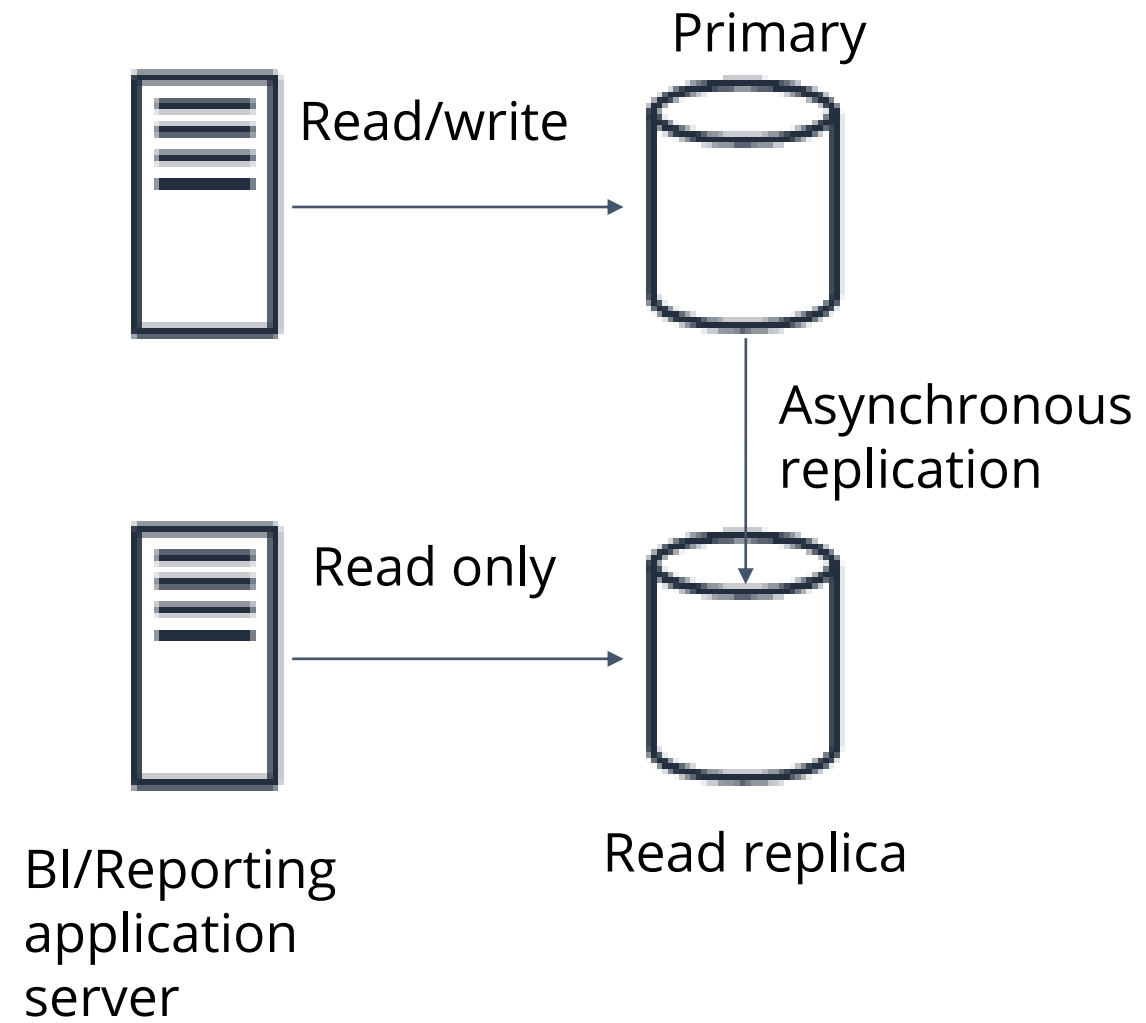
3

Data warehousing where business reporting queries have to be run against read replicas

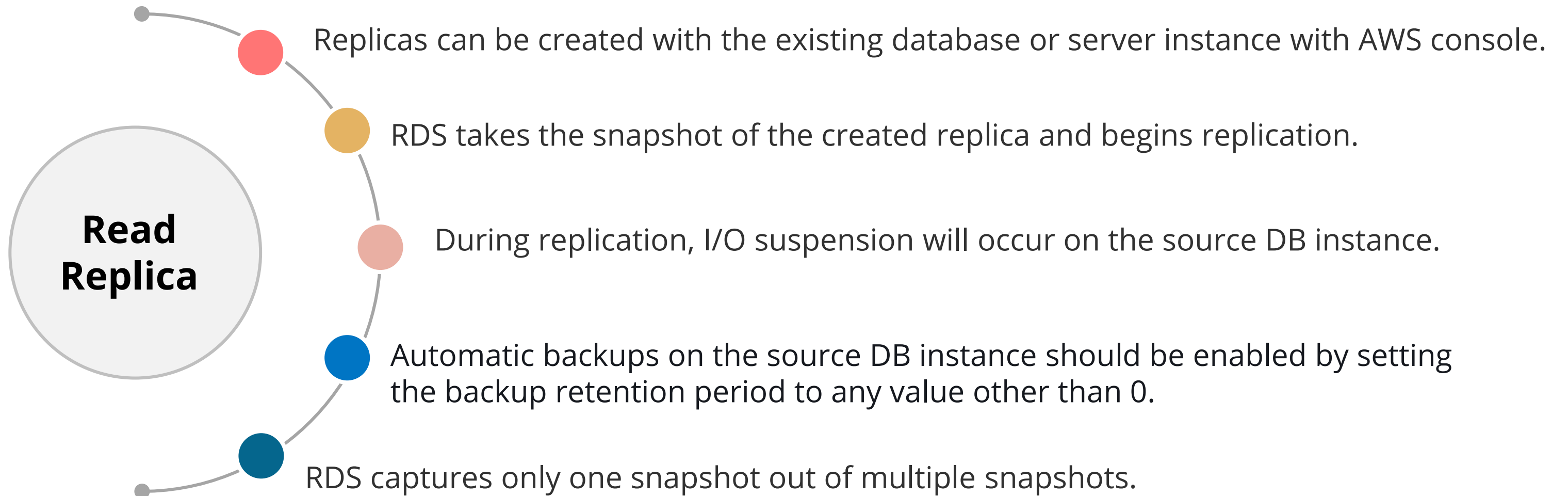
4

Implementing disaster recovery

Overview of RDS Read Replicas



Creating a Read Replica



Benefits of RDS Read Replicas



1 Enhanced Performance

2 Increased Availability

3 Designed for Security

Assisted Practice

Creating a Read Replica

Duration: 10 Min.

Problem Statement:

You've been requested to come up with a solution that will allow your organization to have one or more read-only copies of the database instance in the same AWS Region or a different AWS Region for high availability. You should also make sure that any changes made to the source database are copied asynchronously to your Read Replicas.

Assisted Practice: Guidelines

Steps to create a read replica are as follows:

1. Login to your AWS lab and open **Amazon RDS console**
2. Choose **Databases** in the navigation pane
3. Choose the database that you want to use as a source in the read replica
4. In the **Actions** tab, choose **Create read replica**
5. Choose your instance specification
6. Choose **other settings** and hit **Create read replica**

Assisted Practice

Creating a Read Replica in Multiple Regions

Duration: 10 Min.

Problem Statement:

Demonstrate creating Read Replica in multiple Region With Amazon RDS.

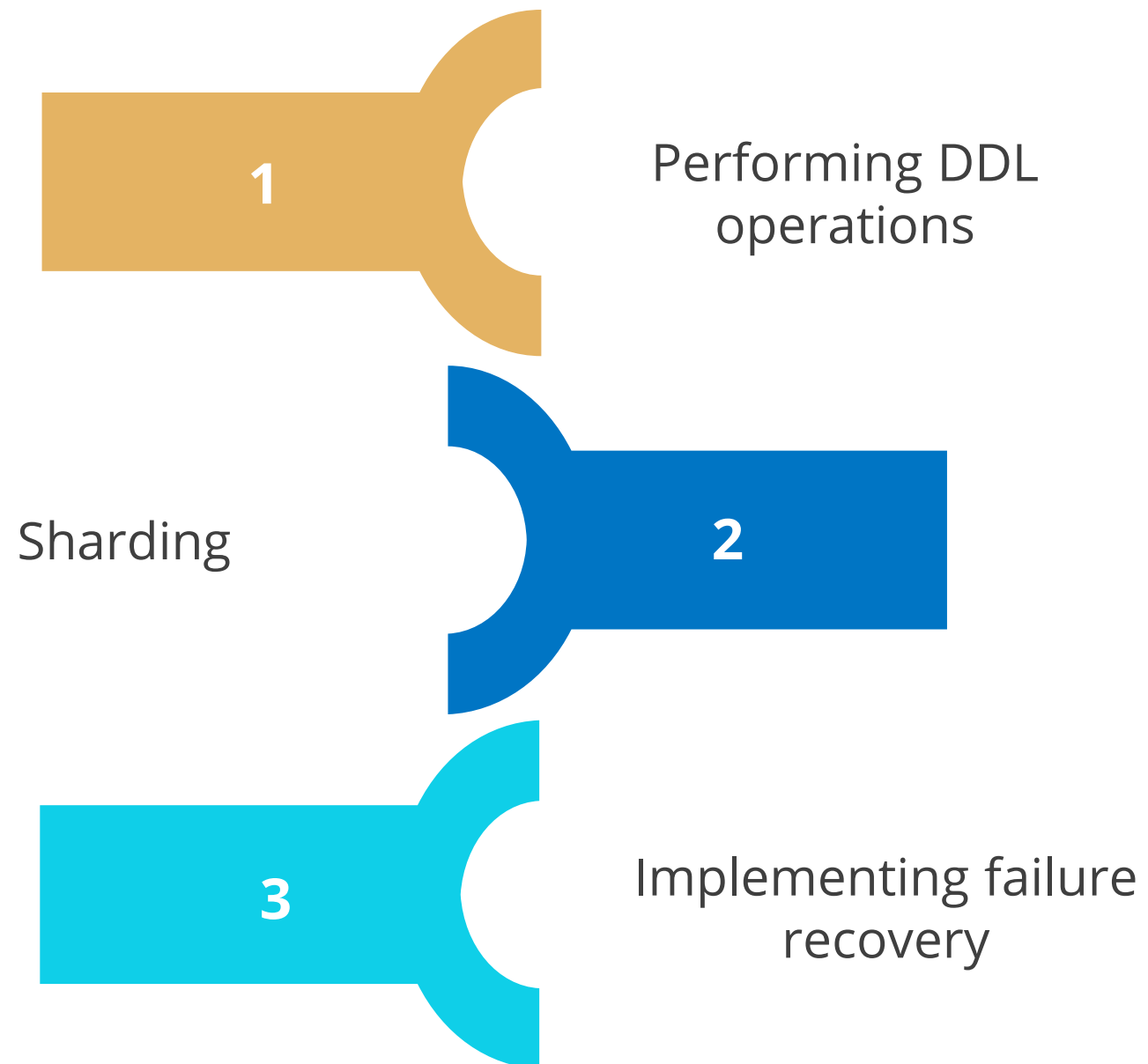
Assisted Practice: Guidelines

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

1. Login to your AWS lab and open **Amazon RDS console**
2. Choose **Databases** in the navigation pane
3. Choose the database that you want to use as a source in the read replica
4. In the other settings, under **Network and Security region**, choose a value for **Destination region** and one for **Destination DB subnet group**
5. Enable encryption through master key in **AWS KMS**
6. 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:



Amazon RDS and Cross-Region Replication

How It Works?

- 01** Begins configuring the source DB instance as a replication source and sets the status to *modifying*
- 02** Sets up the specified read replica in the destination AWS Region and sets the status to *creating*
- 03** Creates an automated DB snapshot of the source DB instance in the source AWS Region
- 04** Begins a cross-region snapshot copy for the initial data transfer
- 05** 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

Assisted Practice

Share an Encrypted RDS Snapshot

Duration: 10 Min.

Problem Statement:

You've been given the task of demonstrating how to share an Encrypted RDS Snapshot so that the authorized AWS accounts can copy it.

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**
2. Add the target account to a custom (non-default) KMS key
3. Copy the snapshot using the customer managed key, and then share the snapshot with the target account
4. Copy the shared DB snapshot from the target account

RDS Performance Insight

RDS Performance Insight

Performance Insights enhances existing Amazon RDS monitoring features by displaying database performance and troubleshooting any issues.

A user can visualize the database load and filter the load by:

- Waits
- SQL statements
- Hosts
- users

In the console create wizard for Amazon RDS engines, Performance Insights is enabled by default.

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)

Assisted Practice

Create an Aurora MySQL Database Cluster

Duration: 10 Min.

Problem Statement:

You've been asked to demonstrate Aurora MySQL Database Cluster, a fully managed MySQL-compatible relational database engine that combines the speed and reliability of high-end commercial databases with the ease and cost-effectiveness of open-source databases. You need to also ensure that routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair are being handled.

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**
2. Choose the AWS region in which you want to create a database
3. In the navigation pane, choose **Databases**
4. Choose **Create database** with **Easy Create** option enabled
5. Configure all the necessary settings and create the database

Assisted Practice

Connect to an Instance in a DB Cluster

Duration: 10 Min.

Problem Statement:

Demonstrate how to connect an Aurora MySQL database to an instance in the DB cluster.

Assisted Practice: Guidelines

Steps to connect to a database instance are as follows:

1. Login to your AWS lab and open **Amazon RDS console**
2. In the navigation pane, choose **Database** with the cluster name which is already created
3. Connect to MySQL using the commands
4. Configure all the necessary settings and connect to an 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

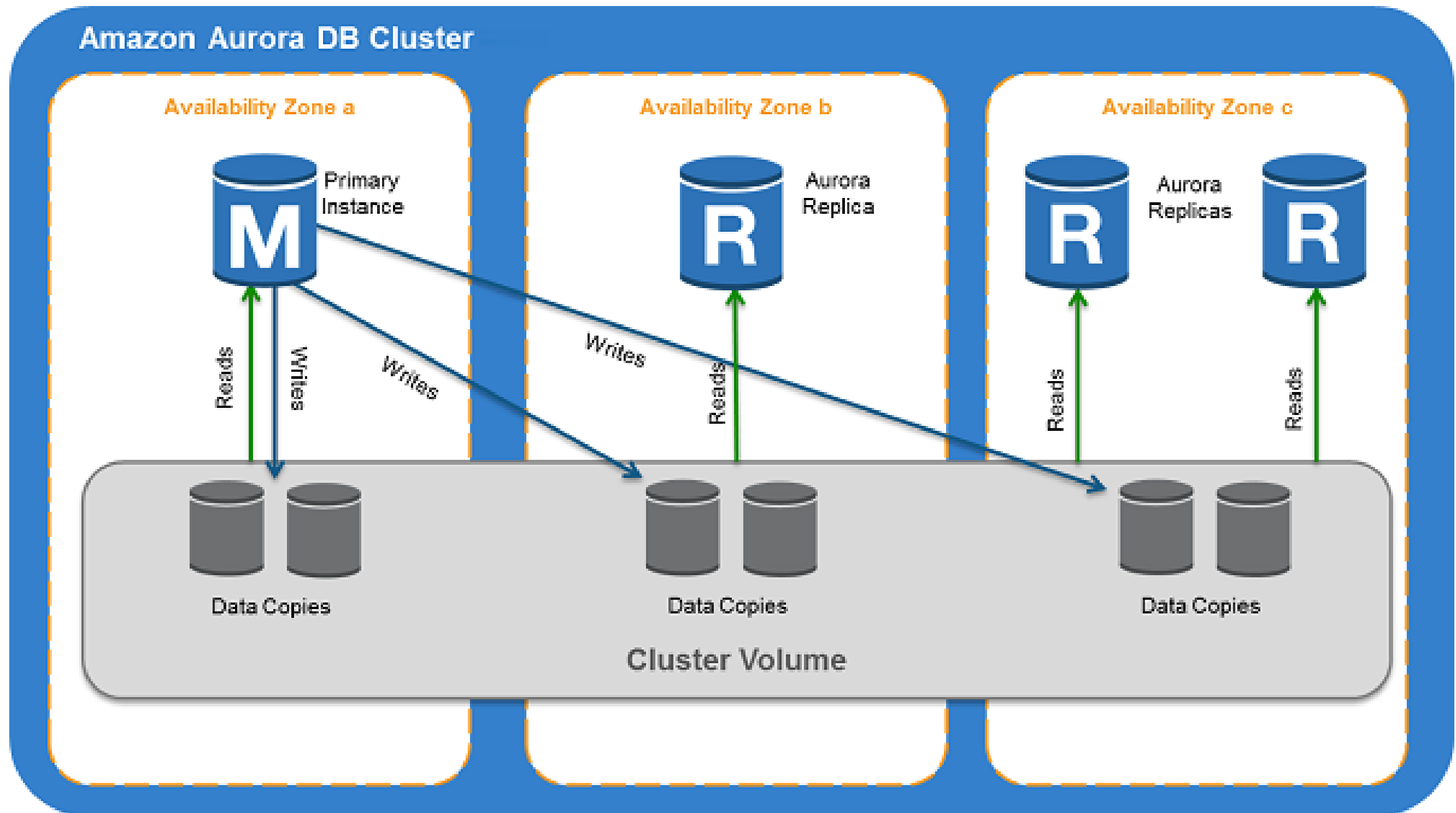
Amazon Aurora DB Clusters

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



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

Assisted Practice

Adding Aurora Replicas to a Database cluster

Duration: 20 Min.

Problem Statement:

Demonstrate how to add Aurora replicas to a database cluster.

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**.
2. In the navigation pane, choose **Databases**, and then select a DB cluster to create a DB instance.
3. For **Actions**, choose **Reader** and specify options for **Read Replica**.
4. Choose **Add Reader** to create the Aurora replica.

Assisted Practice

Modify an Aurora serverless DB Cluster

Duration: 20 Min.

Problem Statement:

Demonstrate how to Modify an Aurora serverless Database Cluster.

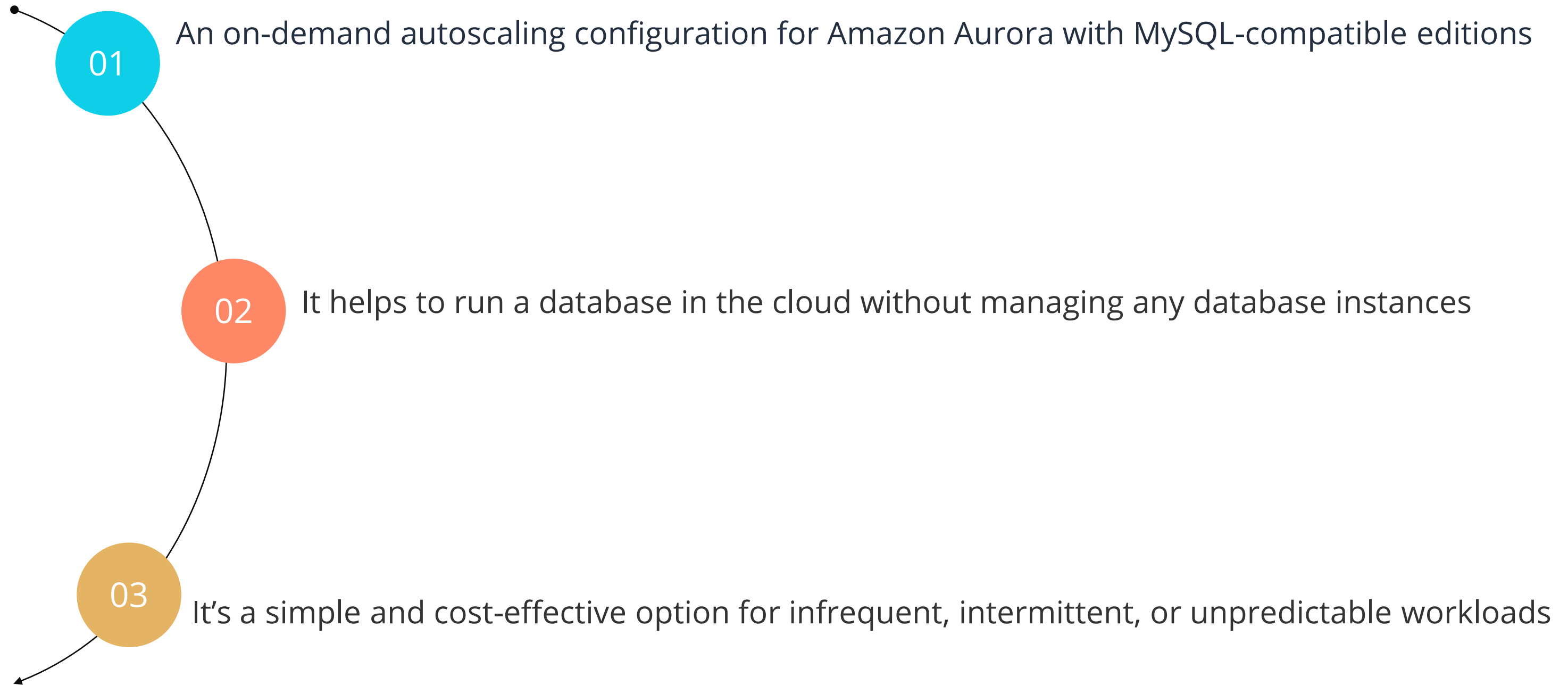
Assisted Practice: Guidelines

Steps to Modify the scaling configurations of an Aurora serverless DB cluster:

1. Open the Amazon RDS console at *<https://console.aws.amazon.com/rds/>*.
2. In the navigation pane, choose **Databases**.
3. Choose the Aurora serverless DB cluster that you want to modify.
4. For **Actions**, choose **Modify cluster**.

Aurora Serverless

Aurora Serverless



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

Elastic Load Balancers

Elastic Load Balancers

A load balancer helps in distributing requests on multiple servers or instances for efficient working of the application and mitigating response delays. The types of load balancers are given below:



Application load balancer

- It works at the application layer of the OSI model.



Network load balancer

- It works at the transport layer of the OSI model.



Classic load balancer

- This is a legacy load balancer.

Application Load Balancer

It is best suited for the application layer for load balancing HTTP and HTTPS traffic.

- It can be used for advanced routing and sending requests to determined servers.
- It can identify the required request for the determined server and can route the requests to the right servers if the sent request is not for the specified server.
- It performs all routing tasks using HTTP packets and headers.



Network Load Balancer

It is best suited for load balancing TCP traffic where high performance is required.

- It is capable of handling millions of requests per second.
- It maintains the lowest latency compared to all other load balancers.
- It is majorly used for production servers where low latency is of utmost importance.
- It is the most expensive load balancer.



Classic Load Balancer

It is a legacy load balancer and can be used on both application and transport layers.

- It provides basic features on layer 7 like X-forwarded and sticky sessions.
- It is rarely used and is not recommended for modern applications.
- It can be strictly used at layer 4 for an application that relies purely on TCP protocol.



Pre-Warming a Load Balancer

- Application load balancers scale automatically to adapt to your workload.
- This changes the IP addresses that the client connects to, as new ALBs are brought into service.
- A network load balancer creates a static IP address in each subnet.
- This keeps the firewall rules simple, as the client only needs to enable a single IP address for each subnet.
- This is done using AWS elastic IP addresses.
- Moreover, keeping an ALB behind an NLB reduces the task of choosing one or other LBs and gives the benefits of both the load balancers.



Load Balancer and Static IP

Pre-warming process is used to make a load balancer scale up if the traffic suddenly increases on the application.

- Example: An e-commerce company's marketing team plans to announce a sale on a public holiday and estimates that there will be five times more traffic on the website.
- To avoid any downtime in this sudden increase in the number of requests, AWS can pre-warm the ELB and configure it to the appropriate level of capacity required to handle requests.
- AWS needs to know the following data to pre-warm the load balancers:
 - Start and end dates of the high-performance capacity
 - Expected request rate per second
 - Size of a typical request



ELB Error Messages

ELB Error Messages

4XX and 5XX are the major error code that can occur in ELB operations.

- Any unsuccessful request generates 4XX and 5XX errors.
- 4XX error message indicates that there is an error on the client side.
- 5XX error message indicates the issue on the server side.



ELB Error Messages

- 400 indicates that it is a malformed request such as an incorrect header and is not per HTTP and HTTPS standards.
- 401 indicates that the user doesn't have access to the webpage.
- 403 indicates that the request is forbidden and the url is blocked.
- 460 indicates the client's timeout period is short and the load balancer doesn't have time to respond.
- 463 indicates that the load balancer has received an X-Forwarded-For request header with more than thirty IP addresses.



ELB Error Messages

- 500 indicates an internal server error such as a configuration issue with ELB.
- 502 indicates bad gateway in cases when an application server has closed the connection or sent a malformed response.
- 503 indicates that the service is unavailable which means there are no registered target or web servers.
- 504 indicates gateway timeout which means the application is not responding due issues with web servers or databases.
- 561 indicates that the load balancer is not getting a response from the ID provider to authenticate a user.



ELB Cloudwatch Metrics

ELB publishes metrics to Cloudwatch for the load balancer and also for the backend instances.

- The metrics help to verify a system's performance.
- Metrics are gathered in an interval of sixty seconds.
- User can also create a Cloudwatch alarm for a specific action.
- Example: User can create a Cloudwatch alarm to send an email if the metrics reach the limit.



ELB Cloudwatch Metrics

Cloudwatch metrics can be categorized based on the operations:

Overall Health and Performance Metrics

Overall Health:

- It checks the overall performance and status of the system.
- It includes issues like:
 1. **BackendConnectionError:** Number of unsuccessful backend connections to instances
 1. **HealthyHostCount:** Number of healthy registered instances
 1. **UnhealthyHostCount:** Number of unhealthy host count with issues in services
 1. **HTTPCode_Backend_2XX_4XX_5XX**



ELB Cloudwatch Metrics

Performance Metrics:

They deal with checks like:

1. **Latency:** Number of seconds taken for a registered instance to respond or connect
1. **RequestCount:** Number of requests completed or connections made during a specified interval
1. **SurgeQueueLength:**
 - Number of pending requests
 - It's for classic load balancers and has the maximum queue size of 1024
 - Any additional request is rejected
1. **SpilloverCount:**
 - Number of requests rejected
 - This metric is for classic load balancers



Assisted Practice

Deploying an Application Load Balancer

Duration: 10 Min.

Problem Statement:

You are given a project to deploy an Application Load Balancer to distribute your incoming traffic across multiple targets in one or more Availability Zones.

Assisted Practice: Guidelines

Steps to deploy an application load balancer:

1. Select a load balancer
2. Select a security group
3. Configure targets
4. Select an instance

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

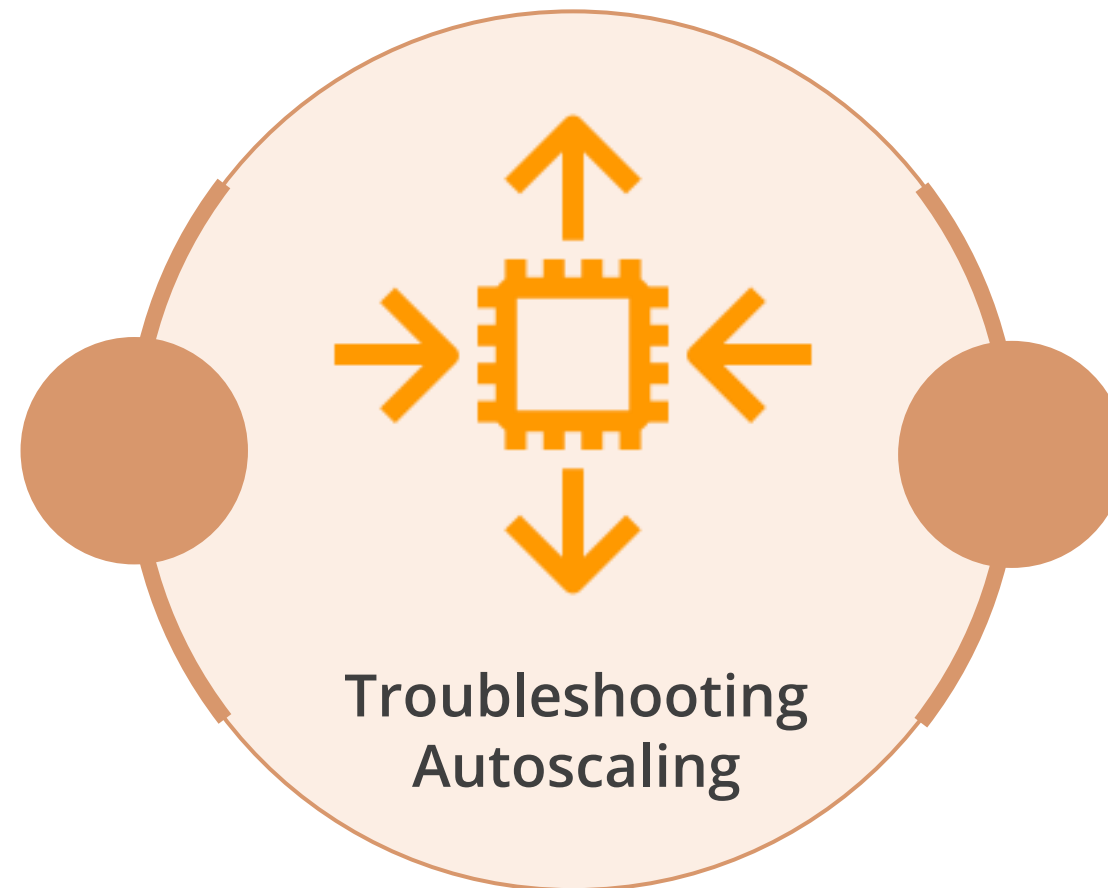
Autoscaling service is not enabled on your account

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.

Elasticache

ElastiCache: Overview

- Amazon ElastiCache is a distributed cache environment which allows to set up, run, and scale popular in-memory data stores on cloud.
- It improves the performance of existing databases by retrieving information from managed in-memory cache.
- It is used in many real-time use cases such as gaming, queuing, and real-time analytics.
- It has fully managed Redis and Memcached engines for in-memory caching.
- To monitor caching engines, one should look at CPU utilization, swap usage, evictions, and concurrent connections.



Amazon FSx

Amazon FSx for Windows File Server

It is a fully managed Windows file server that is backed by a native Windows file system.

- Integrated with Microsoft Active Directory
- Supports SMB protocol and Windows NTFS
- Accessible from AWS or your own on-premises infrastructure



Amazon FSx for Lustre

Amazon FSx for Lustre makes launching and running the popular, high-performance Lustre file system simple and cost-effective.

- Used for applications such as machine learning, high-performance computing (HPC), video processing, and financial modelling where speed matters
- Scales up to 100s GB/s, millions of IOPS, sub-ms latencies
- Uses the existing Linux application without any changes as it is POSIX-compliant



Note

The name Lustre is derived from “Linux” and “cluster”.

Elastic File System (EFS)

Elastic File System

Amazon Elastic File System (Amazon EFS) provides a simple, serverless, set-and-forget elastic file system.

- It enables a user to share file data without provisioning or managing storage.
- It is built to scale on demand to petabytes without disrupting applications.
- A user can grow and shrink a file system automatically, eliminating the need to provision and manage capacity to accommodate growth.



Note

A user can provide highly available, fault tolerant file storage for applications running on AWS using Amazon EFS or Amazon FSx.

Elastic File System

Amazon EFS offers the choice of creating file systems using two different classes.

Standard

Standard storage classes store data within and across multiple availability zones (AZ).

One Zone storage classes

One Zone storage classes store data redundantly within a single AZ, at a 47% lower price compared to file systems using Standard storage classes, for workloads that don't require multi-AZ resilience.

Assisted Practice

Working with EFS

Duration: 10 Min.

Problem Statement:

You are given a project to create a file system that provides a simple, serverless, set-and-forget elastic file system for use with AWS Cloud services and on-premises resources.

Assisted Practice: Guidelines

Steps to customize a file system and access it using a specific EC2 instance:

1. AWS EC2 service is enabled in your practice lab
2. Go to the Amazon dashboard
3. Go to AWS Console and select EFS service
4. Click on **Customize**
5. Fill in the required settings and proceed.

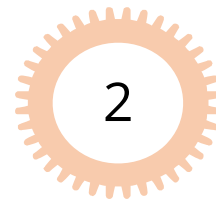
Elastic IP Address

Elastic IP Address

An *Elastic IP address* is a static IPv4 address allocated to a user's AWS account and is available until the user releases it.



A user can mask the failure of an instance or software by rapidly remapping the address to another instance.



A user can specify the Elastic IP address in a DNS record for a domain, so that the domain points to an instance launched in his AWS account.

Disaster Recovery

AWS DataSync

AWS DataSync is an online data transfer service that makes moving data easier, faster, and more automated between:

- On-premises storage systems and AWS storage services
- AWS storage services



Note

DataSync can copy data between Network File System (NFS), Amazon EFS, Amazon S3, and Amazon FSx.

AWS DataSync

Some of the most common use cases for AWS DataSync:

Data migration



Data protection



Archiving cold data

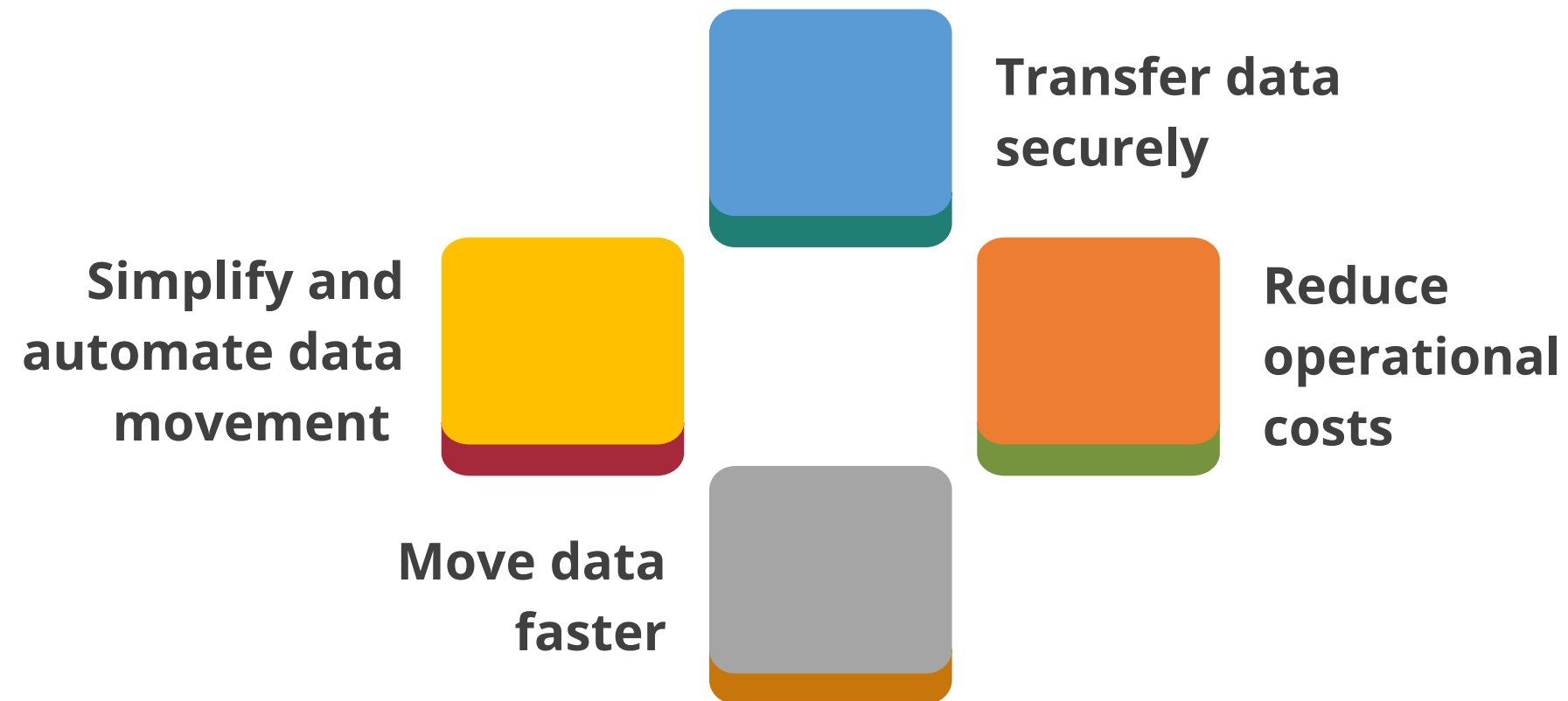


**Data movement for
timely in-cloud
processing**



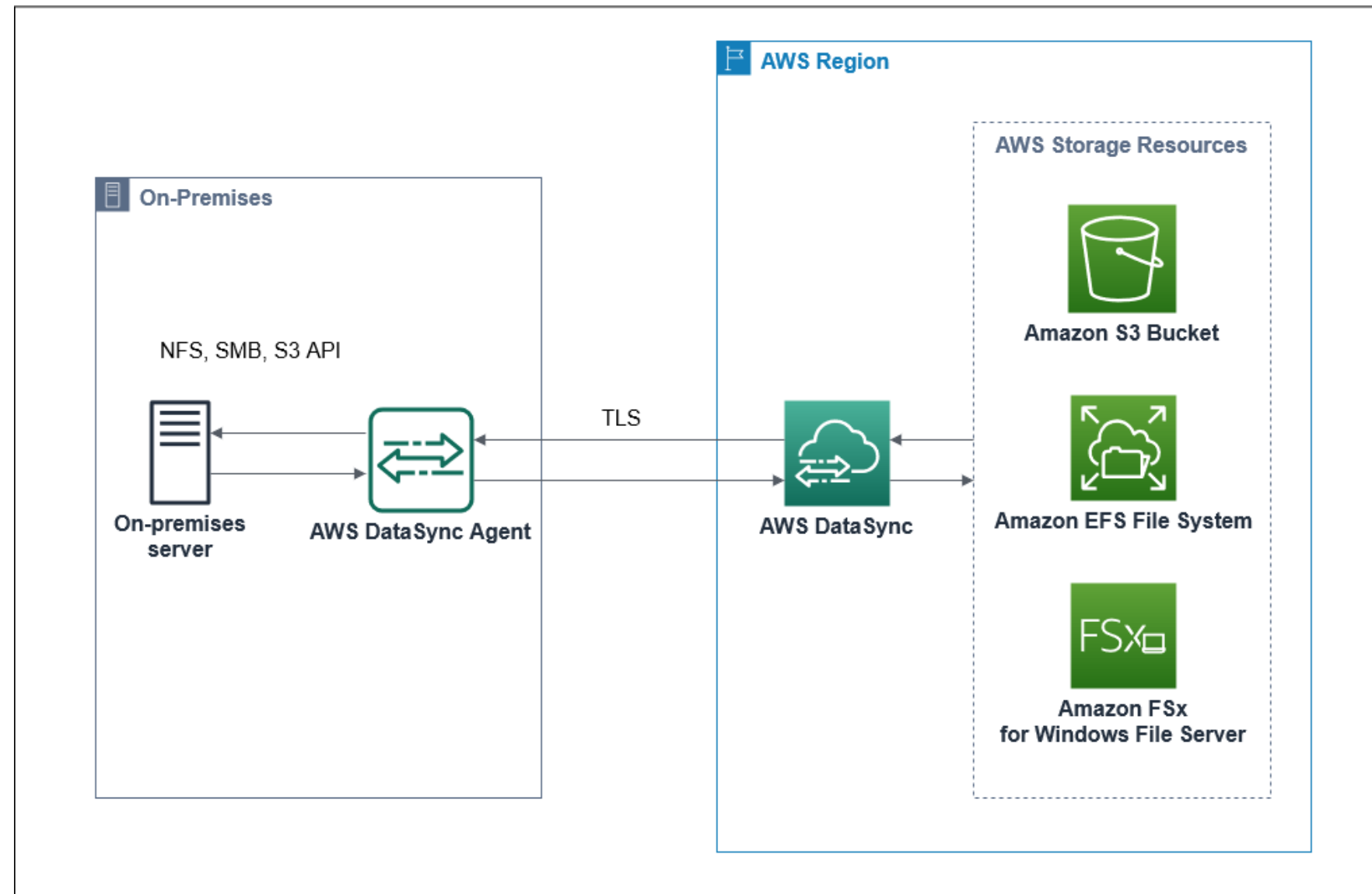
AWS DataSync

The following are some of the advantages of using AWS DataSync:



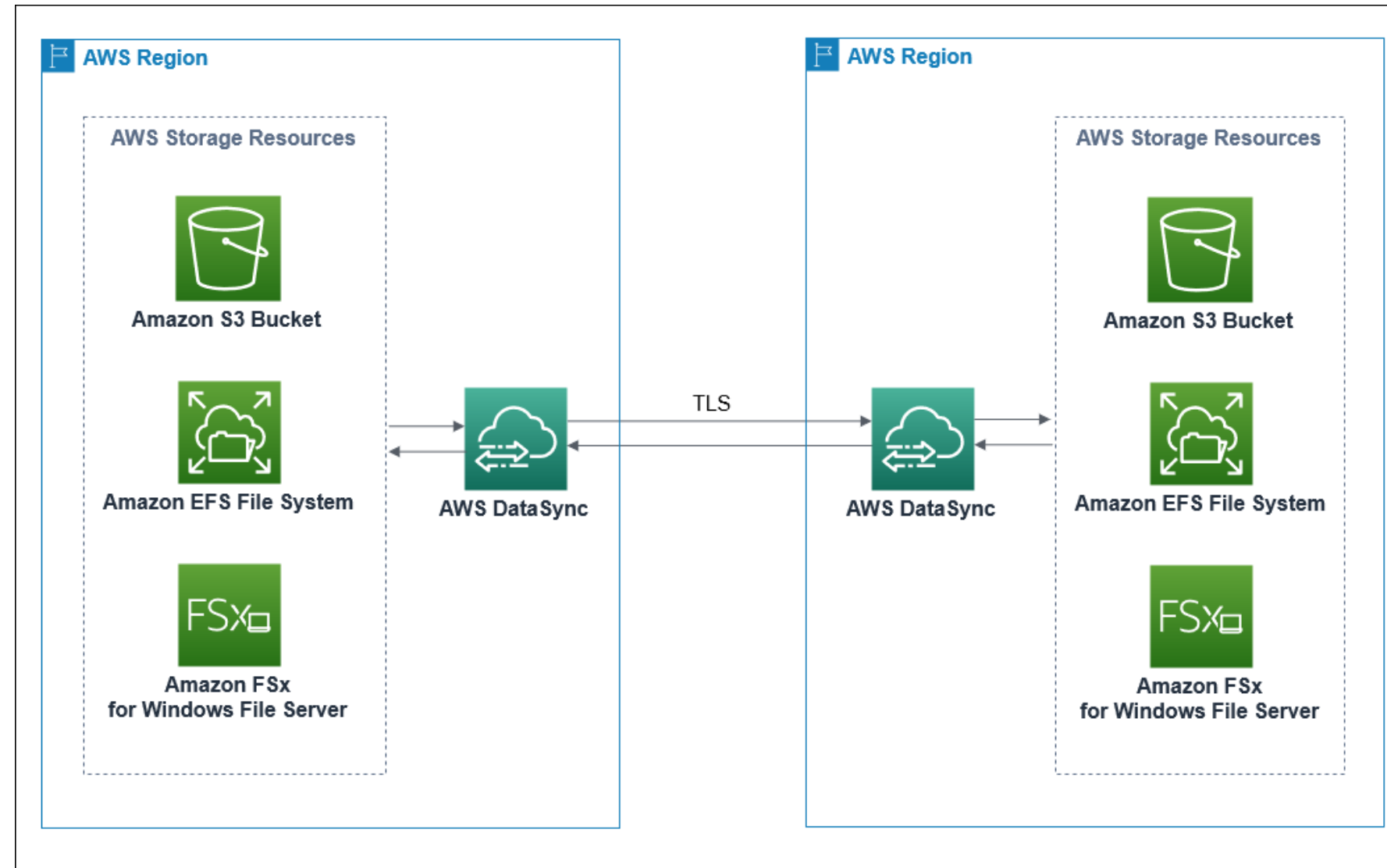
DataSync Architecture

The DataSync architecture for transferring files between self-managed storage and AWS services is depicted in the diagram below:



DataSync Architecture

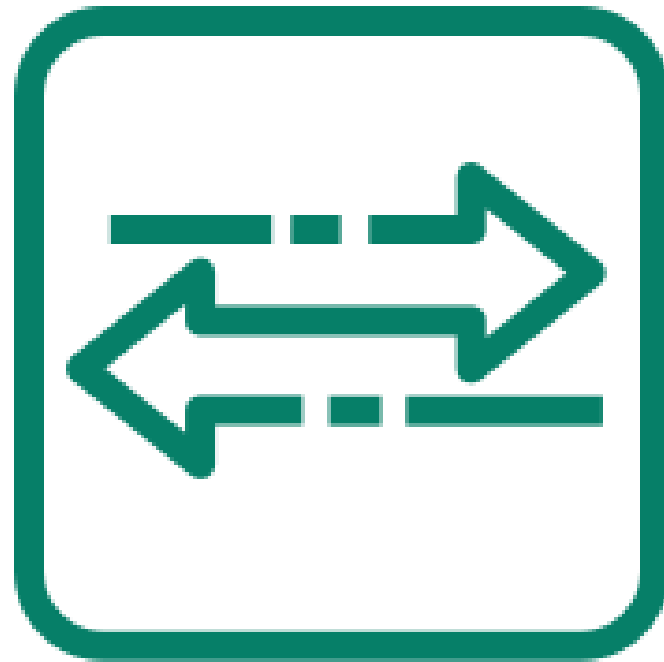
The DataSync architecture for transferring files between AWS services within the same account is depicted in the diagram below:



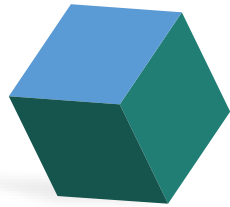
This architecture applies to both in-region and cross-region transfers.

DataSync EC2 Agent

A user can use DataSync to transfer data across AWS services in separate accounts, or between self-managed file systems in AWS and Amazon S3, by deploying the DataSync EC2 agent in an AWS Region.



Components of DataSync



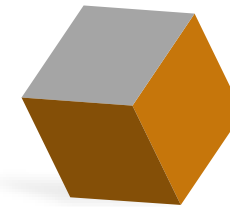
Agent

A virtual machine (VM) that is used to read or write data from or to a self-managed location



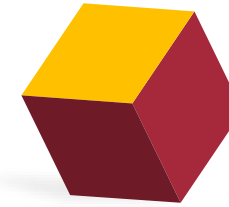
Location

Any location that is used in the data transfer as a source or destination



Task

It consists of a source location and a destination location, as well as data transfer configuration



Task Execution

The start time, end time, bytes written, and status of an individual task run

AWS Backup

AWS Backup is a fully managed data backup solution that simplifies the centralization and automates data protection across AWS services, both in the cloud and on premises.













A user can configure backup policies and monitor activity for your AWS resources in one place.

AWS Backup: Supported AWS Resources and Third-Party Applications

Supported Service	Supported Resource
Amazon FSx	Amazon FSx file systems
Amazon Elastic File System (Amazon EFS)	Amazon EFS file systems
Amazon DynamoDB	DynamoDB tables
Amazon Elastic Compute Cloud (Amazon EC2)	Amazon EC2 instances (excluding store-backed instances)
Windows VSS (Volume Shadow Copy Service)	Windows VSS-supported applications (including Windows Server and Microsoft SQL Server) on Amazon EC2
Amazon Elastic Block Store (Amazon EBS)	Amazon EBS volumes
Amazon Relational Database Service (Amazon RDS)	Amazon RDS databases (including all database engines)
Amazon Aurora	Aurora clusters
AWS Storage Gateway (Volume Gateway)	AWS Storage Gateway volumes

Features of AWS Backup

AWS Backup offers the following features across all supported AWS services and third-party applications:

- | | |
|---|---|
|  Centralized backup management |  Cross-Region backup |
|  Policy-based backup |  Cross-account management and cross-account backup |
|  Tag-based backup policies |  Backup activity monitoring |
|  Lifecycle management policies |  Secure your data in backup vaults |
|  Incremental backups |  Satisfy compliance obligations |

Plan for Disaster Recovery (DR)

A disaster recovery (DR) strategy is defined to meet objectives by choosing a strategy such as backup and restore, active/passive (pilot light or warm standby), or active/active.

A user should:

- Set objectives based on business needs.
- Implement a strategy to meet these objectives, considering the locations and function of workload resources and data.

Recovery Time Objective (RTO) and Recovery Point Objective (RPO) are the objectives for restoration of any workload.

RTO and RPO

Recovery Time Objective (RTO):

The organization defines the RTO as the maximum allowed time between a service interruption and its resumption. When service is unavailable, this controls what is considered an acceptable time window.

Recovery Point Objective (RPO):

The organization defines the RPO as the maximum amount of tolerable time since the last data recovery point. This sets how much data loss is tolerated between the last recovery point and the service disruption.

Amazon Data Lifecycle Manager

It can be used to automate the creation, retention, and deletion of EBS snapshots and EBS-backed AMIs.

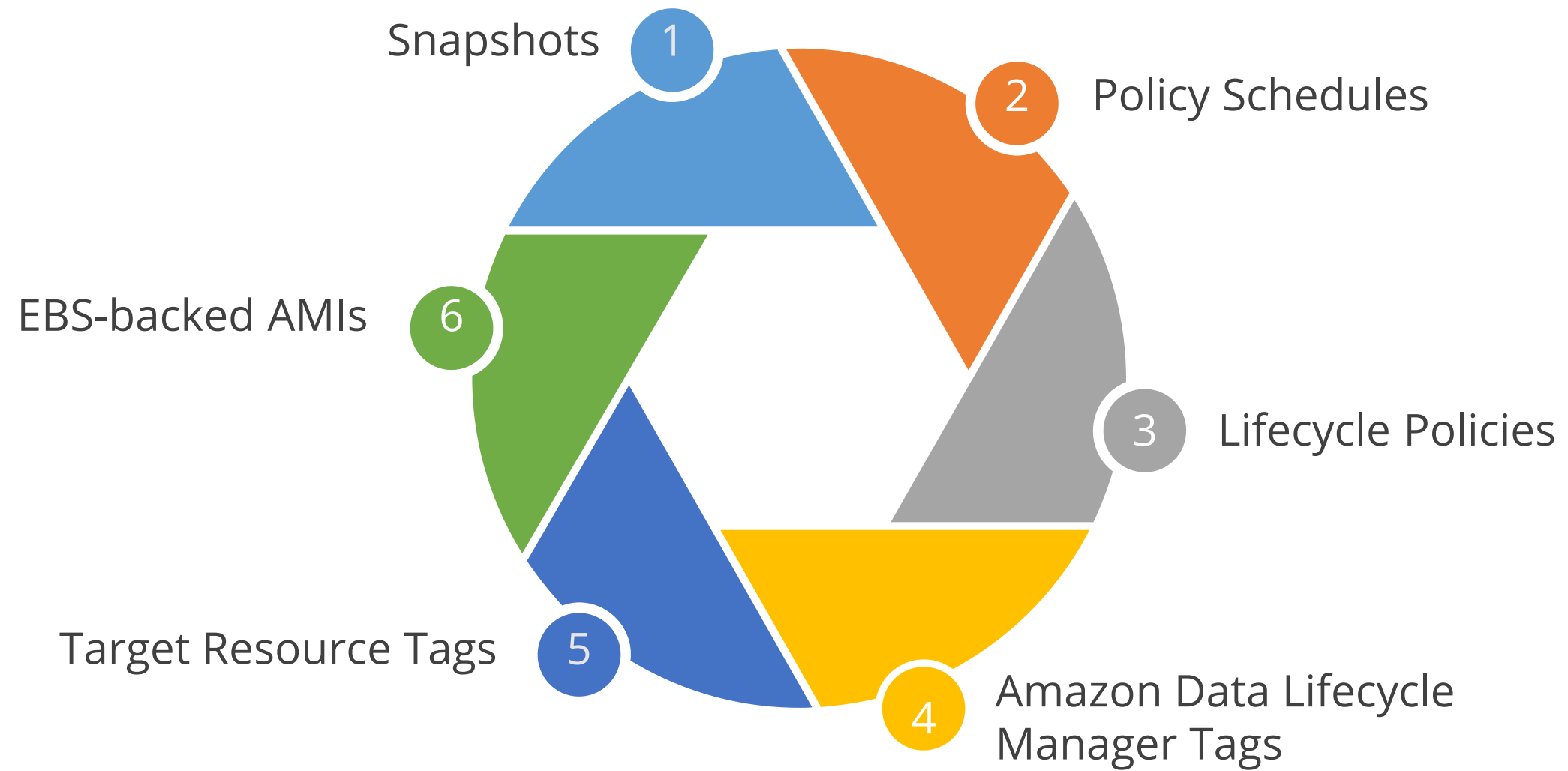
Automation of snapshot and AMI management helps a user to:

- Maintain a regular backup
- Create standardized AMIs
- Keep backups
- Reduce storage costs
- Create backup plans for disaster recovery

Amazon Data Lifecycle Manager provides a complete backup solution for Amazon EC2 instances and individual EBS volumes at no additional cost when combined with the monitoring features of Amazon CloudWatch Events and AWS CloudTrail.

Amazon Data Lifecycle Manager

The key components of Amazon Data Lifecycle Manager are:



Key Takeaways

- ❶ RDS provides cost-efficient and resizable capacity while automating time-consuming administrative tasks.
- ❷ Read replicas let you elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.
- ❸ Aurora is five times faster than the standard MySQL databases and three times faster than the standard PostgreSQL databases.
- ❹ AWS Backup is a fully managed data backup solution that simplifies the centralization and automates data protection across AWS services.



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.

