

# Lecture 6: AWS SQL Database

**Maharishi International University**

**Department of Computer Science**

**M.S. Thao Huy Vu**

# Maharishi International University - Fairfield, Iowa



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi International University (MIU).

# Agenda

- **Database Services in AWS**
- **Amazon RDS (Relational Database Service)**
- **Amazon Aurora**
- **Best Practices and Considerations**

# Database Services in AWS

- A managed service provides database solutions tailored to different needs – from relational databases to NoSQL, in-memory, and graph databases
- Flexibility
- Durability and Availability
- Cost-efficiency
- Security

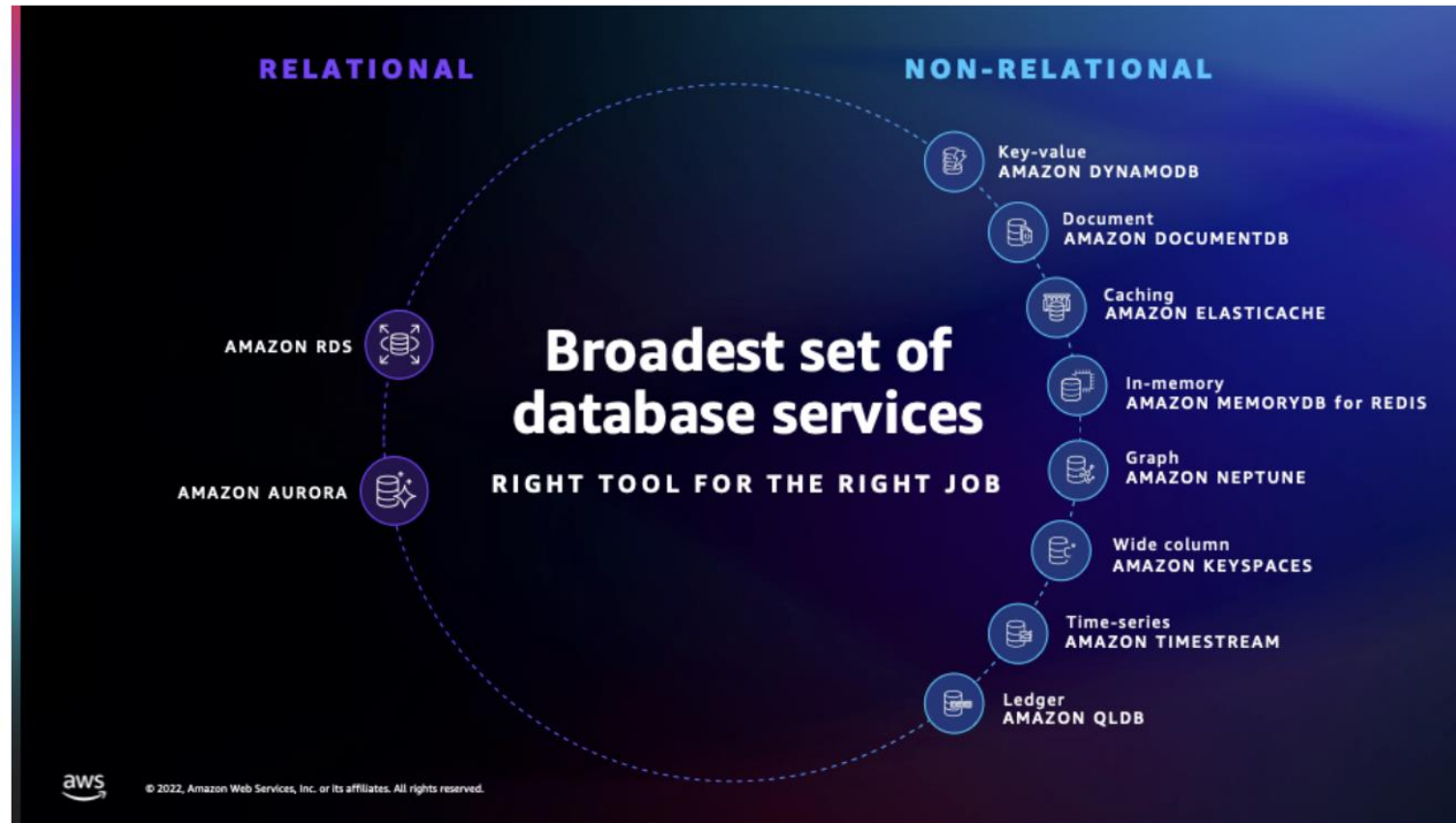
# Database Solutions

Feature	AWS Database (Managed)	Self-Hosted Database
Setup and Management	Fully managed by AWS, with automated setup, updates, and backups.	Requires manual installation, configuration, and maintenance.
Database Options	Offers managed SQL (e.g., RDS) and NoSQL (e.g., DynamoDB).	Supports SQL (e.g., MySQL, PostgreSQL) and NoSQL databases on custom servers.
Scalability	Automatic scaling (vertical and horizontal) with minimal effort.	Scaling requires manual intervention and infrastructure management.
Availability	Built-in high availability (e.g., Multi-AZ for RDS).	Must configure replication and failover manually.
Cost	Pay-as-you-go pricing; no upfront hardware costs.	Potentially lower hardware costs but higher operational overhead.
Control	Limited access to underlying infrastructure and database engine.	Full control over the OS, database engine, and configurations.

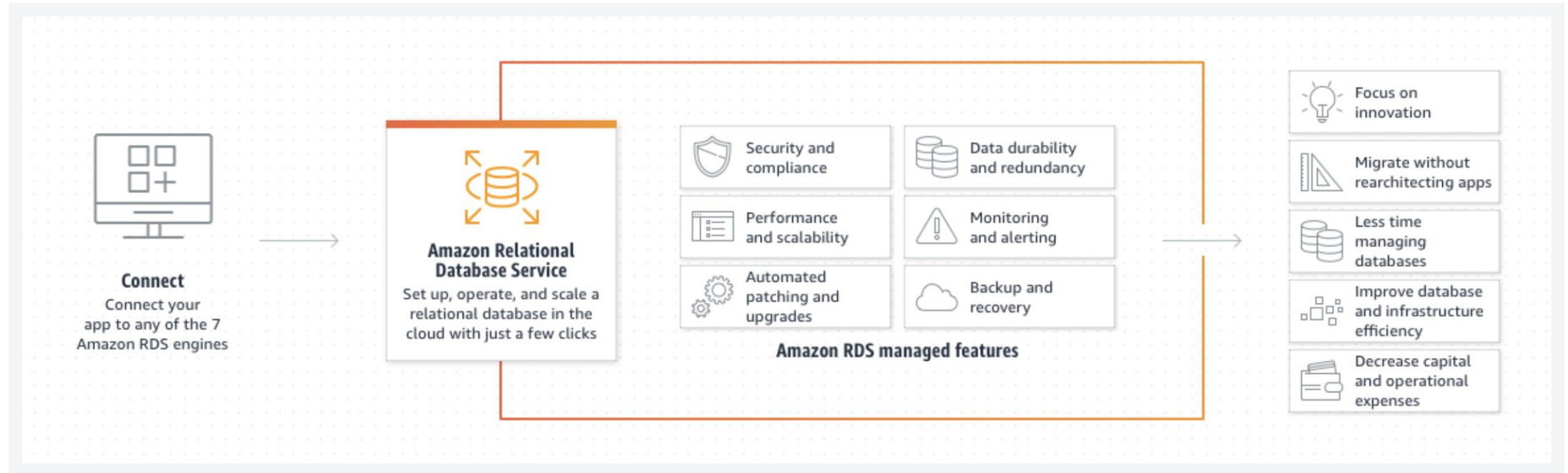
# Database Services in AWS

- Rational Databases: Amazon RDS, Amazon Aurora
- NoSQL Databases: DynamoDB, DocumentDB
- In-memory Databases: Amazon ElasticCache, Redis, Memcached
- Graph Databases: Amazon Neptune

# Database Services in AWS



# Amazon RDS





# Amazon RDS

- **Database Engines:** Supports popular engines like **MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, and Aurora.**
- **Managed Service:** Automates database setup, patching, backups, and scaling.
- **Scalability:**
  - **Vertical scaling:** Easily resize compute and memory by changing instance type.
  - **Horizontal scaling:** Add **Read Replicas** to scale read-heavy workloads and improve performance.
- **Availability and Reliability:**
- **Multi-AZ deployments:**
  - Provide high availability with **synchronous standby instances** in different Availability Zones.
  - **Automated failover:** Ensures minimal downtime during outages or maintenance.
- **Performance:**
  - **Optimized SSD-backed storage** (gp3, io1) with support for **Provisioned IOPS.**
  - Designed for **high throughput** and **low latency** workloads.
  - Supports integration with **Amazon ElastiCache** for enhanced caching.

# Amazon RDS

- **Security**

- **Encryption** at rest (using KMS) and in transit (SSL/TLS).
- **IAM integration** for fine-grained access control.
- **Network isolation** via Amazon VPC.
- **Supports compliance** with standards like **HIPAA**, **GDPR**, and **PCI DSS**.

- **Automated Backups with Point-in-Time Recovery (PITR)**

- AWS automatically takes a **daily full snapshot** of your RDS database during your preferred backup window.
- **Transaction logs** are continuously backed up, enabling **restores to any specific second** within your retention period (up to **35 days**).
- The entire process is **fully managed** by AWS, ensuring continuous protection with minimal overhead.

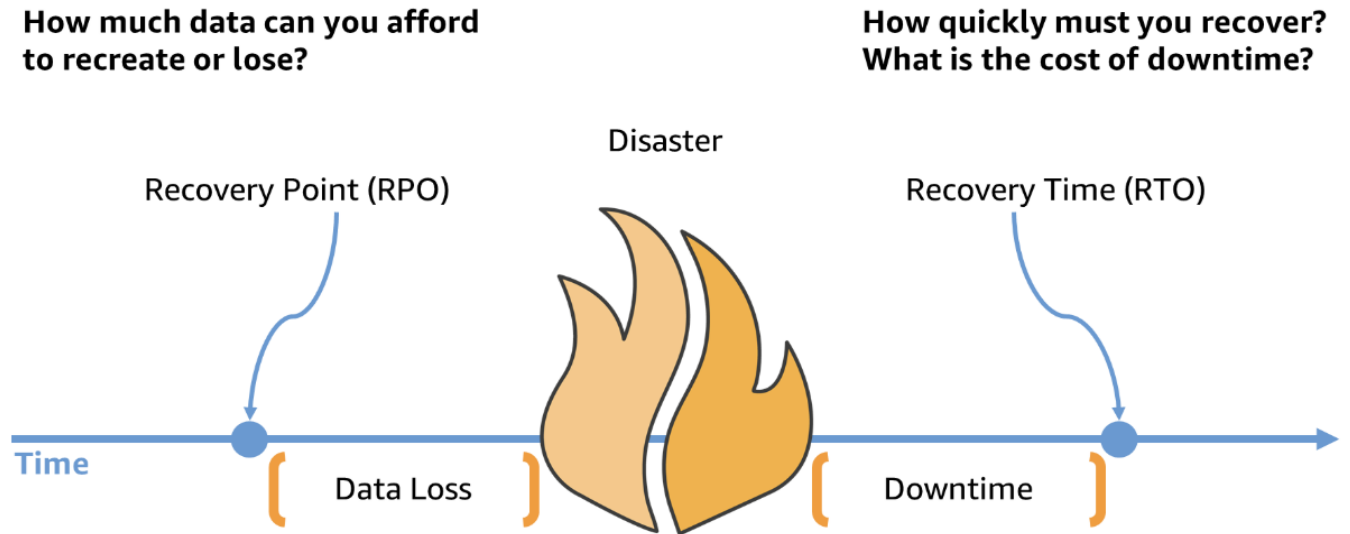
# RTO & RPO in RDS

- **Recovery Point Objective (RPO)**

- Maximum Acceptable **data loss**.
- In RDS, RPO is **very low**, minimizing data loss.

- **Recovery Time Objective (RTO)**

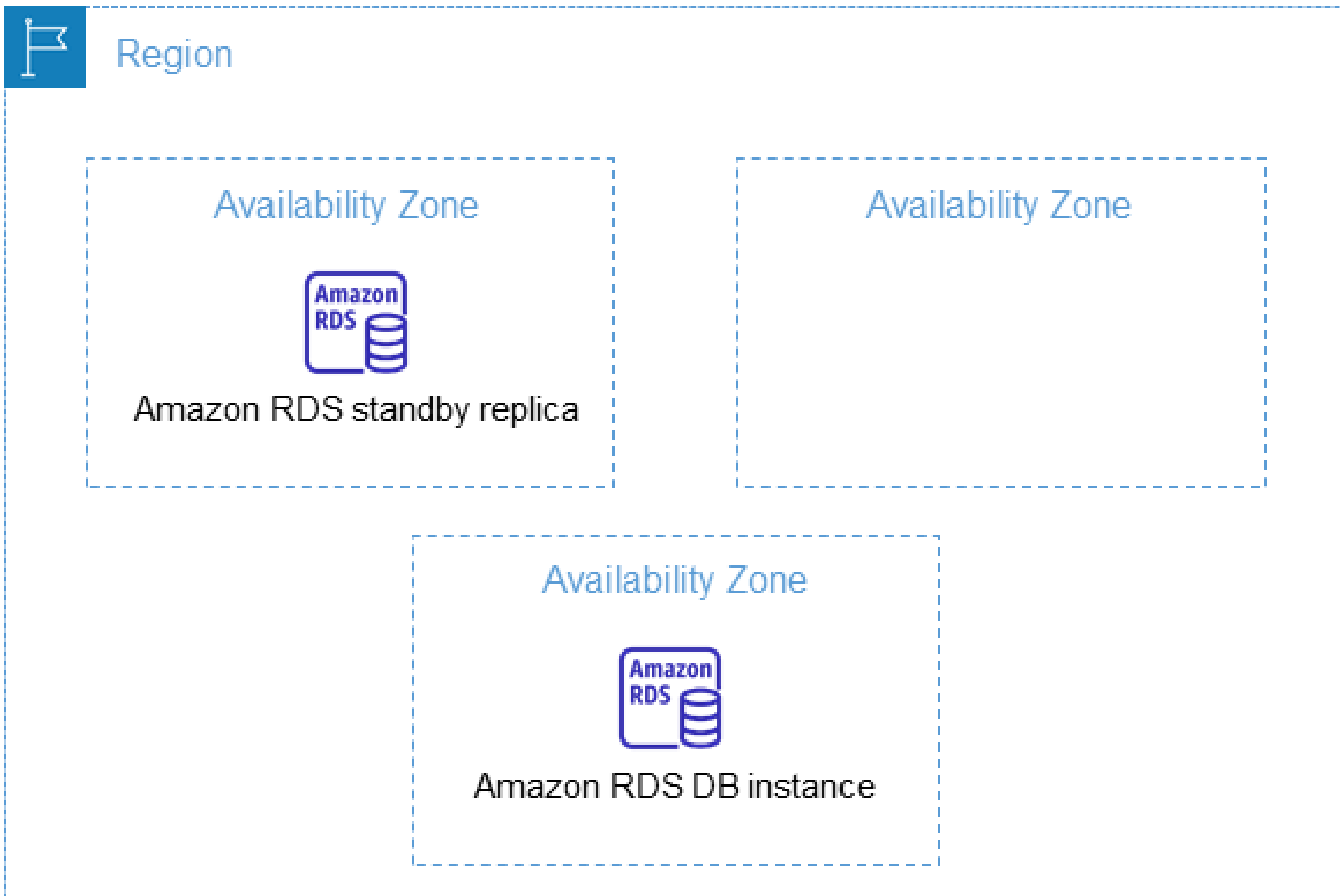
- Maximum acceptable **downtime**.
- In RDS, RTO is typically a **few minutes** during failover.



*Image1: Data loss is measured from most recent backup to the point of disaster. Downtime is measured from the point of disaster until fully recovered and available for service.*

# RDS Multi-AZ

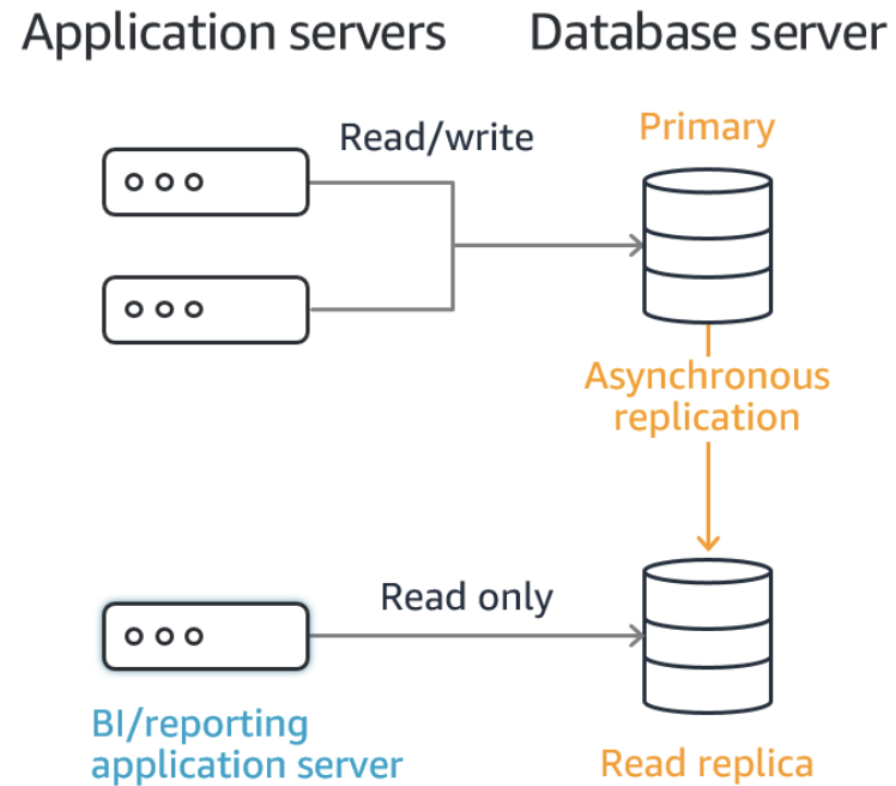
- Multi-AZ deployment improves database availability by maintaining a standby Replica in a different Availability Zone (AZ).
- **For RDS (Non-Aurora):**
  - Creates a **synchronous** standby instance in another AZ.
  - **Standby Instance:** is not readable – only for failover.
- **Failover Process:**
  - In case of an outage, the standby instance or read replica is promoted to primary
  - Failover typically completes within **~1–2 minutes**.
  - The database API remains the same during failover, so applications continues without changes.



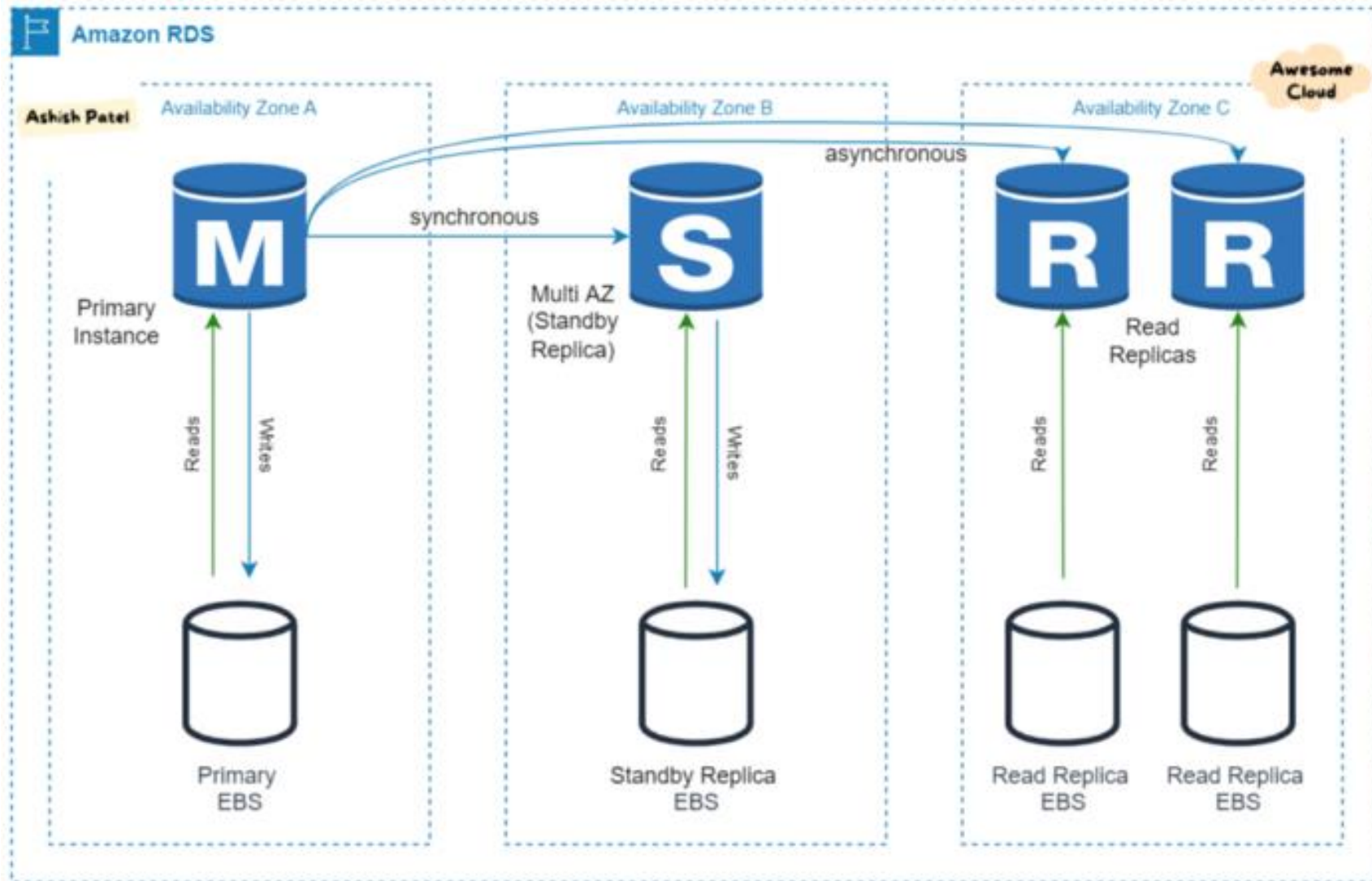
# RDS Read Replica

- Amazon RDS Read Replicas enhance database performance and scalability by enabling read-heavy workloads to be offloaded from the primary database.
- **Benefits:**
  - **Enhanced Performance:** Offloads read operations to replicas.
  - **Scalability:** Supports high read throughput by distributing read traffic.
  - **Redundancy:** Adds redundancy for disaster recovery.
- **Replication:**
  - **Asynchronous:** Ensures eventual consistency for reads.
  - **Secure:** Transactions during replication are encrypted.
- **Flexibility:**
  - Replicas can be **promoted** to standalone databases during failover or migration.
  - Improve availability: Read replicas can be deployed in **Multi-AZ**.

# RDS Read replica



# Amazon RDS (Non-Aurora)





# Amazon Aurora



# Amazon Aurora

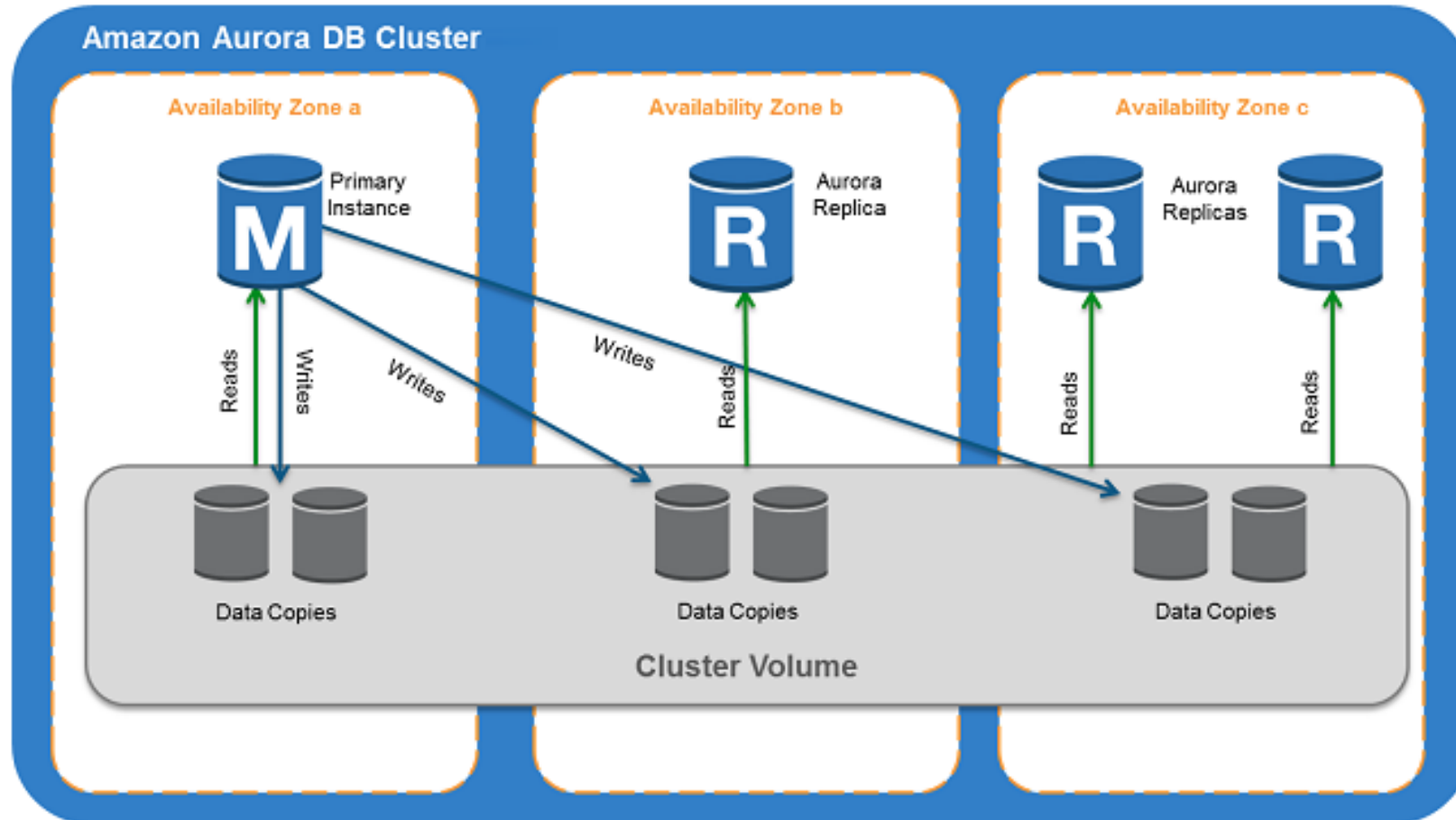
- Amazon Aurora is a fully managed relational database engine designed for high performance, scalability, and availability, compatible with **MySQL** and **PostgreSQL**.
- **Performance:**
  - Up to **5x faster** than standard MySQL and **3x faster** than PostgreSQL.
  - Supports millions of transactions per second.
- **Scalability:**
  - Automatically scales storage up to **128 TiB**.
  - Supports up to **15 low-latency read replicas** for read-heavy workloads.
- **High Availability:**
  - Replicates data across **6 copies** in 3 Availability Zones (AZs).
  - Automatic failover occurs in less than 30 seconds, ensuring high resilience.
- **Cost-Effective:** Offers the performance of commercial databases at a fraction of the cost, with pay-as-you-go pricing.

# Amazon Aurora

- **Backup and Durability:**
  - Continuous: Automatically backs up to S3 in real time
  - Incremental: Only changes are backed up
  - Support PITR.
- **Security:**
  - Encryption at rest and in transit.
  - Integrated with AWS IAM.
- **Global Database:**
  - **Cross-region replication** for global applications.
  - Enables **low-latency reads** and **disaster recovery** across continents.
  - Failover from primary to secondary regions in **under a minute**.

# Amazon Aurora

- It contains a primary instance and multiple replicas



# Aurora Cluster

- **Cluster Components:**

- **DB Instances:** One **Primary DB Instance** + up to **15 Aurora Replicas**.
- **Cluster Volume:** A **distributed, virtual storage layer** spanning **3 Availability Zones (AZs)**, with **6 copies** of the data (2 per AZ).

- **Primary DB Instance:**

- Handles **read and write operations**.
- Performs all **data modifications** to the cluster volume.
- Each cluster has **only one primary instance**.

# Aurora Replicas

- **Shared Storage:** Aurora replicas share the same underlying storage as the primary instance, reducing costs and avoiding data duplication.
- **Scalability:** Each Aurora DB cluster supports up to **15 replicas** in addition to the primary DB instance.
- **High Availability:**
  - Replicas can be placed in **separate Availability Zones (AZs)** for fault tolerance.
  - Automatic failover to a replica occurs within **30 seconds** if the primary instance becomes unavailable.

# Aurora Failover

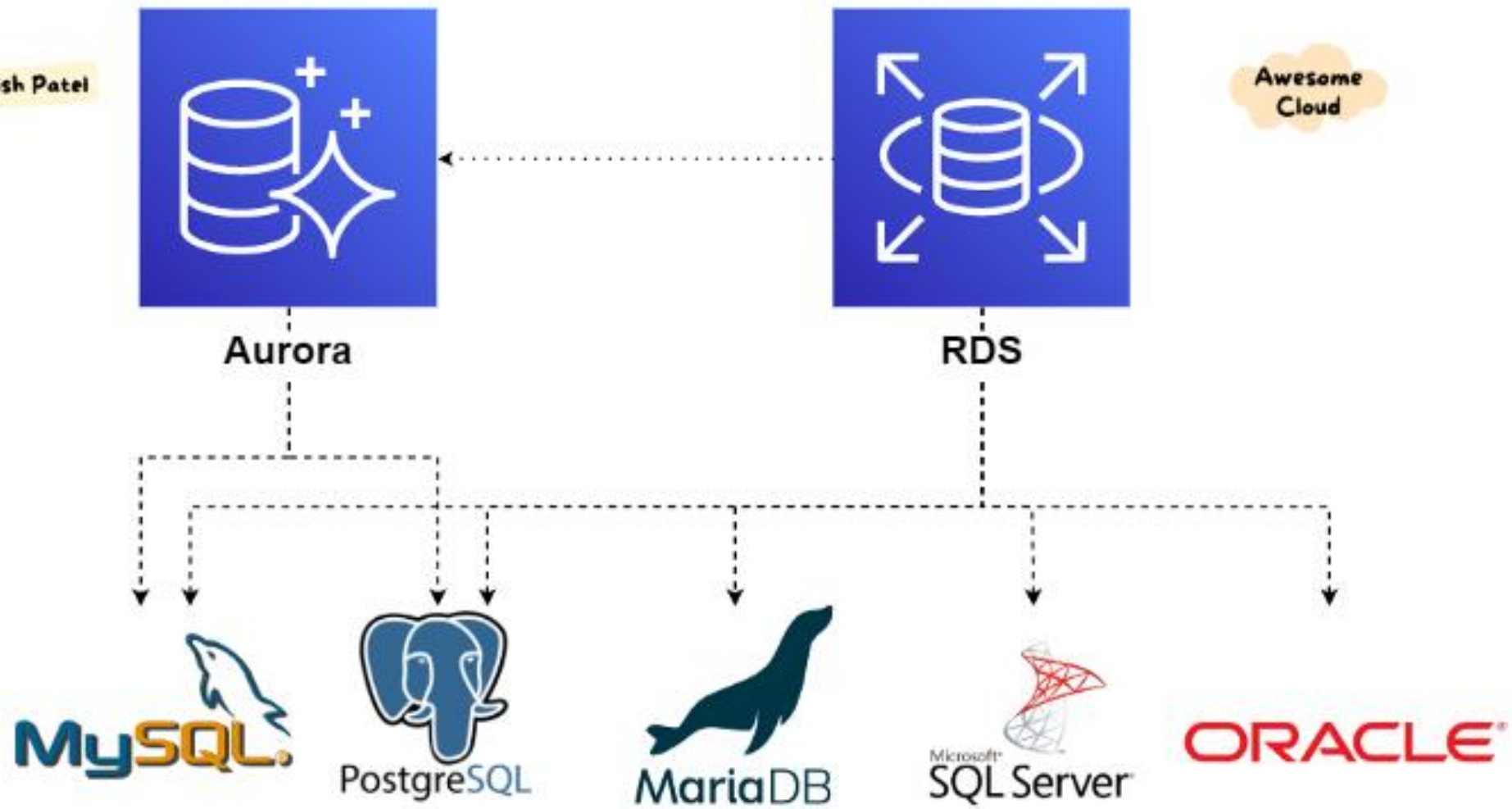
- **Replication Efficiency:**
  - **Not copy or transfer data** during replication.
  - Minimize **replication lag**, typically **less than 10 milliseconds** after updates by the primary instance.
- **Failover Mechanisms:**
  - **With Aurora Replica:** Promotes an existing replica to primary within **30 seconds**.
  - **Without Aurora Replica:** Recreates a new primary instance, which takes about **10 minutes**.

Feature	Amazon RDS (Non-Aurora)	Amazon Aurora
Database Engines	Supports MySQL, PostgreSQL, Oracle, SQL Server, MariaDB.	Compatible with MySQL and PostgreSQL.
Performance	Performance depends on instance type and storage configuration.	Up to 5x faster than MySQL and 3x faster than PostgreSQL.
Replication	Synchronous standby replica in Multi-AZ for high availability.	Aurora Replicas share the same distributed storage with the primary; typical replication lag is under 10 ms.
Scalability	Manual scaling; limited to a single instance for writes.	Auto-scales storage (up to 128 TiB) and supports up to 15 low-latency read replicas.
Failover	Promotes standby replica during failover (~1-2 minutes).	Automatic failover to Aurora Replica (~30 seconds); creates new primary if no replica is available (~10 minutes).
Storage	Single AZ or Multi-AZ replication for durability.	Distributed, fault-tolerant storage across 3 AZs with 6-way replication.
Cost	Lower cost; charges based on instance type and storage.	Higher cost but optimized for high performance and scalability.
Use Case	Suitable for standard workloads with traditional RDBMS needs.	Ideal for high-performance, high-availability, and large-scale applications.



Ashish Patel

Awesome  
Cloud



Source: [AWS — Difference between Amazon Aurora and Amazon RDS](#)

# Best practices and considerations

- Security:
  - **Encryption:** Encrypt data both at rest using AWS Key Management Service (KMS) and in transit using SSL/TLS to protect sensitive data.
  - **Access Control:** Implement least privilege access using IAM to control who can manage and access database resources.
  - **Network Security:** Advocate for hosting databases in private subnet within a VPC to isolate them from the public internet and utilize security groups and network ACLs effectively.
- Performance Optimization
  - **Right sizing:** choose the correct instance size and database type based on the workload's specific needs for appropriate provisioning.
  - **Indexing and queries:** Using indexing strategies and writing efficient SQL queries to improve performance.
  - **Monitoring and tuning:** Regularly monitor databases to identify the performance bottlenecks and tune the database accordingly by using tools like CloudWatch or Performance Insights.

# Best practices and considerations

- **Cost Management:**
  - **Managed Services:** Select the right types of services like Amazon RDS or Aurora
  - **Storage Management:** Select the appropriate storage type and use automated storage scaling
  - **Reserved Instances:** Purchasing Reserved Instances for databases with predictable workloads can save up to 75% over equivalent on-demand capacity.
- **High Availability and Disaster Recovery**
  - **Multi-AZ deployments:** Deploy databases across multiple AZs to ensure high availability and failover protection.
  - **Backup and Restore:** Regular backups, point-in-time recovery, and testing the restore process to ensure data durability and recoverability.
  - **Read Replicas:** For read-heavy database workloads, recommend the use of read replicas to enhance application performance and distribute loads.

# Demo

aws

Search

[Option+S]

United States (N. Virginia)

aws-demo @ 8928-1874-5168

RDS

>

Create database

You can use settings from aurora-mysql to simplify configuration of suggested database add-ons while we finish creating your DB for you.

Create database [Info](#)


Choose a database creation method


☒ Standard create  
You set all of the configuration options, including ones for availability, security, backups, and maintenance.


☐ Easy create  
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.


Engine options

Engine type [Info](#)

☐ Aurora (MySQL Compatible)  


☒ Aurora (PostgreSQL Compatible)  


☐ MySQL  


☐ PostgreSQL  


# References

- <https://docs.aws.amazon.com/>
- ChatGPT: <https://chatgpt.com/>
- Google AI: <https://gemini.google.com/app>