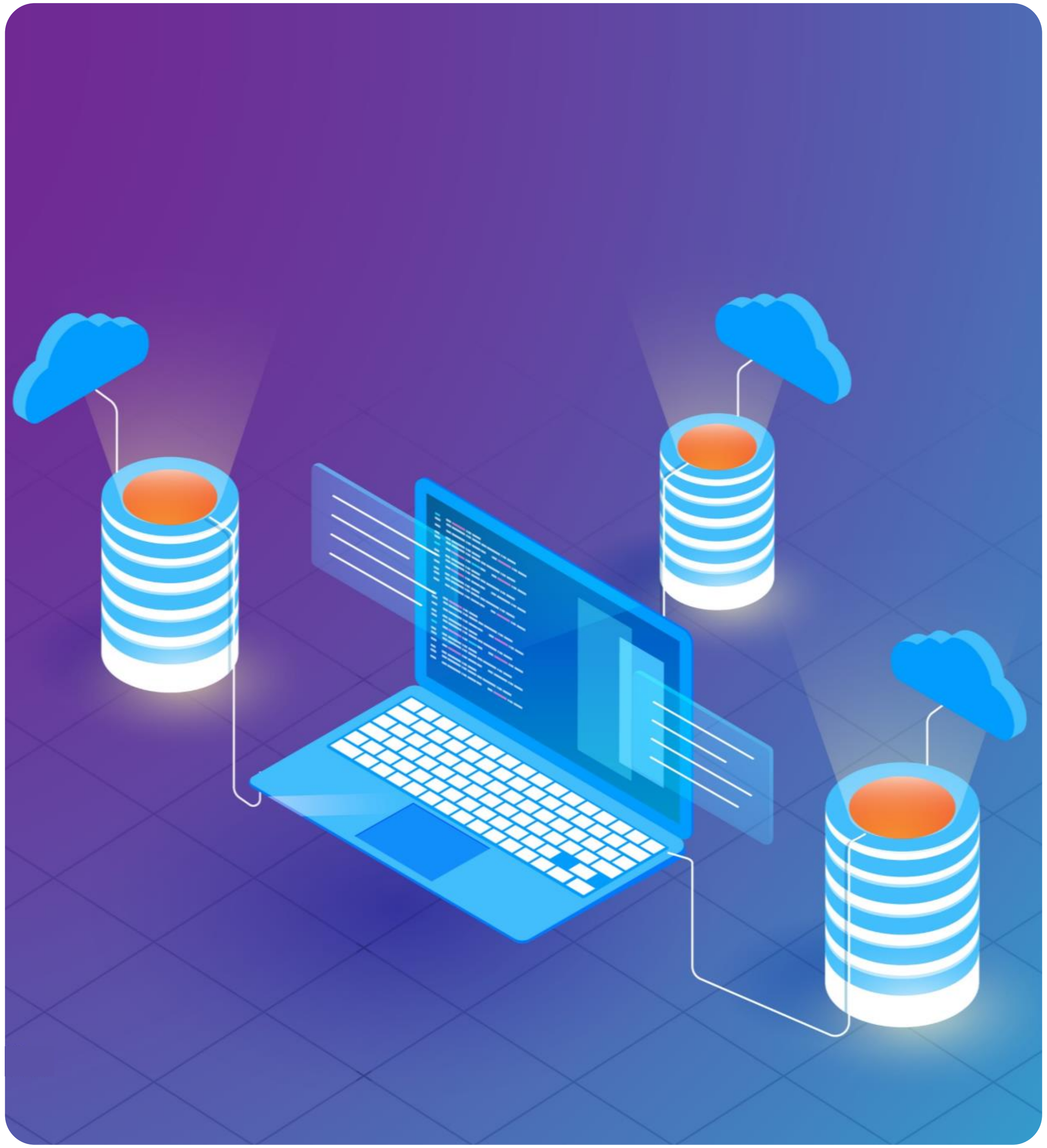
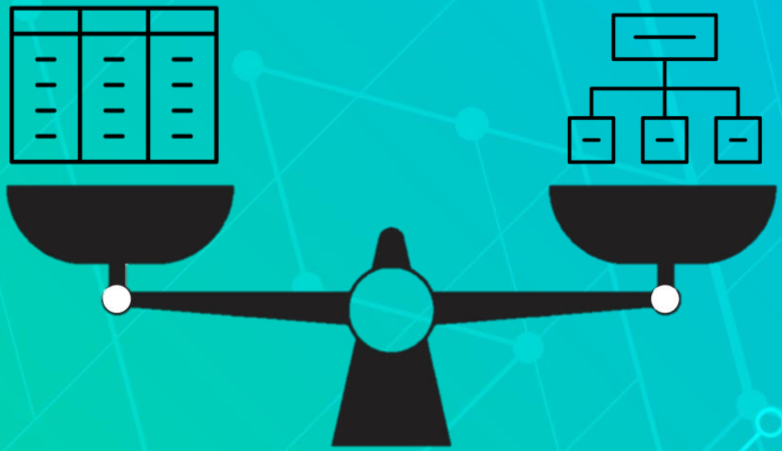
BeSA
Become a Solutions Architect

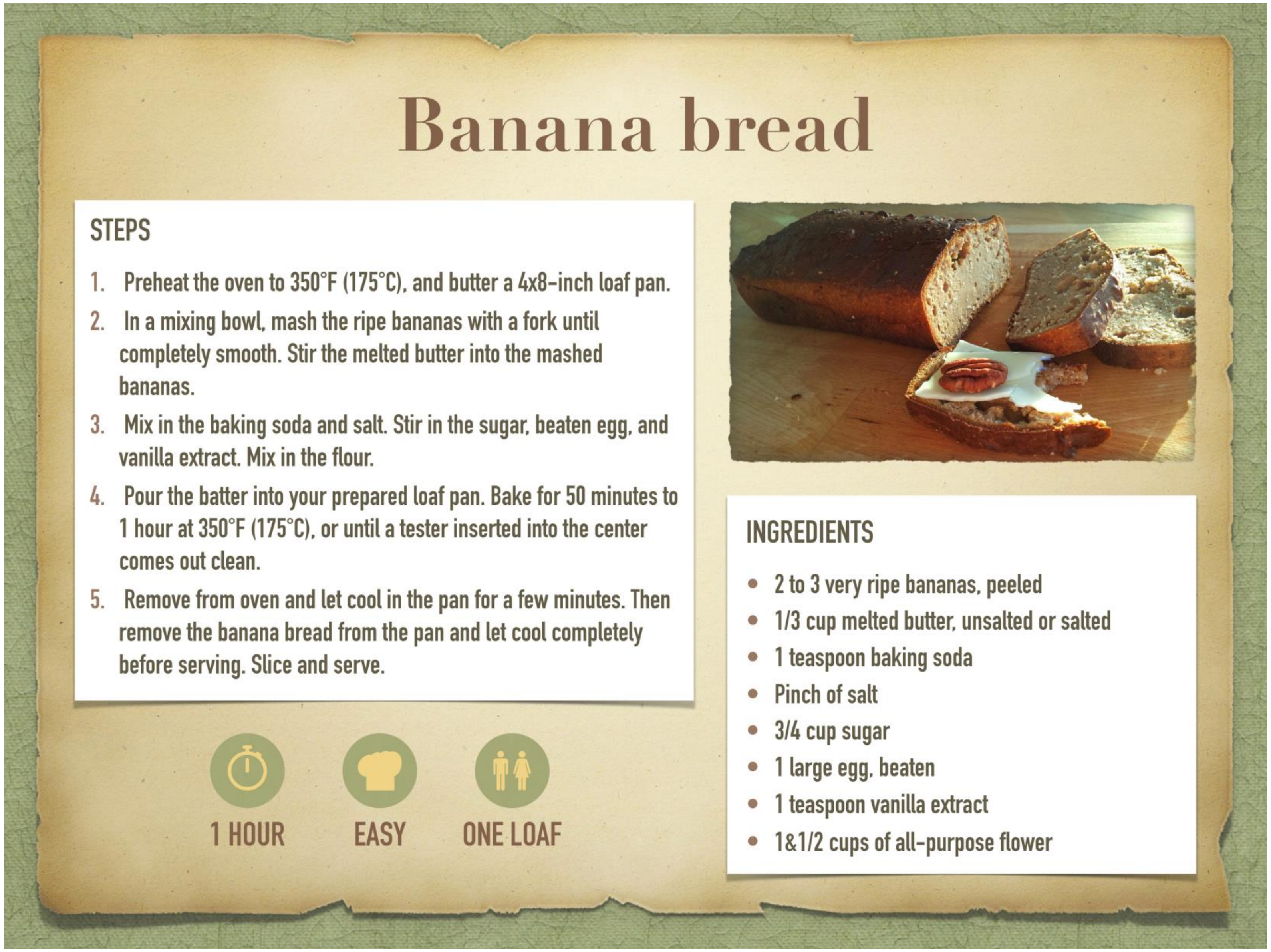
Databases in Cloud





SQL vs No-SQL Databases
Or
Relational vs Non-relational

Let’s cook Banana bread



Banana bread

STEPS

1. Preheat the oven to 350°F (175°C), and butter a 4x8-inch loaf pan.
2. In a mixing bowl, mash the ripe bananas with a fork until completely smooth. Stir the melted butter into the mashed bananas.
3. Mix in the baking soda and salt. Stir in the sugar, beaten egg, and vanilla extract. Mix in the flour.
4. Pour the batter into your prepared loaf pan. Bake for 50 minutes to 1 hour at 350°F (175°C), or until a tester inserted into the center comes out clean.
5. Remove from oven and let cool in the pan for a few minutes. Then remove the banana bread from the pan and let cool completely before serving. Slice and serve.

1 HOUR EASY ONE LOAF

INGREDIENTS

- 2 to 3 very ripe bananas, peeled
- 1/3 cup melted butter, unsalted or salted
- 1 teaspoon baking soda
- Pinch of salt
- 3/4 cup sugar
- 1 large egg, beaten
- 1 teaspoon vanilla extract
- 1&1/2 cups of all-purpose flower

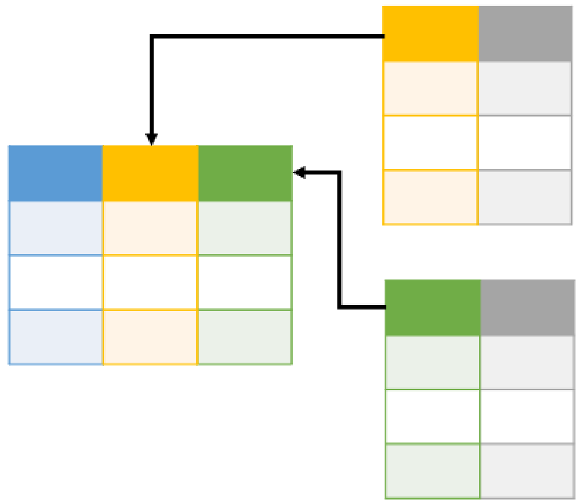
Grocery shopping in a supermarket

CHEESE
CRISPS
FRUITS
DRINKS

SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph

Rows and Columns



Document

```
{
  "Id": "1",
  "FullName":
   {
     "first": "Jane",
     "last": "Doe"
    }
  "Year": "2022",
}
```

How storage evolved?

- In early days of computing Storage was very costly

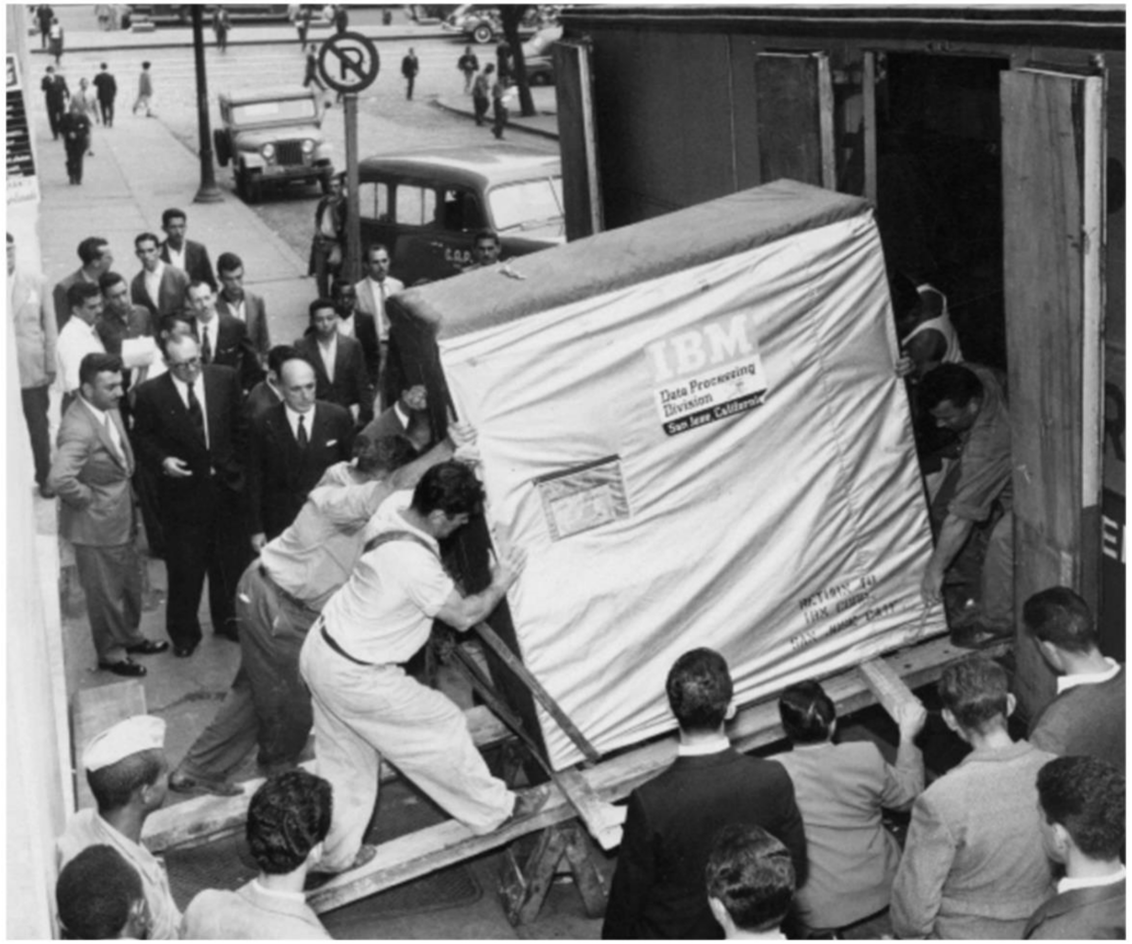


Image Source:
https://www.reddit.com/r/computerscience/comments/ak27u0/ibm_5mb_hard_drive_1956/

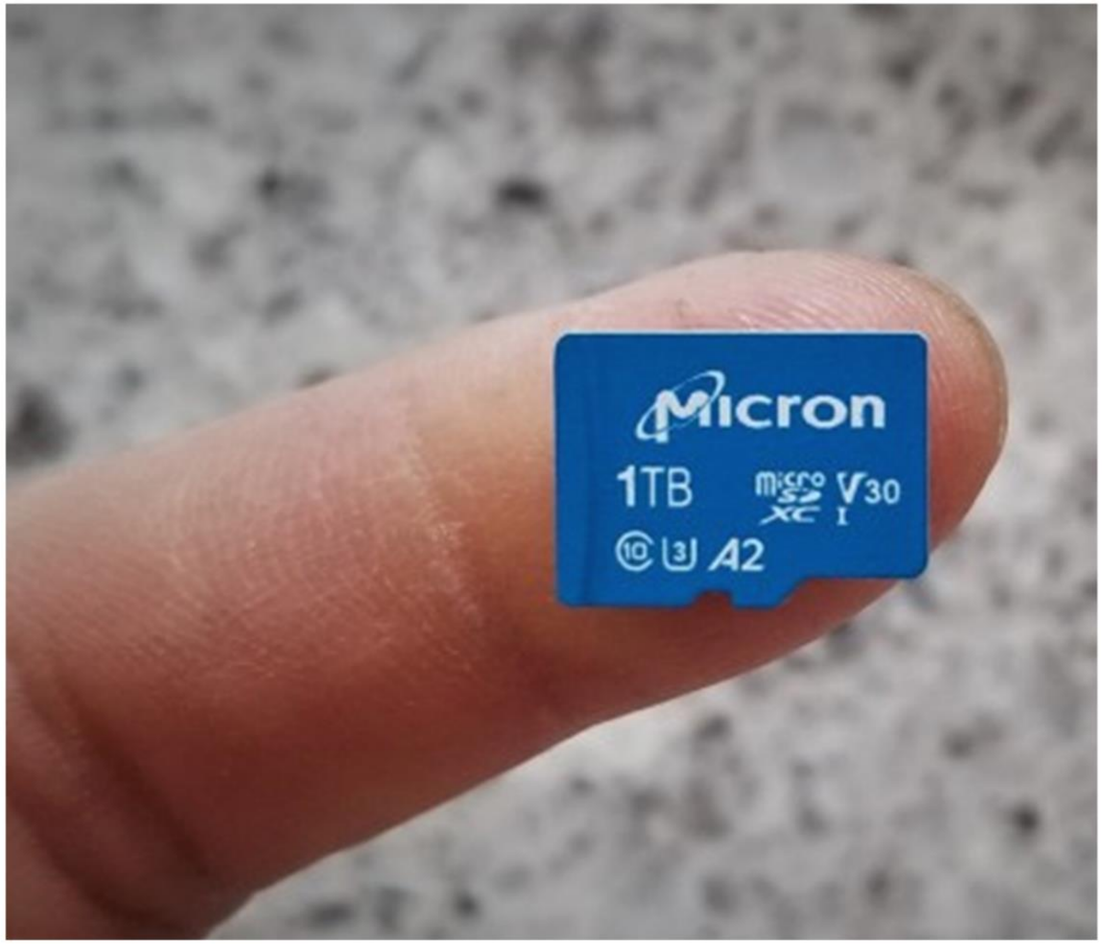
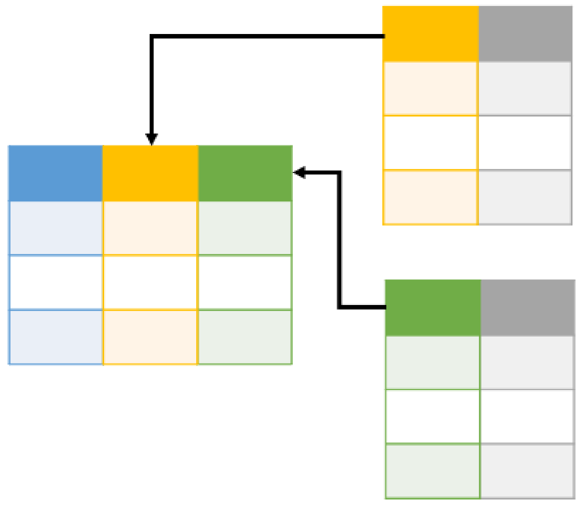


Image Source:
https://www.thessdreview.com/featured/micron-c200-microsd-card-review-1tb-as-high-capacity-becomes-the-norm-in-microsd/

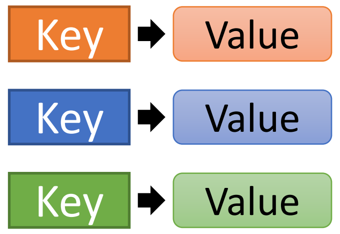
SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph

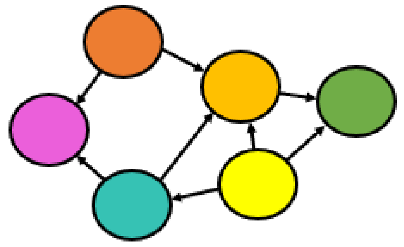
Rows and Columns



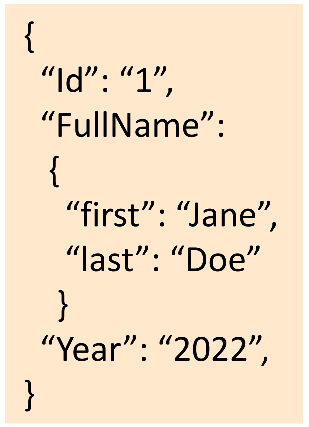
Key Value



Graph

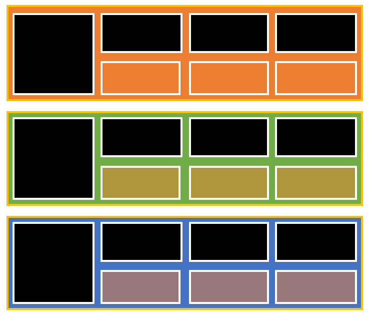


Document



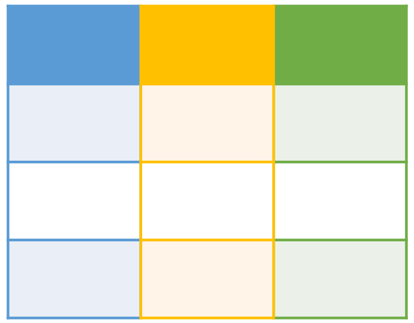
```
{
  "Id": "1",
  "FullName":
   {
     "first": "Jane",
     "last": "Doe"
    }
  "Year": "2022",
}
```

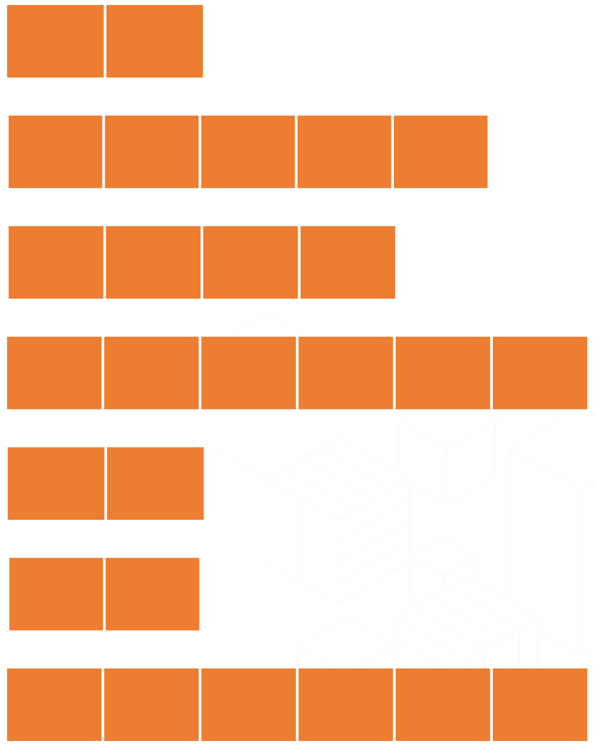
Wide Column



SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic





SQL vs. NoSQL Databases

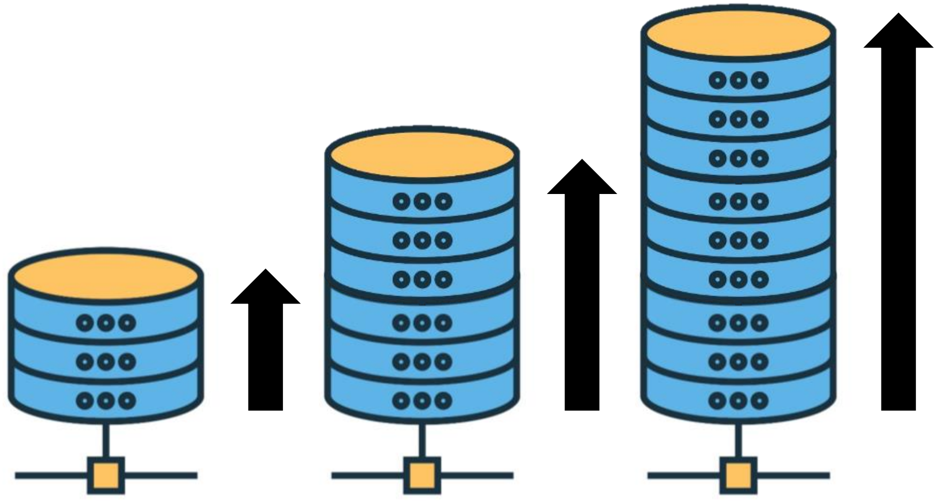
	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents

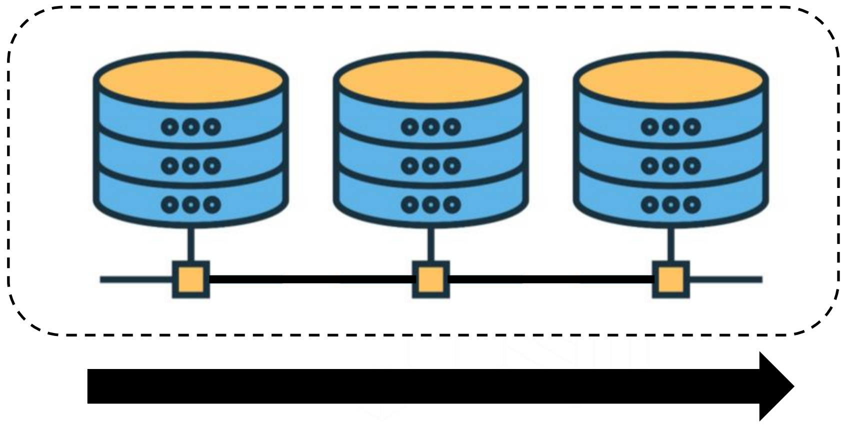
```
/*  Return all of the songs by an artist Elvis   */
SELECT * FROM Music
WHERE Artist= ‘Elvis’;
```

```
/*  Return all of the songs by an artist Elvis   */
{
   TableName: “Music”,
   KeyConditionExpression: “Artist =
:a”,
   ExpressionAttributeValues:
   {
    “:a”: “Elvis”
    }
}
```

SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scaling	Vertical	Horizontal



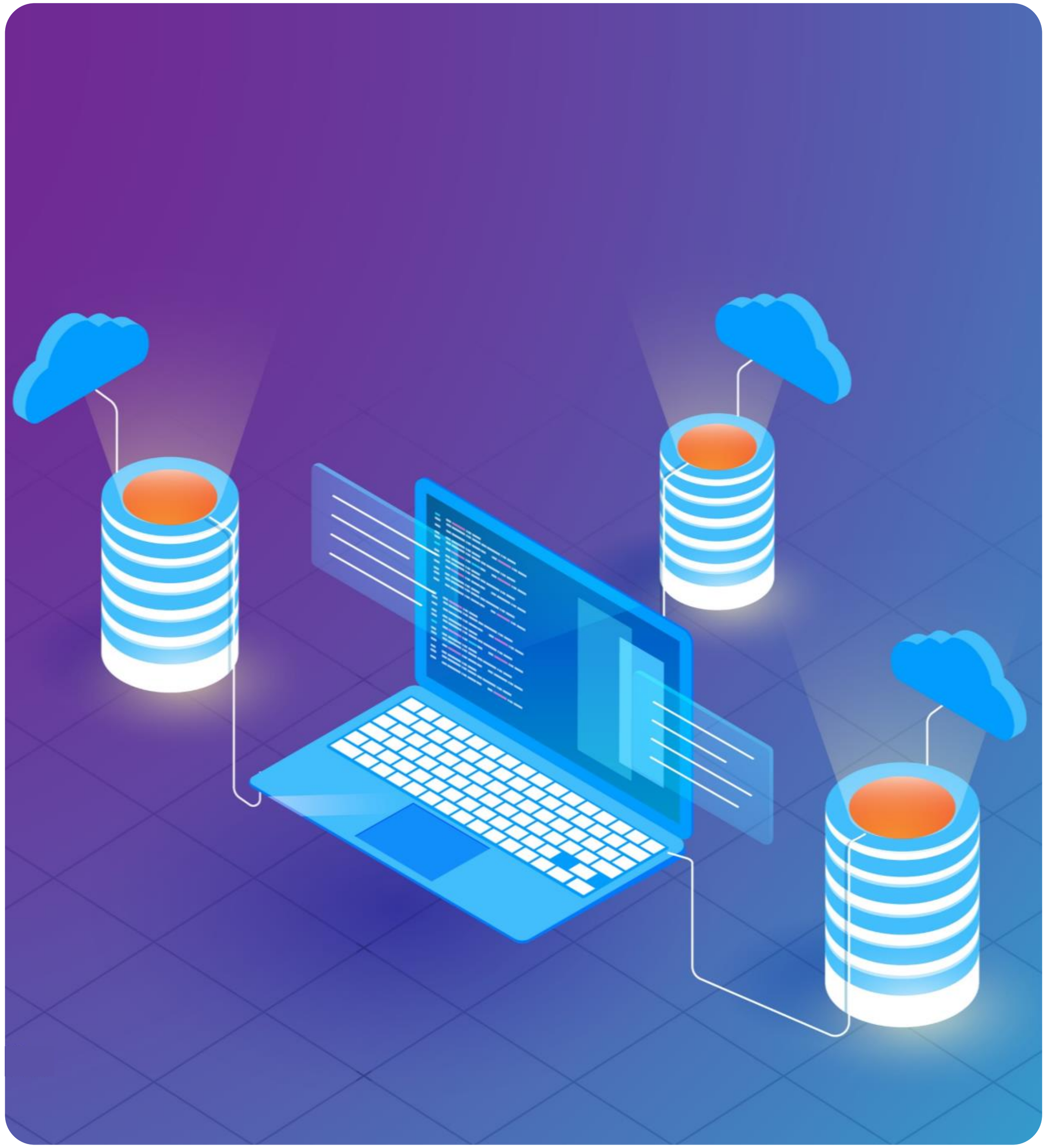


SQL vs. NoSQL Databases

	SQL (Optimized for Storage)	NoSQL (Optimized for performance)
Data Storage	Rows and Columns	Key-value, Document, Wide-column, Graph
Schema	Fixed	Dynamic
Querying	Using SQL	Focused on collection of documents
Scaling	Vertical	Horizontal
Transactions	Supported	Support varies

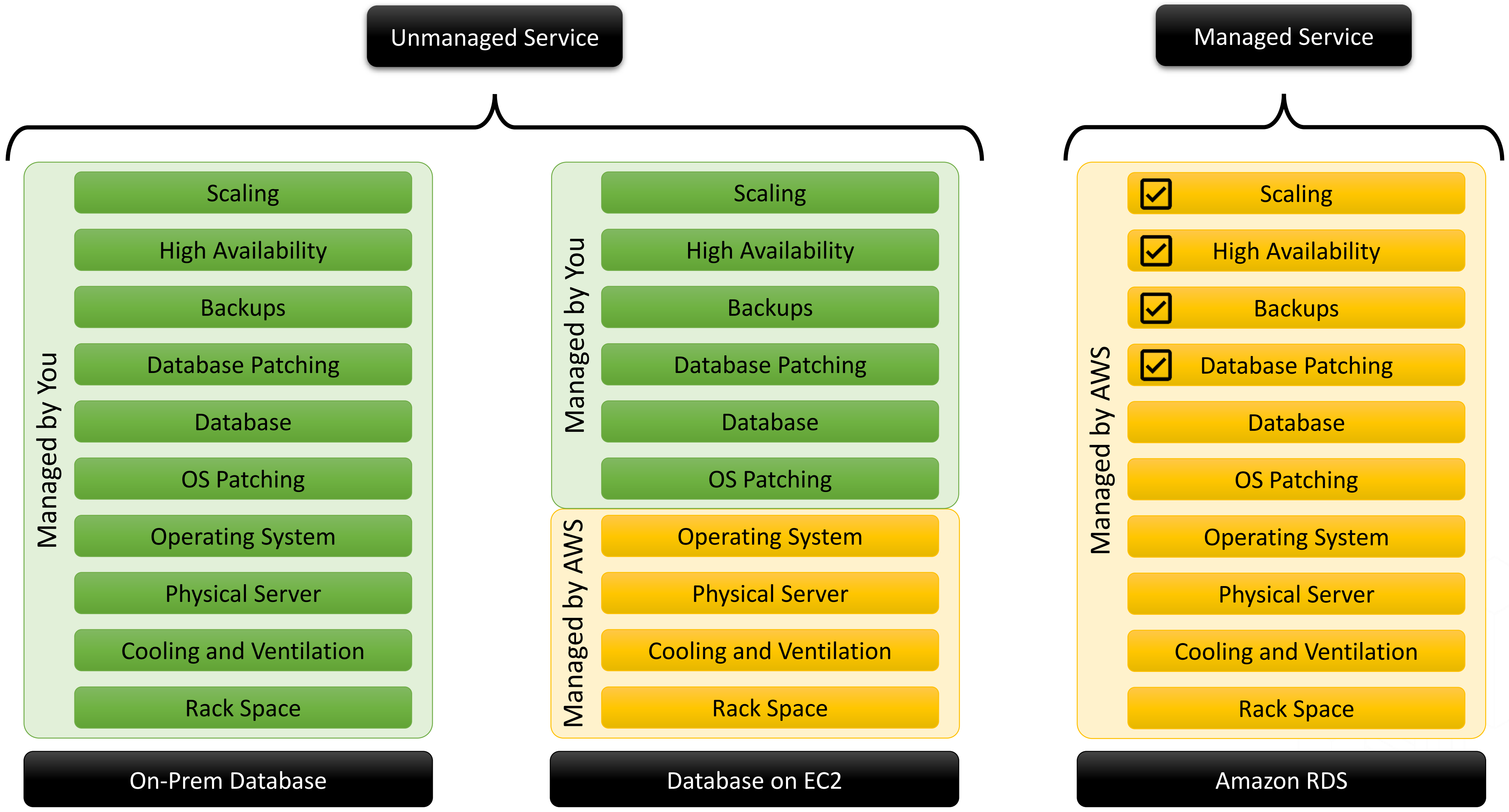
A	C	I	D
Atomicity Transitions are all or nothing	**Consistency** Only valid data is saved	**Isolation** Transactions do not affect each other	**Durability** Written data won't be lost

B A	S	E
Basically Available System does guarantee availability	**Soft state** System may change over time	**Eventual consistency** system will become consistent over time



Amazon RDS

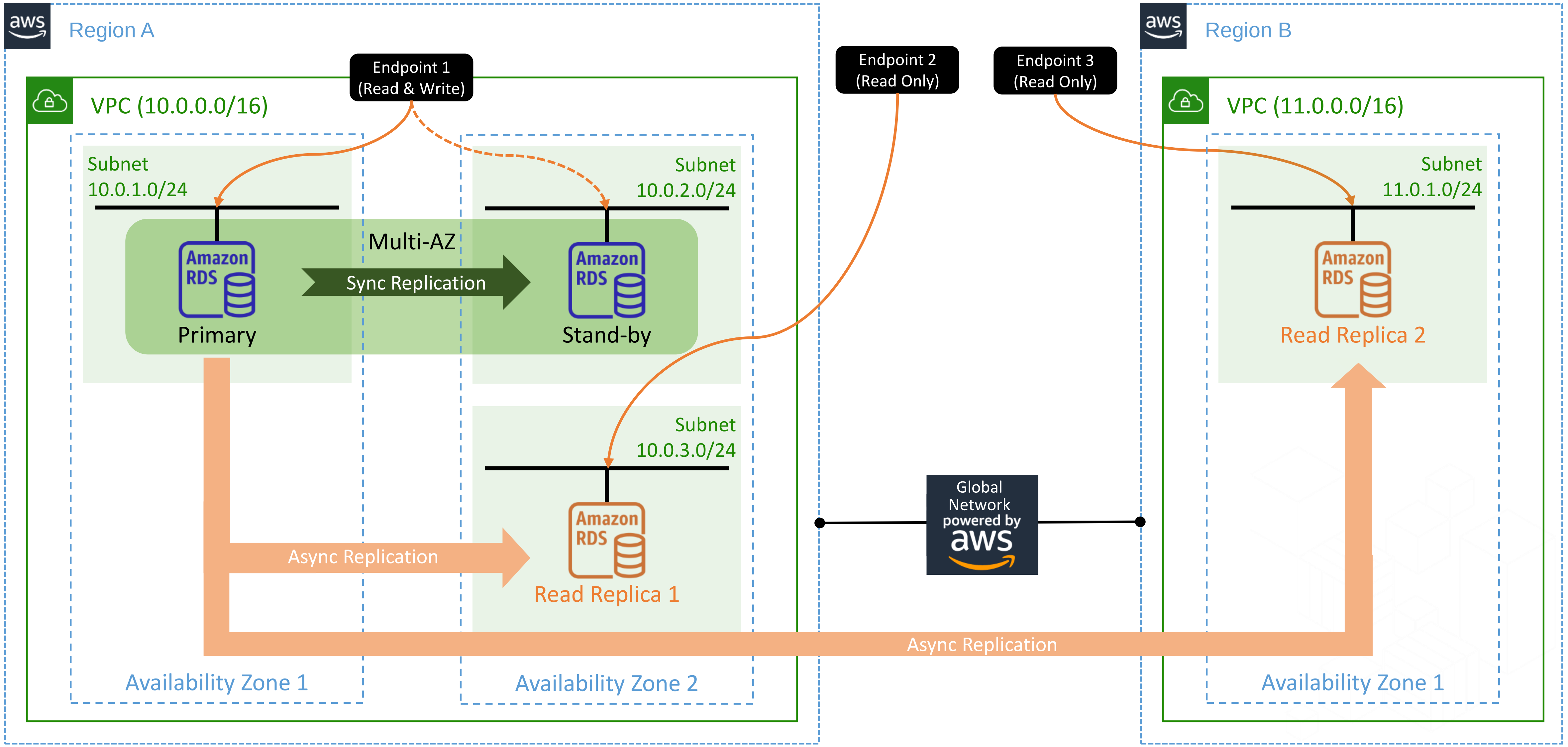
Running and Maintaining Databases



Amazon RDS

- Choice of different database engine
  - MySQL / PostgreSQL / Maria DB
  - MS SQL Server / Oracle
  - Amazon Aurora
- Supports High Availability and Read-Replica
- Snapshots can be copied across region
- Can migrate databases using Database Migration Service (DMS)
- Pricing
  - On-demand or Reserved Instance

RDS – Multi-AZ and Read Replica



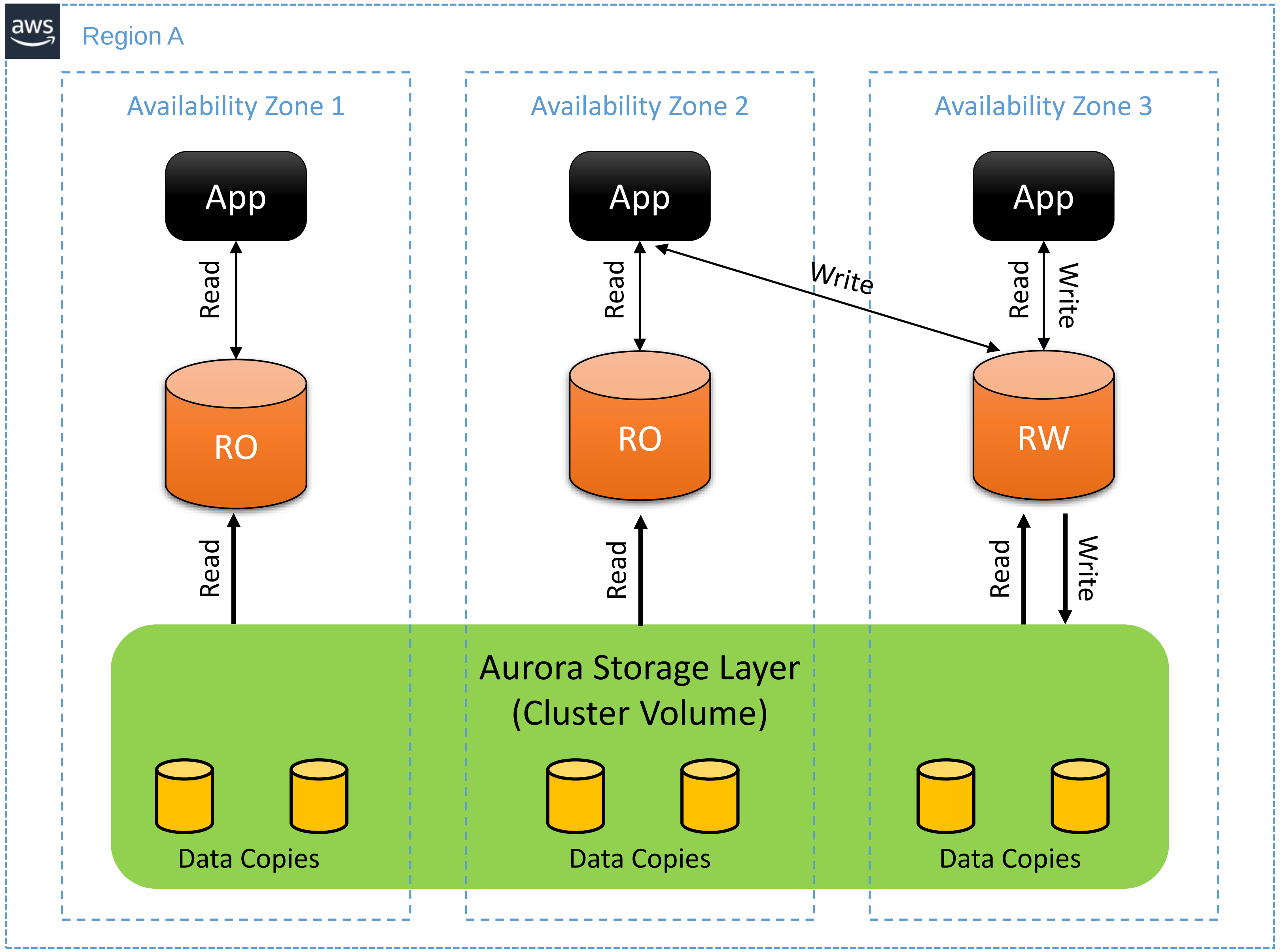
Why Amazon Aurora?

- Commercial Databases – Oracle, MS SQL Server
  - Features/Performance ++++
  - Cost $$$$
- Open Source Databases – MySQL, MariaDB, PostgreSQL
  - Features/Performance ++
  - Cost $$
- Amazon Aurora
  - Features/Performance ++++
  - Cost $$$

Amazon Aurora Flavours

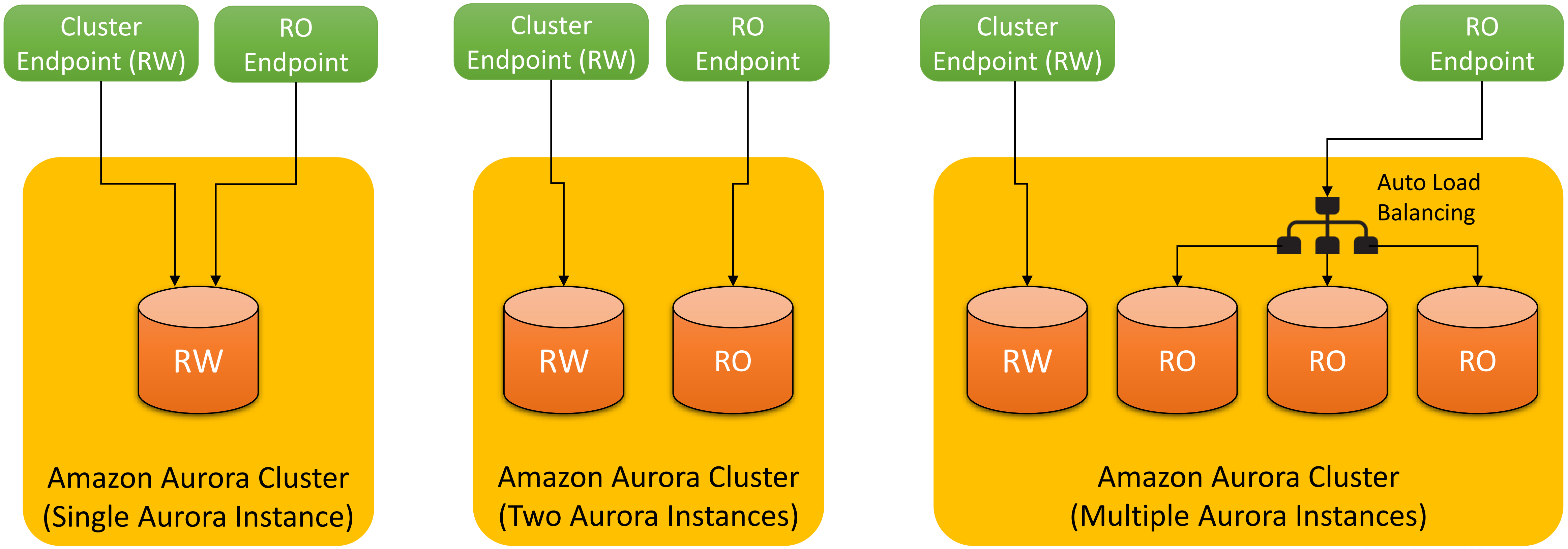
- Single-Master
- Multi-Master
- Global Database
- Serverless v2 and v1

Amazon Aurora



- Separates compute and storage layers
- Drop in compatibility with MySQL and PostgreSQL
- Six-way replication across three AZs
- Up to 15 read replicas with replica lag under 10-ms
- Automatic monitoring with failover

Amazon Aurora Cluster – Single Master


Cluster Endpoint (RW)
RO Endpoint
RW
Amazon Aurora Cluster (Single Aurora Instance)
Cluster Endpoint (RW)
RO Endpoint
RW
RO
Amazon Aurora Cluster (Two Aurora Instances)
Cluster Endpoint (RW)
RO Endpoint
Auto Load Balancing
RW
RO
RO
RO
Amazon Aurora Cluster (Multiple Aurora Instances)

- In a single-master cluster, a single DB instance performs all write operations and any other DB instances are read-only. If the writer DB instance becomes unavailable, a failover mechanism promotes one of the read-only instances to be the new writer.

Reference:

FAQs

Category:

Database

Amazon Aurora

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- Amazon Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.
- It features a distributed, fault-tolerant, and self-healing storage system that is decoupled from compute resources.

Why?

- Aurora automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups while providing the security, availability, and reliability of commercial databases at 1/10th the cost.

When?

- Amazon Aurora is a great option for any enterprise application that can use a relational database.
- You need high performance and availability with up to 15 low-latency read replicas, point-in-time recovery, continuous backup to Amazon S3, and replication across three AZs.

Where?

- Amazon Aurora is a regional service, it automatically maintains six copies of your data across three AZs.
- Cross-region Aurora replicas can be setup using either physical or logical replication. Physical replication uses Amazon Aurora Global Database, logical replication uses binlog for MySQL and PostgreSQL replication slots for PostgreSQL

Who?

- Amazon Aurora is fully managed by RDS and it automatically and continuously monitors and backs up your database to Amazon S3, enabling granular point-in-time recovery.
- Customer can scale the compute resources allocated to your DB Instance by changing your DB Instance class.

How?

- You choose Aurora as the DB engine option when setting up new database servers through Amazon RDS.
- After launching an Aurora instance, you can connect to it using any database client that supports MySQL or PostgreSQL.

How much?

- For provisioned Aurora, you can choose On-Demand Instances and pay for your database by the hour with no long-term commitments or upfront fees, or choose Reserved Instances for additional savings.
- Aurora storage is billed in per GB-month increments, while I/Os consumed are billed in per million request increments.

Reference:

FAQs

Category:

Database

Amazon Aurora Serverless

Complete book:

Click Here

Created by:

Ashish Prajapati

What?

- Amazon Aurora Serverless is an on-demand, autoscaling configuration for Amazon Aurora. It automatically starts up, shuts down, and scales capacity up or down based on your application's needs.
- Aurora Serverless v2 is available for Aurora MySQL-Compatible and PostgreSQL-Compatible editions.

Why?

- Manually managing database capacity can take up valuable time and can lead to inefficient use of database resources. With Aurora Serverless, you create a database, specify the desired database capacity range, and connect your applications.
- It supports the full breadth of Aurora features, including global database, Multi-AZ deployments, and read replicas.

When?

- You want to run your database on AWS provisioning and managing database capacity.
- You have intermittent, infrequent, or unpredictable bursts of requests and want your database to automatically scale capacity to meet the needs of the application's peak load and scale back down when the surge of activity is over.

Where?

- Aurora Serverless is a Regional service. A Multi-AZ Aurora DB cluster has compute capacity available more than one AZ.
- Amazon Aurora Global Database allows a single Amazon Aurora database to span multiple AWS Regions.
- The storage for each Aurora DB cluster consists of six copies of all your data, spread across three AZs.

Who?

- There is no database capacity for you to manage. Amazon Aurora Serverless automatically starts up, scales compute capacity to match your application's usage, and shuts down when it's not in use.
- You can upgrade or switch existing clusters to use Aurora Serverless v2.

How?

- Create a database, specify the desired database capacity range (minimum and maximum amount of resources needed), and connect your application. Aurora automatically adjusts the capacity within the range based on your application's needs.

How much?

- Aurora Serverless measures database capacity in Aurora Capacity Units (ACUs) billed per second when the database is active.
- 1 ACU has approximately 2 GiB of memory with corresponding CPU and networking. Amazon Aurora database storage consumption is billed in per GB-month increments, and I/Os consumed are billed in per million request increments.