SWINBURNE UNIVERSITY OF TECHNOLOGY

Cloud Computing Architecture

Lecture 05 Database Services

includes material from
ACF Module 2.4 – Databases
ACA Module 7 – Designing Web-scale media

Reminder: Assignment 1B due by week 7

- **Due date to <u>Canvas</u>**: by Week 7
- Contribution to final assessment: 20%
- Late submission penalty: 10% of total available marks per day.

Last week

- **Storage in the Cloud**
  - **Big Data**
  - **NFS**
  - **Distributed File Systems (databases next week)**
- **AWS Storage services (ACF Module 7)**
  - Amazon Elastic Block Store (Amazon EBS)
    - Plus some extra notes on Instance Storage
  - Amazon Elastic File System (Amazon EFS)
  - Amazon Simple Storage Service (Amazon S3)
  - Amazon Glacier

Quiz…

This week

- **Relational vs non-relational 'NoSQL' databases**
  - **High-level Comparison**
  - **NoSQL data models**
- **AWS Database Services**
  - **RDS**
  - **Dynamo DB**
  - **(Redshift data warehouse)**
  - **(Aurora)**

Quizzes:
ACF 2.0.4 Database
ACA Mod 7 Web Scale media

SQL and NoSQL Databases

	SQL	NoSQL
Data Storage	Rows and Columns	Key-Value, Document, Graph based, ...
Schemas	Predefined, Fixed	Dynamic
Querying	Using SQL, complex joins possible	Diverse (can have SQL-like QLs)
Scalability	Vertical	Horizontal

ISBN	Title	Author	Format
9182932465265	Cloud Computing Concepts	Wilson, Joe	Paperback
3142536475869	The Database Guru	Gomez, Maria	eBook

```
{
   ISBN: 9182932465265,
   Title: “Cloud Computing Concepts”,
   Author: “Wilson, Joe”,
   Format: “Paperback”
}
```

ACID vs. BASE model

ACID

- SQL
- **A**tomicity, **C**onsistency, **I**solation, and **D**urability

→ Highly consistent but poorly scalable

BASE

- NoSQL
- **B**asically **A**vailable, **S**oft state, **E**ventually Consistent

→Highly scalable but could be inconsistent at some point

NoSQL Databases

- Can be an alternative to relational databases for some types of applications
- Can process large amounts of data with high availability (depending on the NoSQL solution, configuration, and architecture)
- Form a broad category with different implementations and data models
- Have the common feature of distributed fault tolerance

NoSQL Data models and examples

- Wide Column:
  - AWS DynamoDB, Cassandra, Hbase, …
- Document:
  - Apache CouchDB,, IBM Domino, MongoDB, …
- Key-value:
  - Apache Ignite, MemcacheDB, Oracle NoSQL Database, Redis, …
- Graph:
  - Apache Giraph, Neo4J, …

AWS NoSQL Databases

Database type	Use cases	AWS service
Relational	Traditional applications, ERP, CRM, e-commerce	Amazon Aurora, Amazon RDS, Amazon Redshift
Key-value	High-traffic web apps, e-commerce systems, gaming applications	Amazon DynamoDB
In-memory	Caching, session management, gaming leaderboards, geospatial applications	Amazon ElastiCache for Memcached Amazon ElastiCache for Redis
Document	Content management, catalogs, user profiles	Amazon DocumentDB (with MongoDB compatibility)
Wide column	High scale industrial apps for equipment maintenance, fleet management, and route optimization	Amazon Keyspaces (for Apache Cassandra)
Graph	Fraud detection, social networking, recommendation engines	Amazon Neptune
Time series	IoT applications, DevOps, industrial telemetry	Amazon Timestream
Ledger	Systems of record, supply chain, registrations, banking transactions	Amazon QLDB

NoSQL Databases

Does your app need transaction support, ACID compliance, joins, SQL?

- Can it do without these for all, some, or part of its data model?
  Refactor database hotspots to NoSQL solutions

- NoSQL databases can offer increases in flexibility, availability, scalability, and performance

This week

- Relational vs non-relational 'NoSQL' databases
  - High-level Comparison
  - NoSQL data models
- **AWS Database Services**
  - **RDS**
  - **DynamoDB**
  - **Redshift data warehouse**
  - **Aurora**

This week

- Relational vs non-relational ‘NoSQL’ databases
  - High-level Comparison
  - NoSQL data models
- AWS Database Services
  - **RDS**
  - DynamoDB
  - (Redshift data warehouse)
  - (Aurora)

Amazon RDS
Relational Database Service

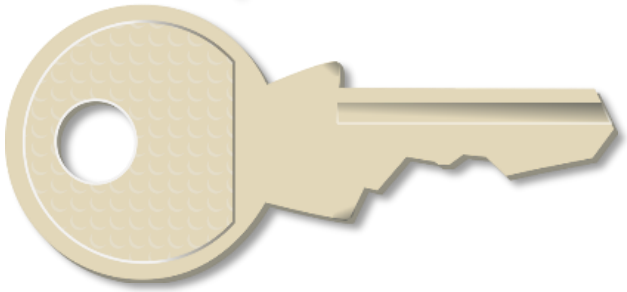
Unmanaged vs. Managed Services

Unmanaged:

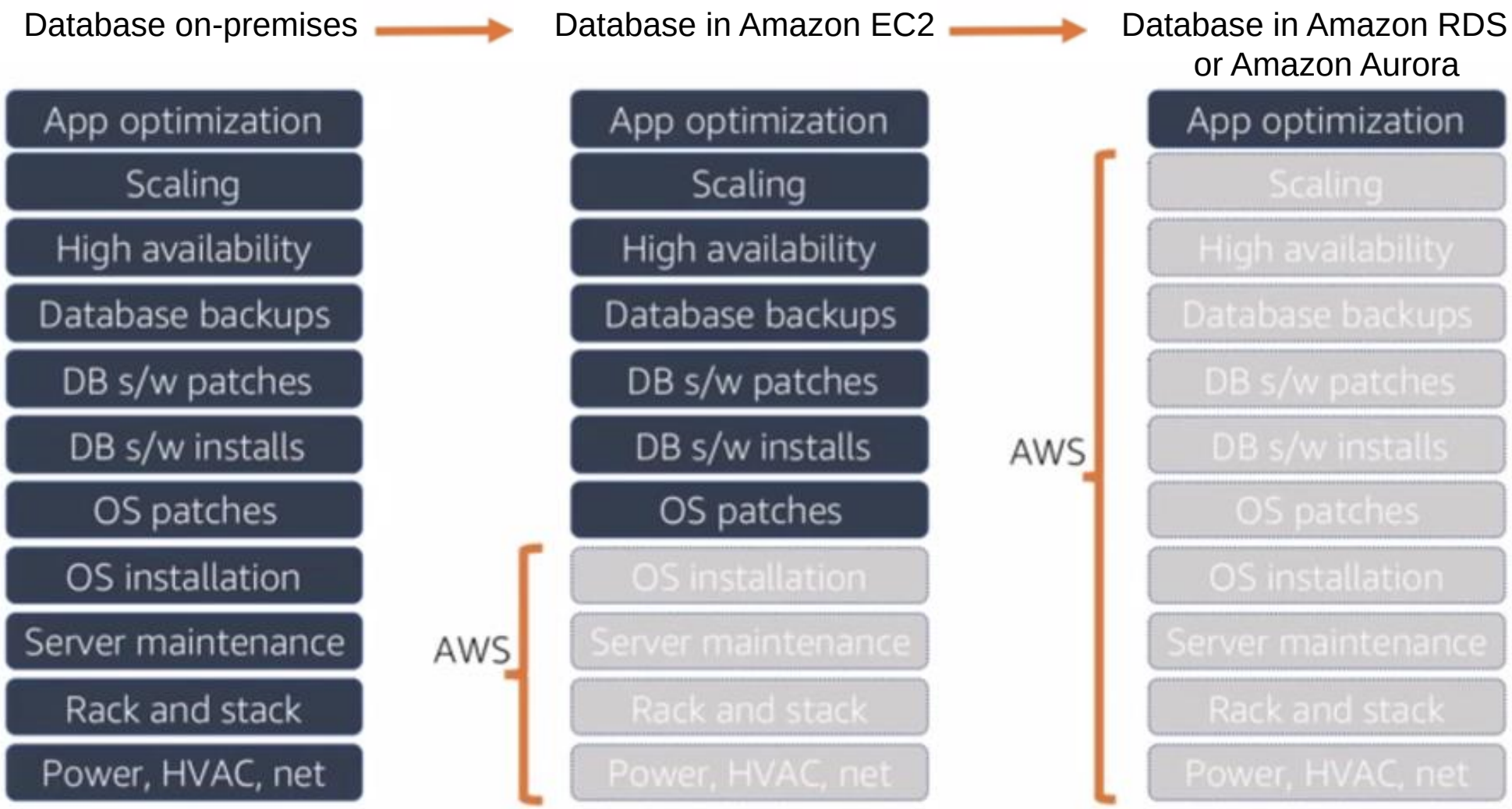
Scaling, fault tolerance, and availability are managed by you.

Managed:

Scaling, fault tolerance, and availability are typically built in to the service.



From On-Premises to Amazon RDS



- Need slide on backup and redundancy etc.

Amazon RDS DB Instances

Amazon RDS

RDS DB master instance

DB Instance Class

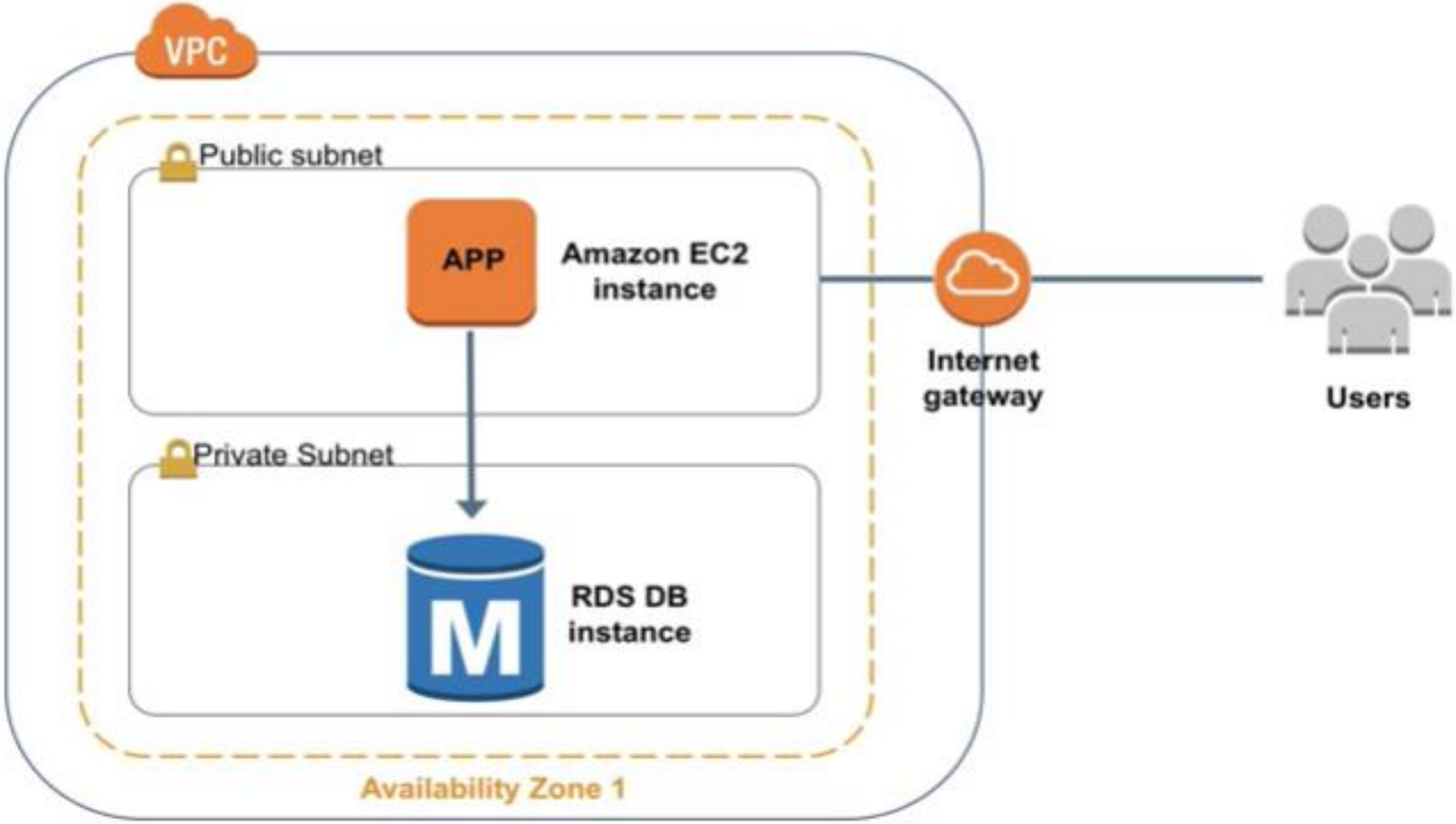
- CPU
- Memory
- Network Performance

DB Instance Storage

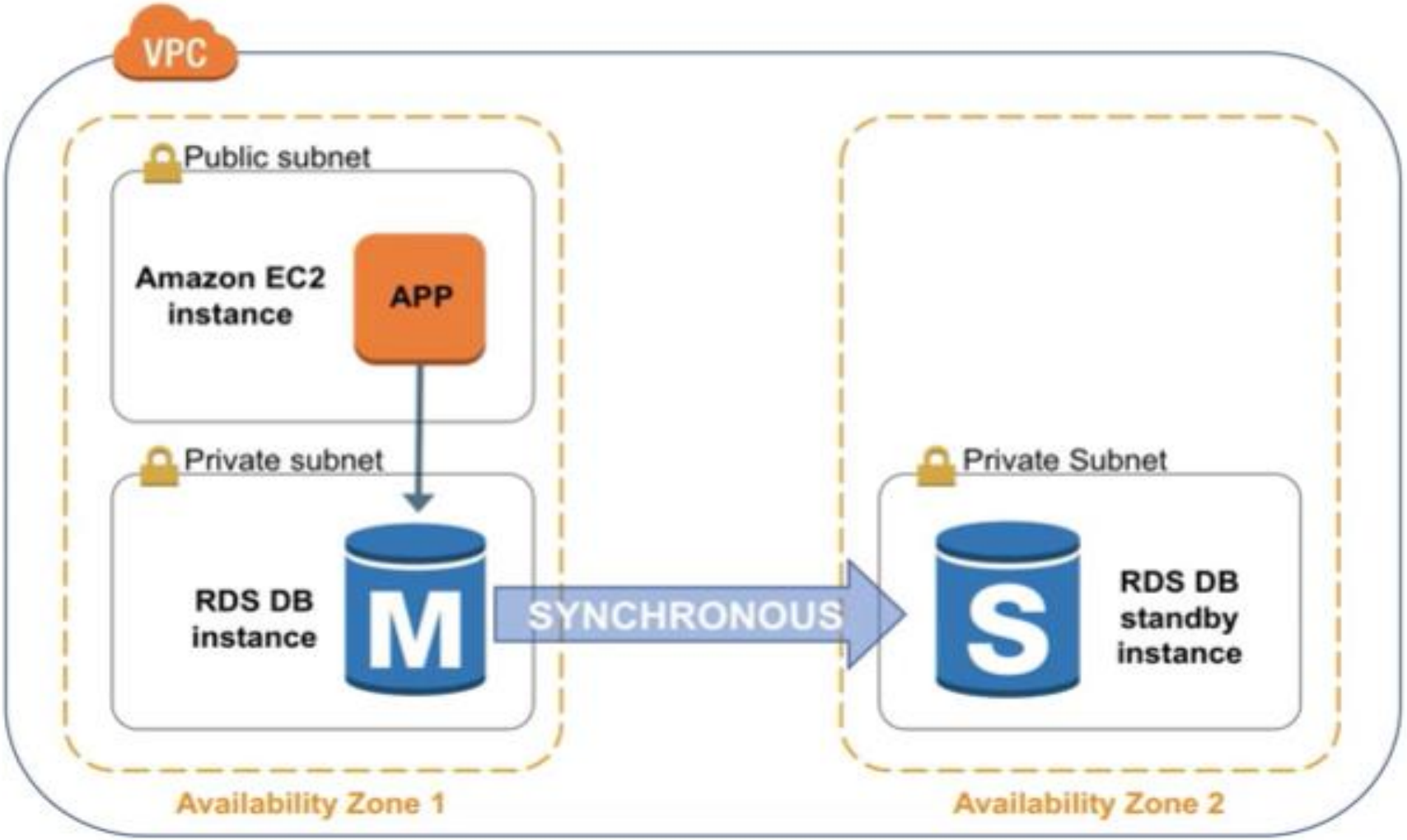
- Magnetic
- General Purpose (SSD)
- Provisioned IOPS

DB Engines

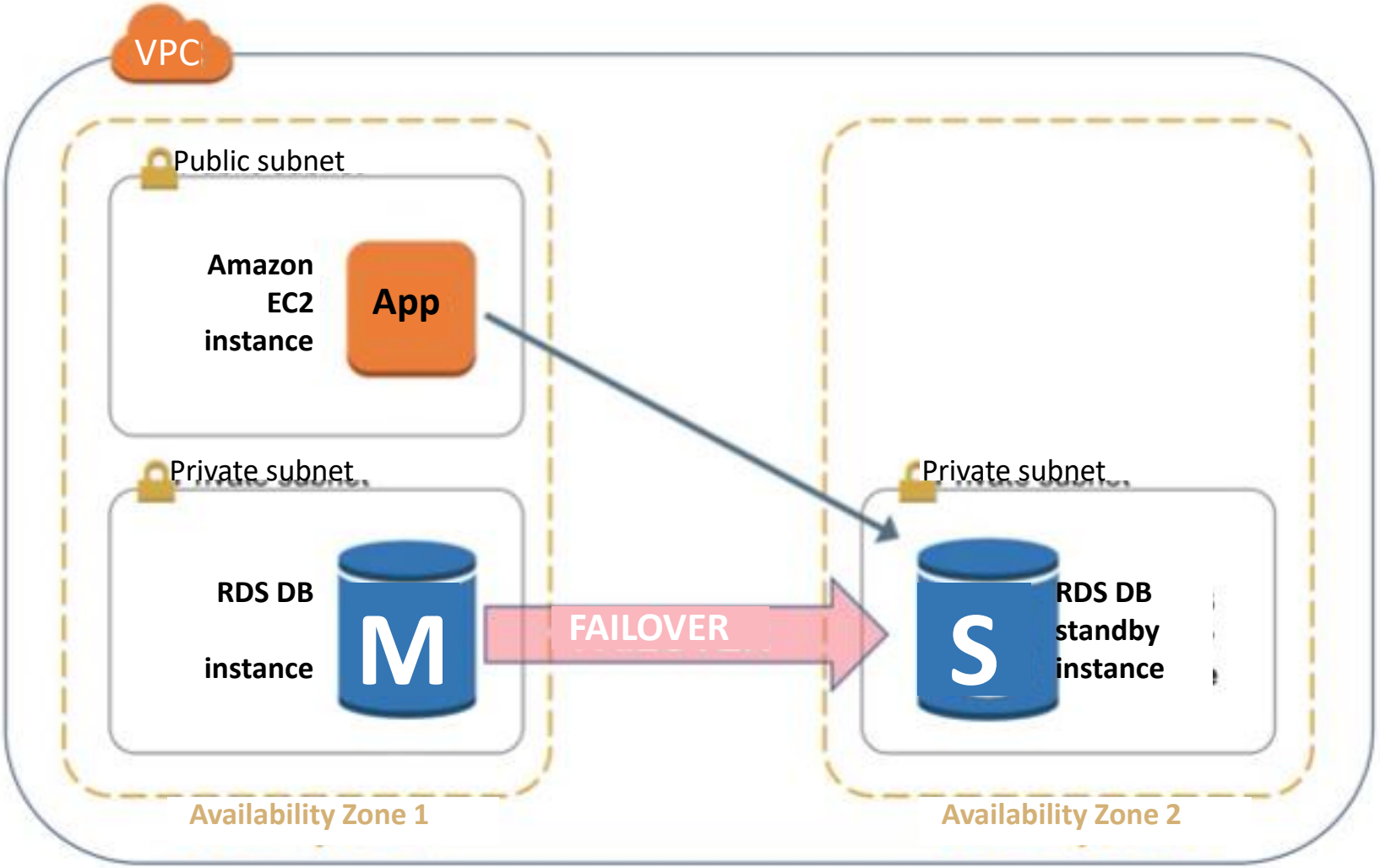
Amazon RDS In a Virtual Private Cloud



High Availability with Multiple Availability Zones



High Availability with Multiple Availability Zones



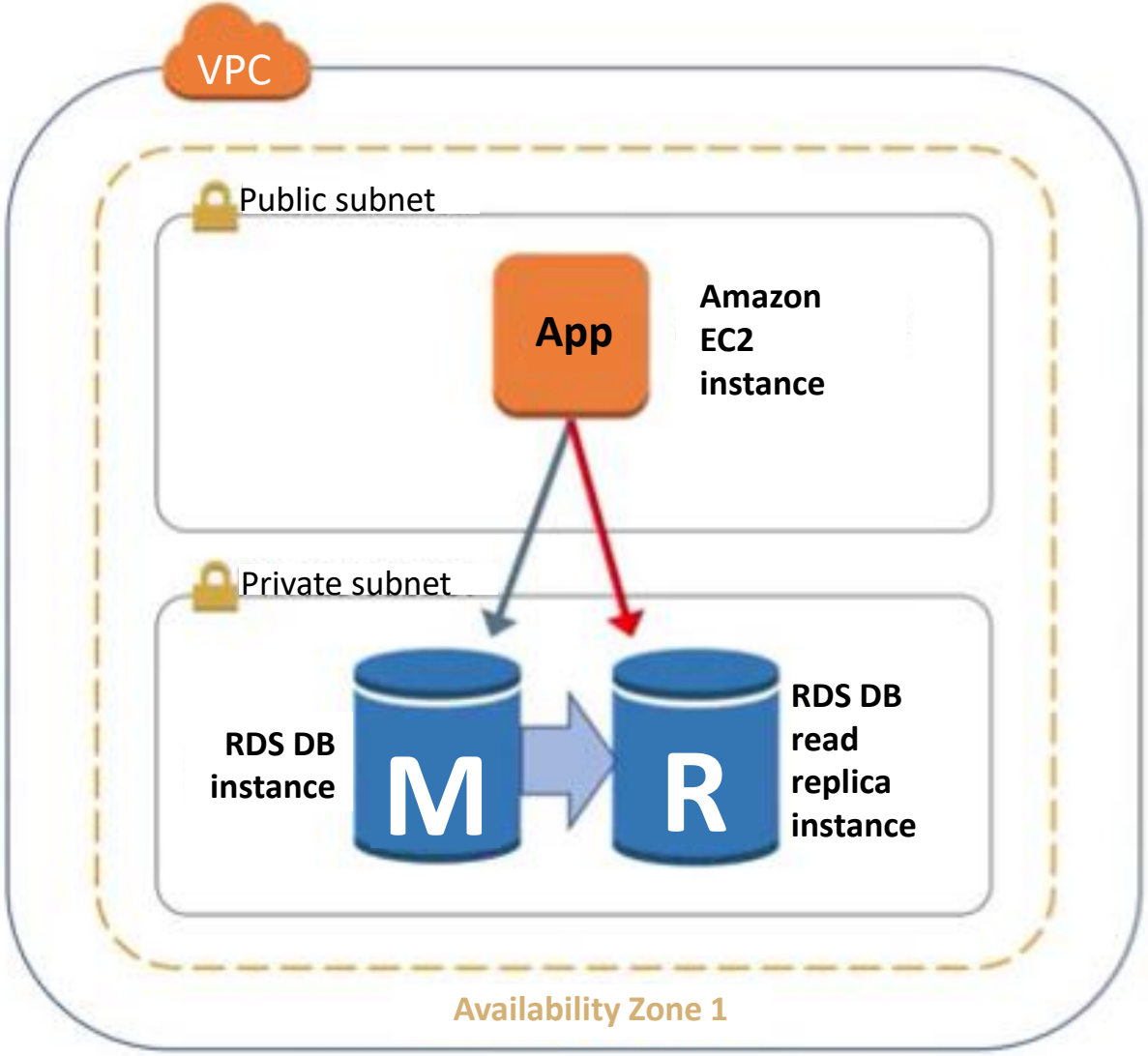
Amazon RDS Read Replicas

Features

- Asynchronous replication
- Promote to master if needed

Functionality

- Read-heavy database workloads
- Offload read queries



When to Use Amazon RDS

Use Amazon RDS when your app requires:

- Complex transactions or complex queries
- A medium to high query/write rate – up to 30K IOPS (15K reads + 15K writes)
- No more than a single worker node/shard
- High durability

Do not use Amazon RDS when your app requires:

- Massive read/write rates (e.g., 150K write/second)
- Sharding due to high data size or throughput demands
- Simple GET/PUT requests and queries that a NoSQL database can handle
- RDBMS customization

Amazon RDS: Clock-Hour Billing and Database Characteristics

1. **Clock-Hour Billing**
   - Resources incur charges when running

2. **Database Characteristics**
   - Physical capacity of database:
     - Engine
     - Size
     - Memory class

Amazon RDS: DB Purchase Type and Multiple DB Instances

aws academy

3. DB Purchase Type

- On-demand database instances
  - Compute capacity by the hour
- Reserved database instances
  - Low, one time, up-front payment for database instances reserved with 1 or 3 year term

4. Number of DB Instances

- Provision multiple DB instances to handle peak loads

Amazon RDS: Storage

5. **Provisioned Storage**
   - No charge
     - Backup storage of up to 100% of database storage for active database
   - Charge (*GB/month*)
     - Backup storage for terminated DB instances

6. **Additional Storage**
   - Charge (*GB/month*)
     - Backup storage in addition to provisioned storage

Amazon RDS: Deployment Type and Data Transfer

7. **Requests**
   - The number of input and output request made to the database

8. **Deployment Type - Storage and I/O charges vary depending**
   - Single Availability Zones
   - Multiple Availability Zones

9. **Data Transfer**
   - No charge for Inbound data transfer
   - Tiered charges for outbound data transfer

In Review

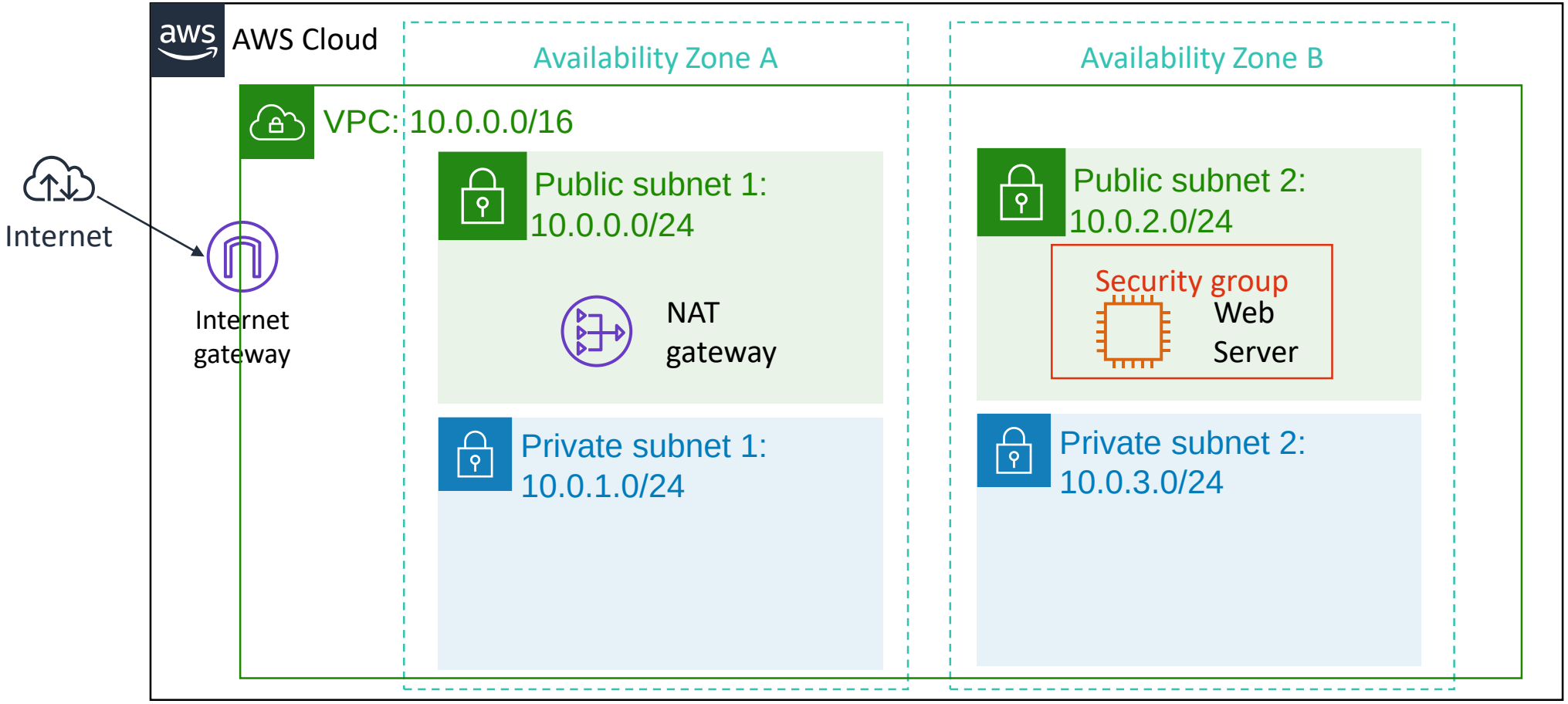
*Set up, operate, and scale **relational databases** in the cloud. Features:*

- Managed service
- Accessible via the console, AWS RDS CLI, or simple API calls
- Scalable (compute and storage)
- Automated redundancy and backup available
- Supported database engines:
  - Amazon Aurora, PostgreSQL, MySQL, MariaDB, ORACLE, Microsoft SQL Server

Amazon RDS Demo

Lab 5: Scenario

This lab is designed to show you how to use an AWS managed database instance to solve a need for a relational database.



Lab 5: Tasks

Security group

Private subnet

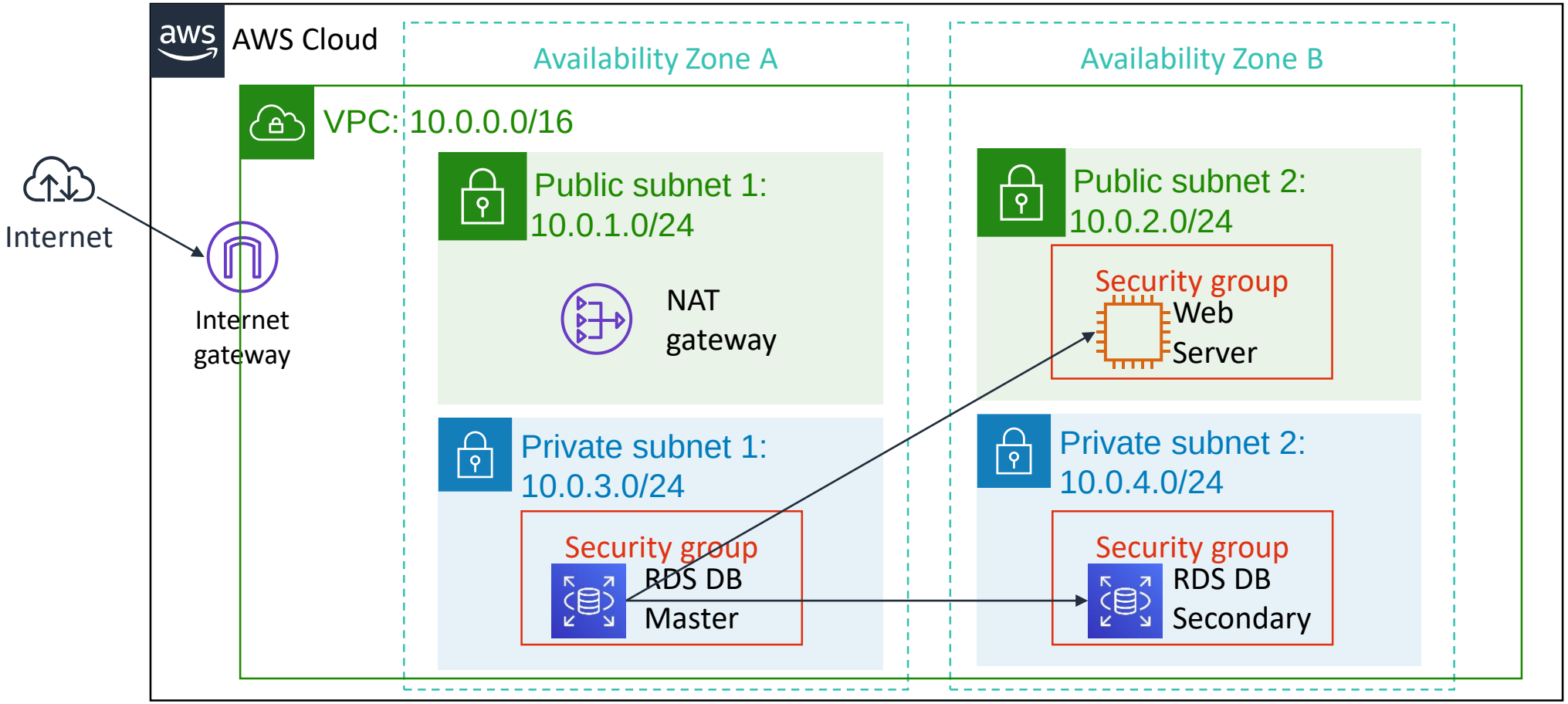
Amazon RDS

Create a **VPC security group.**

Create a **DB subnet group.**

Create an **Amazon RDS DB** instance and interact with your database.

Lab 5: Final product



This week

- Relational vs non-relational 'NoSQL' databases
  - High-level Comparison
  - NoSQL data models
- AWS Database Services
  - RDS
  - **DynamoDB**
  - Redshift data warehouse
  - (Aurora)

Amazon DynamoDB

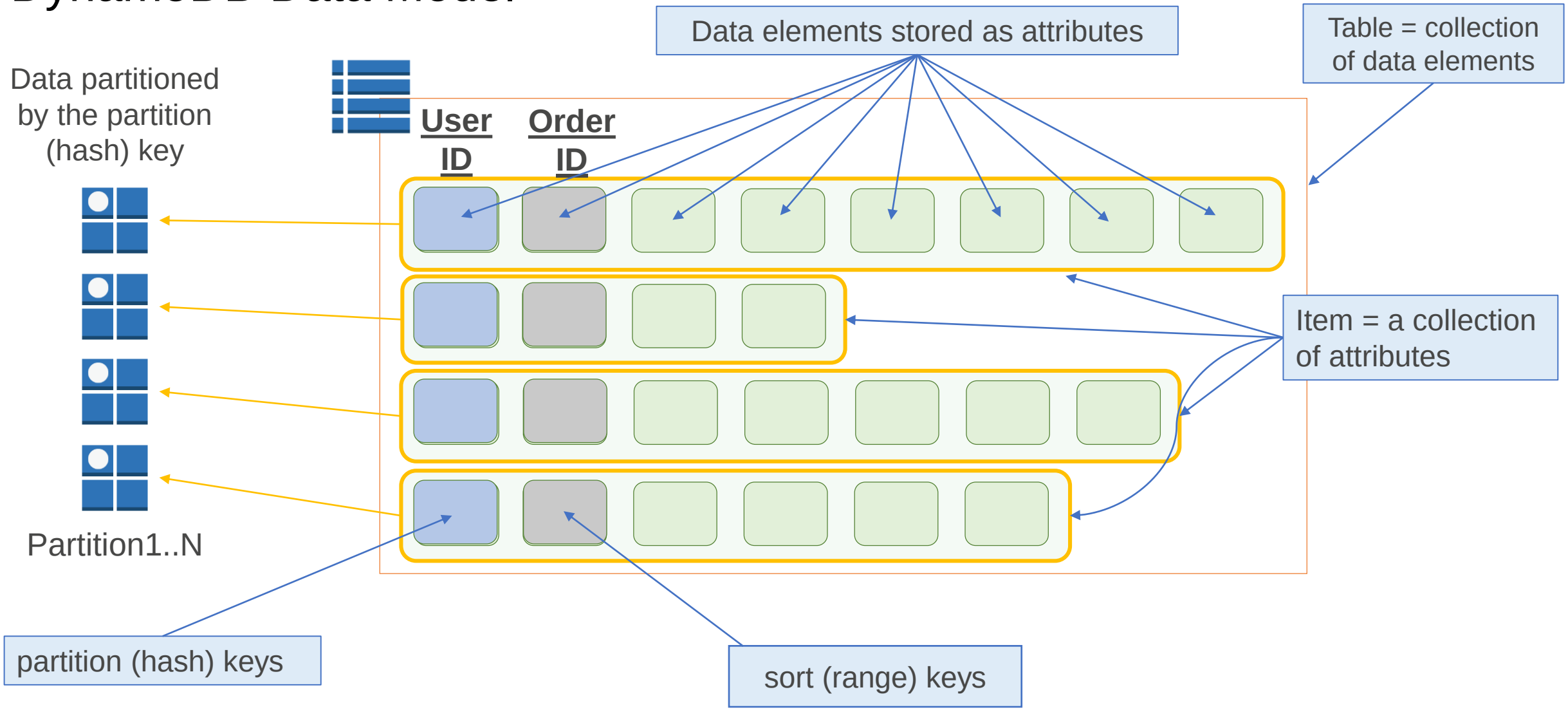
What is Amazon DynamoDB?

- NoSQL database tables (column data model)
- Virtually unlimited storage
- Items may have differing attributes
- Low-latency queries
- Scalable read/write throughput

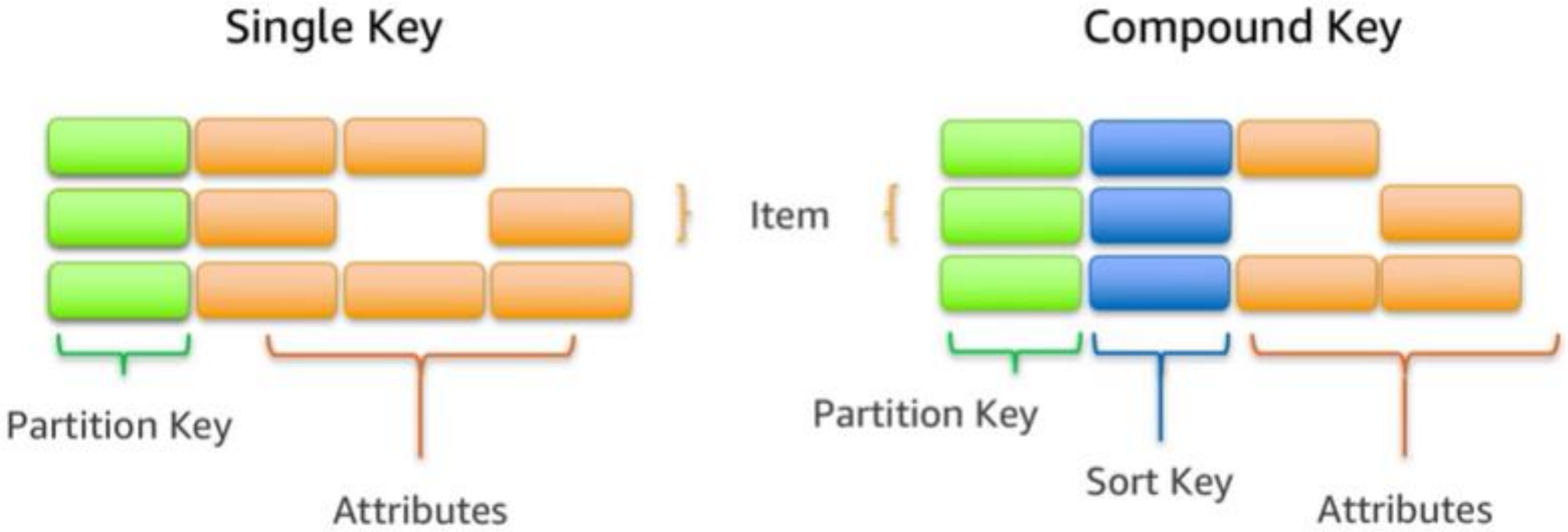
Amazon DynamoDB Core Components

- Tables, items, and attributes are the core DynamoDB components
- DynamoDB supports two different kinds of primary keys: Partition Key or a Partition and Sort Key

DynamoDB Data Model



Items in a Table Must Have a Key



Use Case: Social Network

Users table

Attribute (string, binary, number, string set, etc.)

User	Nicknames
Bob	[Rob, Bobby]
Alice	[Allie]
Caroline	[Carol]
Daniel	[Dan, Danny]

Item

Primary key (partition)

Friends table

User	Friend
Bob	James
Alice	James
Alice	Sarah
Alice	Terrence

Composite primary key

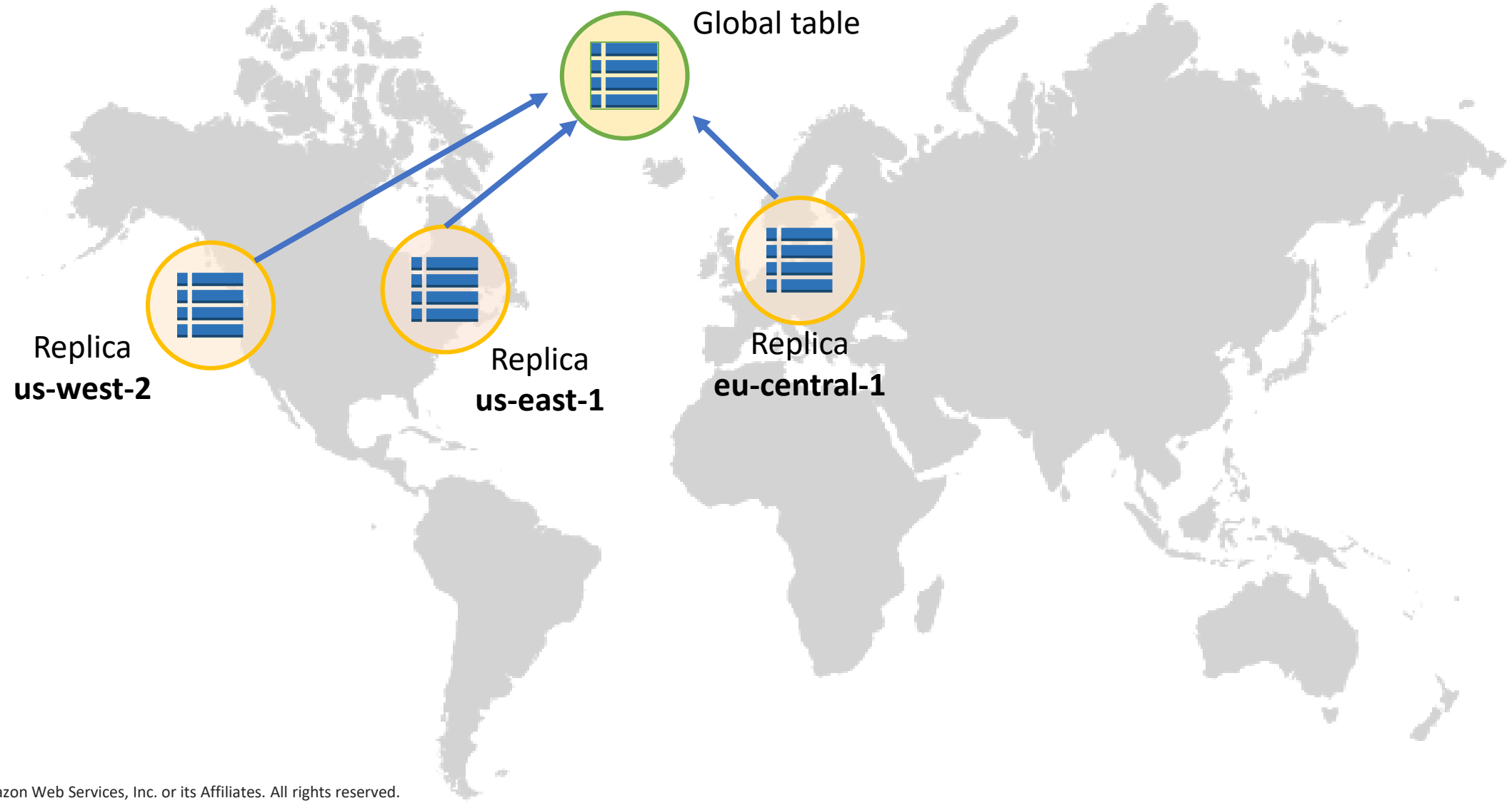
- Partition key = User
- Sort key = Friend
- Partition + sort = unique primary key

Query for Alice returns friends

Amazon DynamoDB Consistency

- DynamoDB synchronously replicates data across three facilities within an AWS Region.
- You can specify, at the time of the read request, whether a read should be **eventually** or **strongly** consistent.

Global Tables



Global Tables

1

user	noteID	note
AAA	**123A**	**Buy coffee**
AAA	**45B**	**Buy milk**

Replica
us-east-1

2

DynamoDB Stream

3

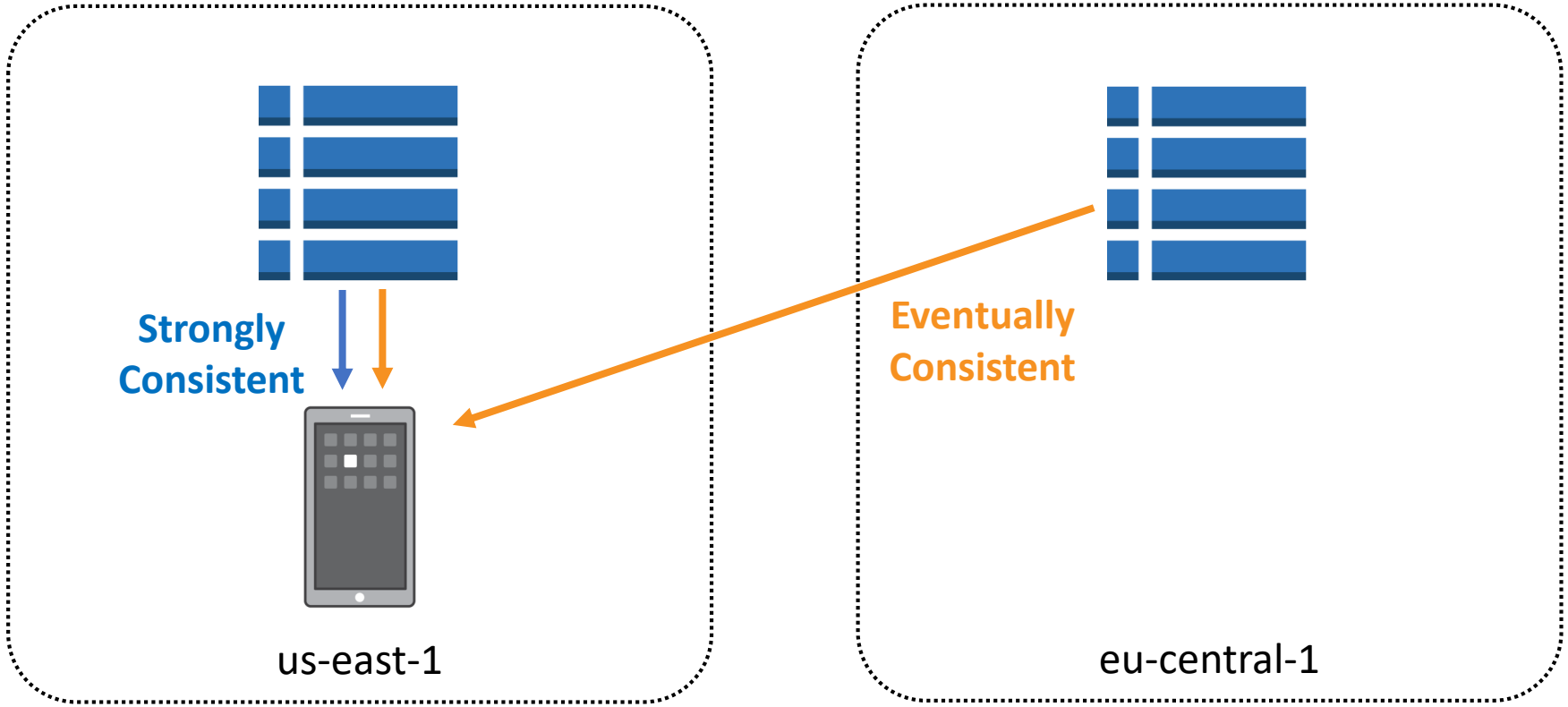
user	noteID	note
AAA	**123A**	**Buy coffee**
AAA	**45B**	**Buy milk**

Replica
eu-central-1

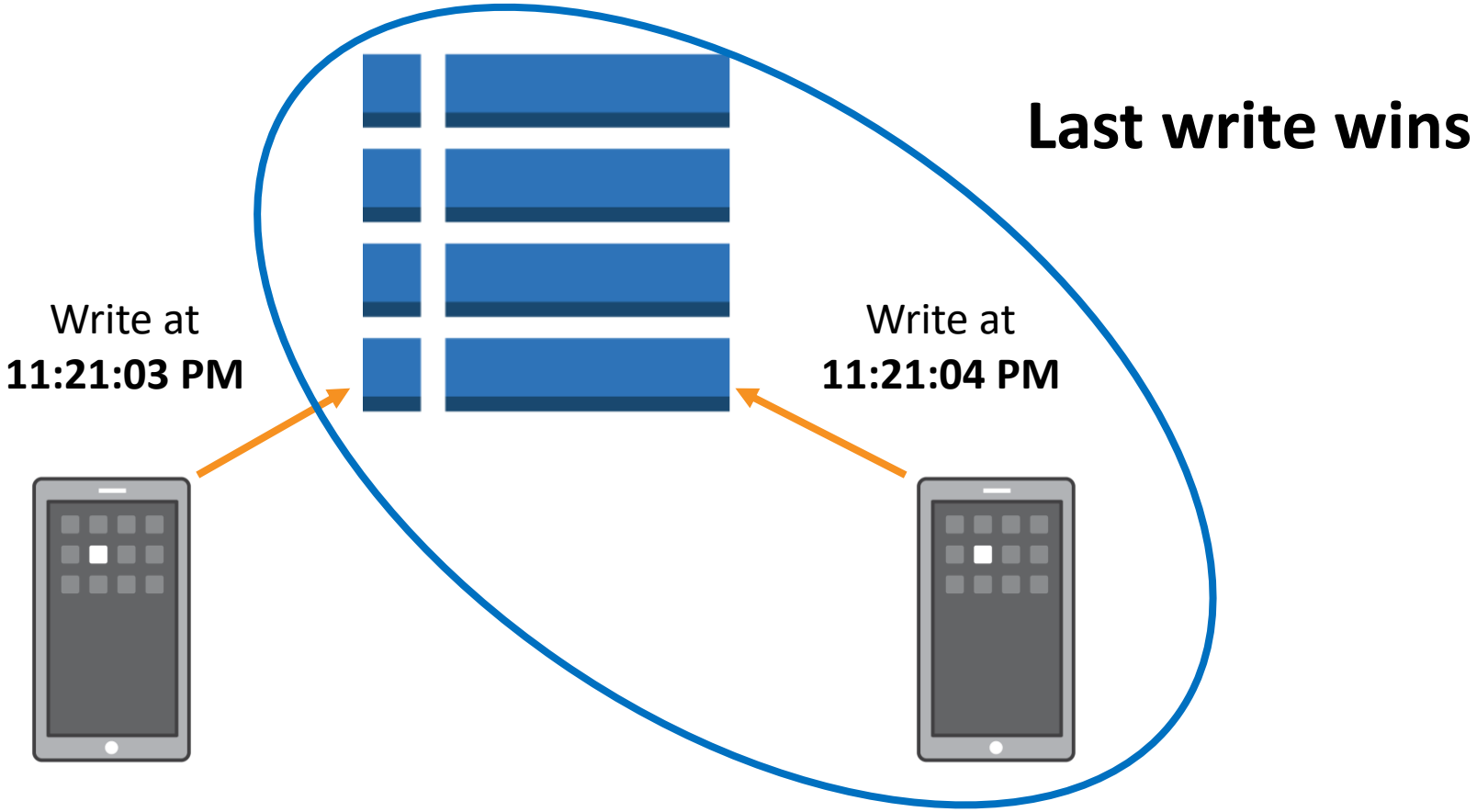
user	noteID	note
AAA	**123A**	**Buy coffee**
AAA	**45B**	**Buy milk**

Replica
us-west-2

Global Tables

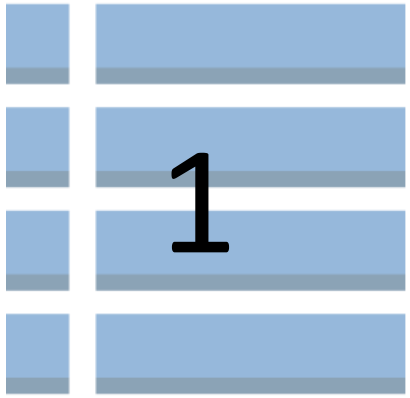


Global Tables

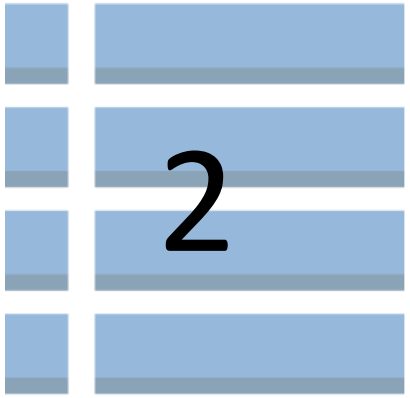


Global Tables

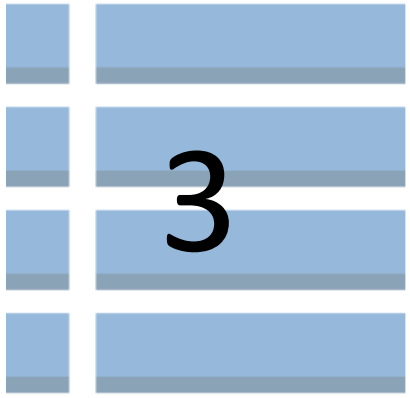
DynamoDB global tables can be created in three steps.



Create DynamoDB table with Streams enabled



Repeat first step for other regions



Define a global table based on tables just created

Learn more.

Amazon DynamoDB Best Practices

- Keep item size small.
- Store metadata in DynamoDB and large BLOBs in Amazon S3.
- Use a table per day, week, month, etc., for storing data about time series.
- Use conditional or Optimistic Concurrency Control (OCC) updates.
- Avoid hot keys and hot partitions.

DynamoDB Overview

- Runs exclusively on SSDs
- Supports document and key-value store models
- The Global Tables feature replicates your DynamoDB tables automatically across your choice of AWS Regions
- Ideal for mobile, web, gaming, ad tech, and IoT applications
- Accessible via the console, the CLI, and simple API calls.

In Review

*DynamoDB is a **fully managed NoSQL database service**.*

- Consistent, single-digit millisecond latency at any scale
- No table size or throughput limits
- Global Tables eliminate the difficulty of replicating data between regions and resolving update conflicts

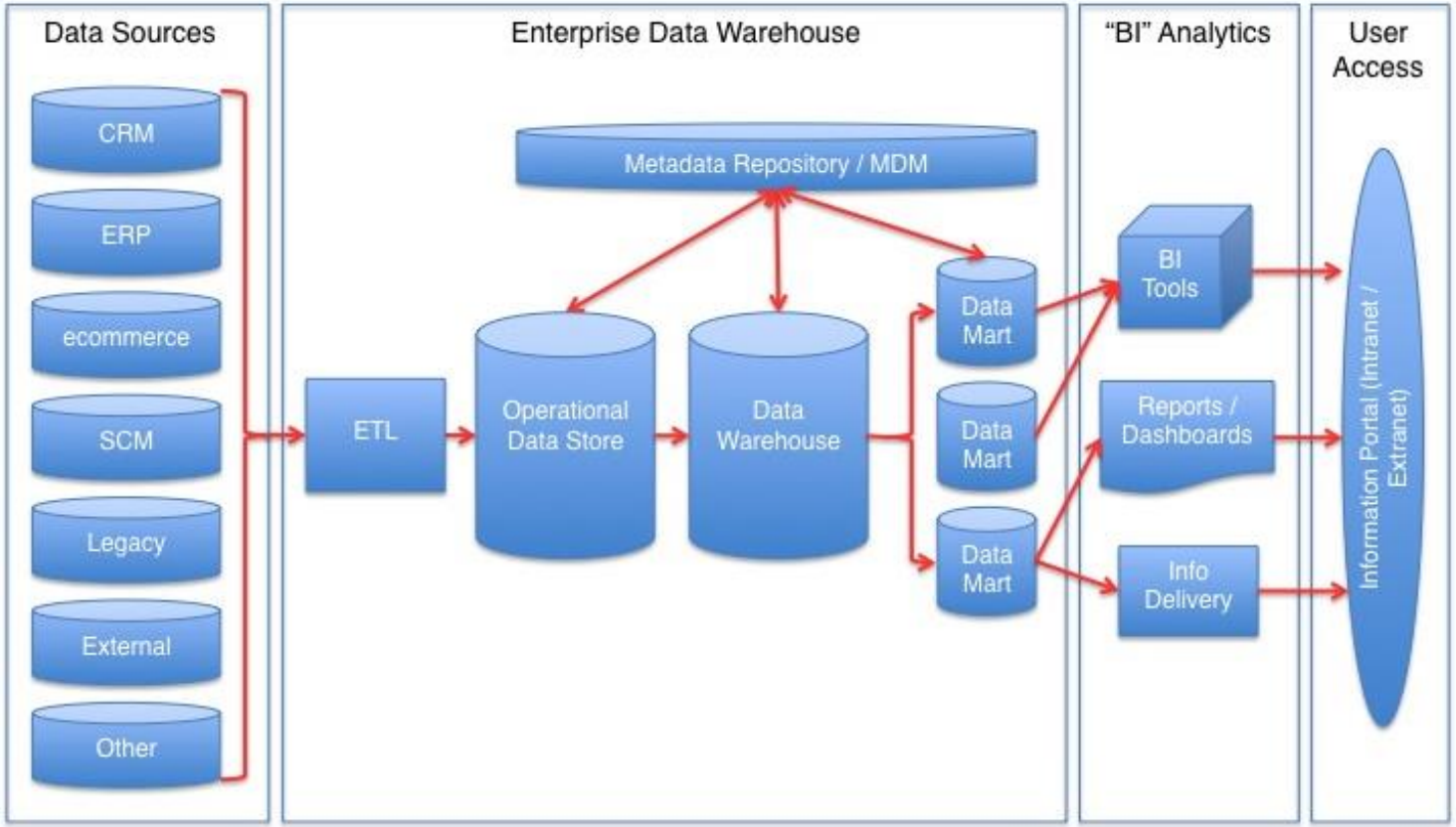
This week

- Relational vs non-relational 'NoSQL' databases
  - High-level Comparison
  - NoSQL data models
- AWS Database Services
  - RDS
  - DynamoDB
  - **Redshift data warehouse**
  - (Aurora)

Amazon Redshift

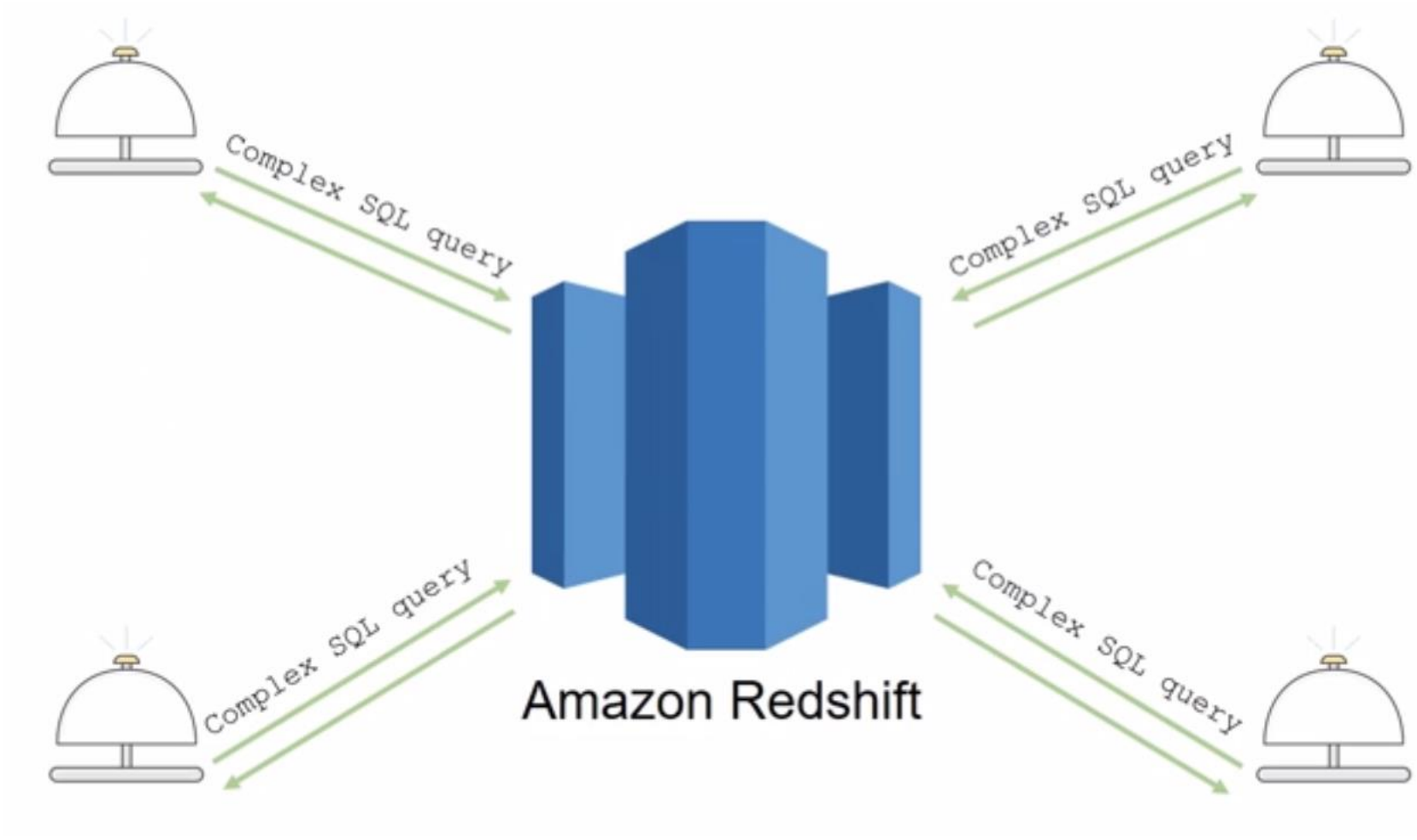
Data Warehousing

- Enterprise data store used for reporting and data analysis → "Business Intelligence"
- Structured data – needs to be cleansed and integrated.

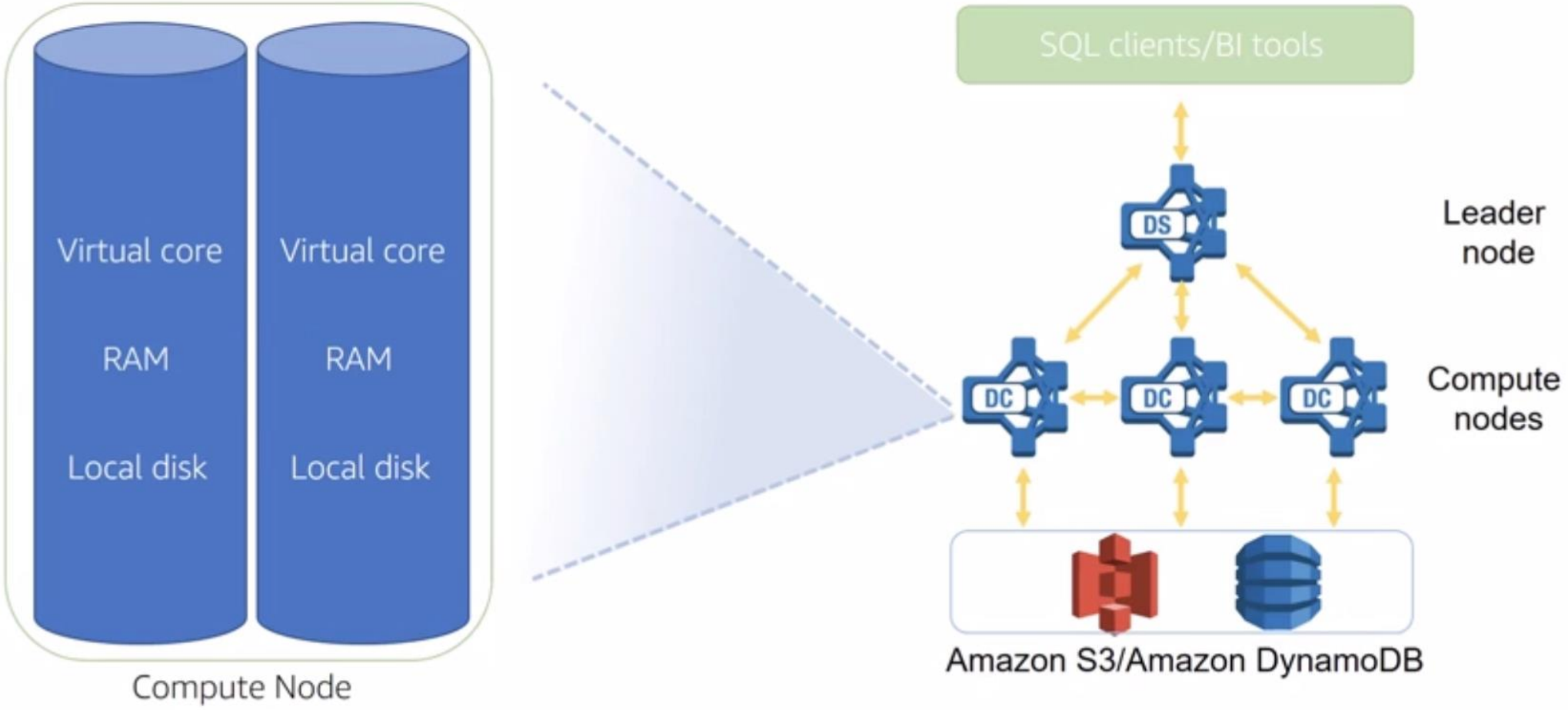


Source:https://en.wikipedia.org/wiki/Data_warehouse

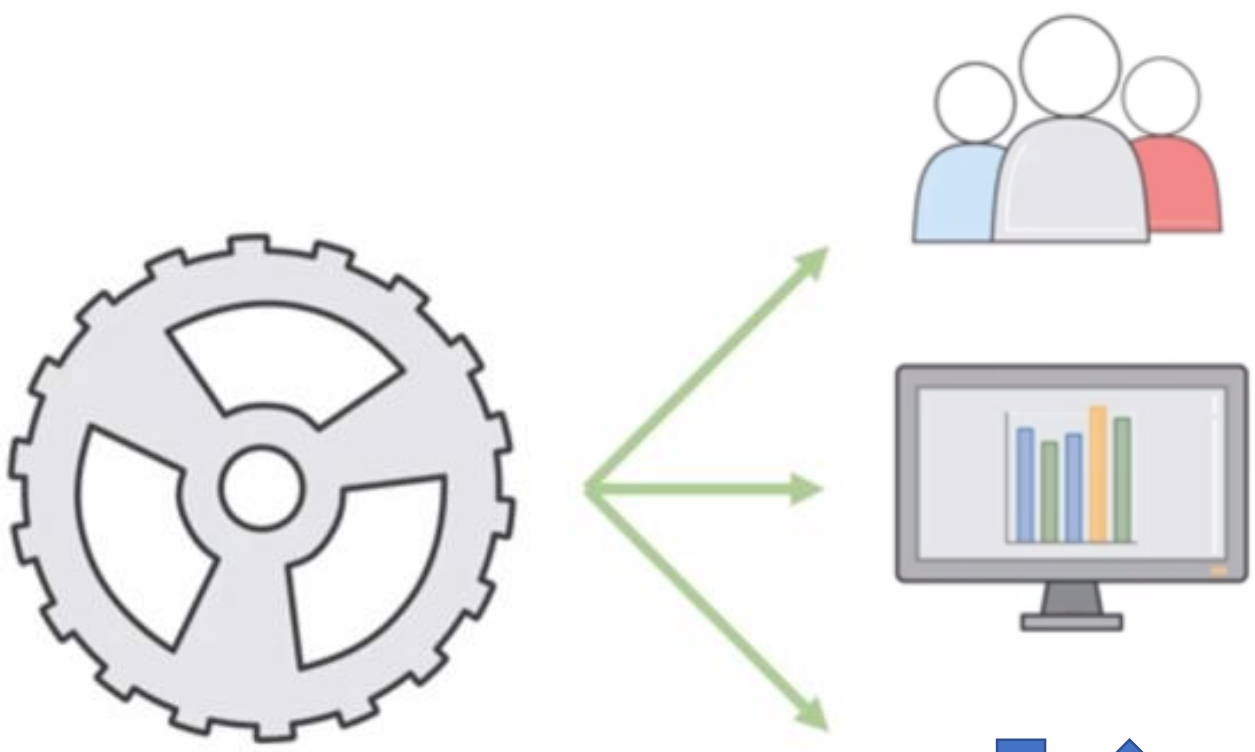
Introduction to Amazon Redshift



Parallel Processing Architecture



Automation and Scaling

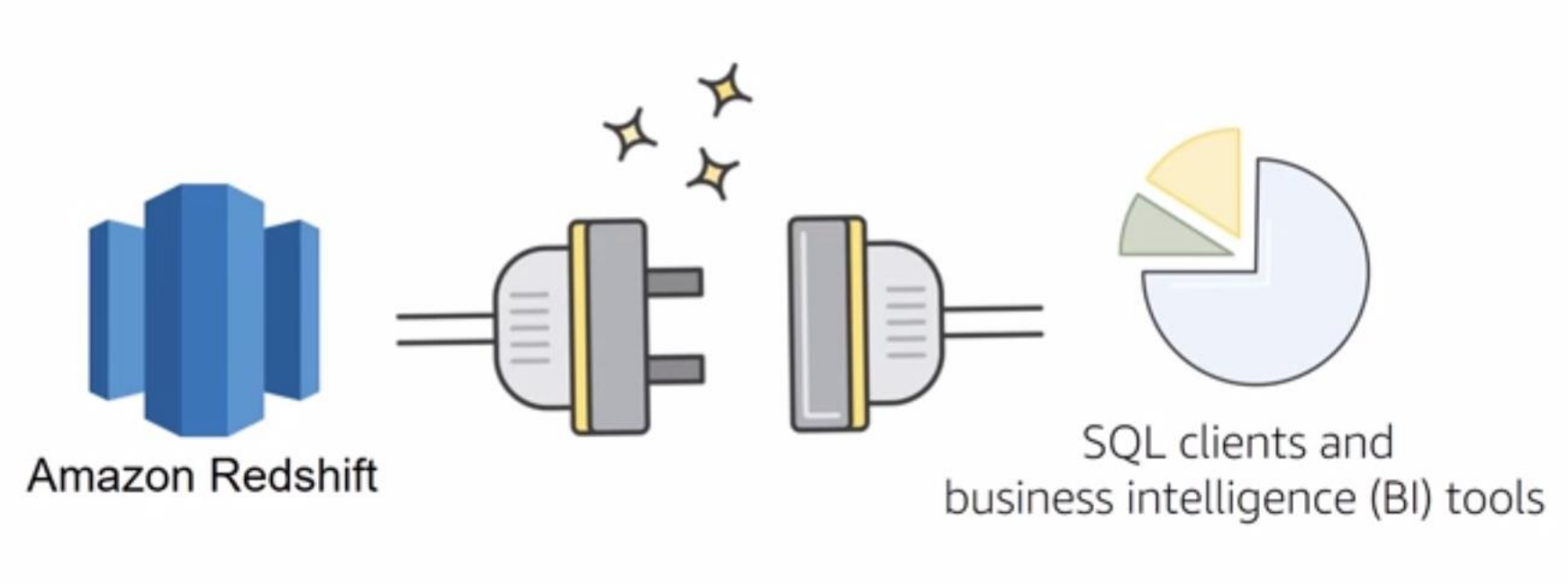


Manage

Monitor

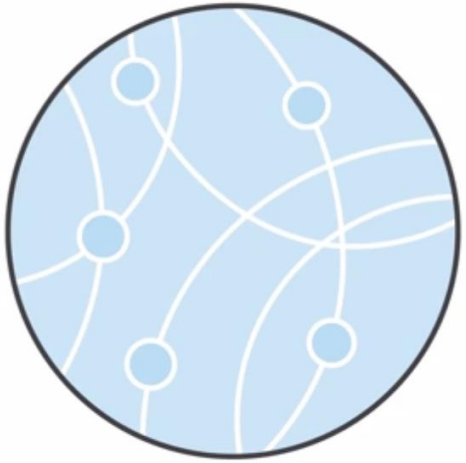
Scale

Compatibility



Amazon Redshift Use Cases

- Enterprise Data Warehouse (EDW)
  - Migrate at a pace that customers are comfortable with
  - Experiment without large upfront cost or commitment
  - Respond faster to business needs
- Big Data
  - Low price point for small customers
  - Managed service for ease of deployment and maintenance
  - Focus more on data and less on database management



Amazon Redshift Use Cases

- Software as a Service (SaaS)
  - Scale the data warehouse capacity as demand grows
  - Add analytic functionality to applications
  - Reduce hardware and software costs by an order of magnitude

In Review

- Fast, fully managed data warehouse service
- Easily scaled with no downtime
- Columnar storage and parallel processing architectures
- Automatically and continuously monitors cluster
- Encryption is built in

This week

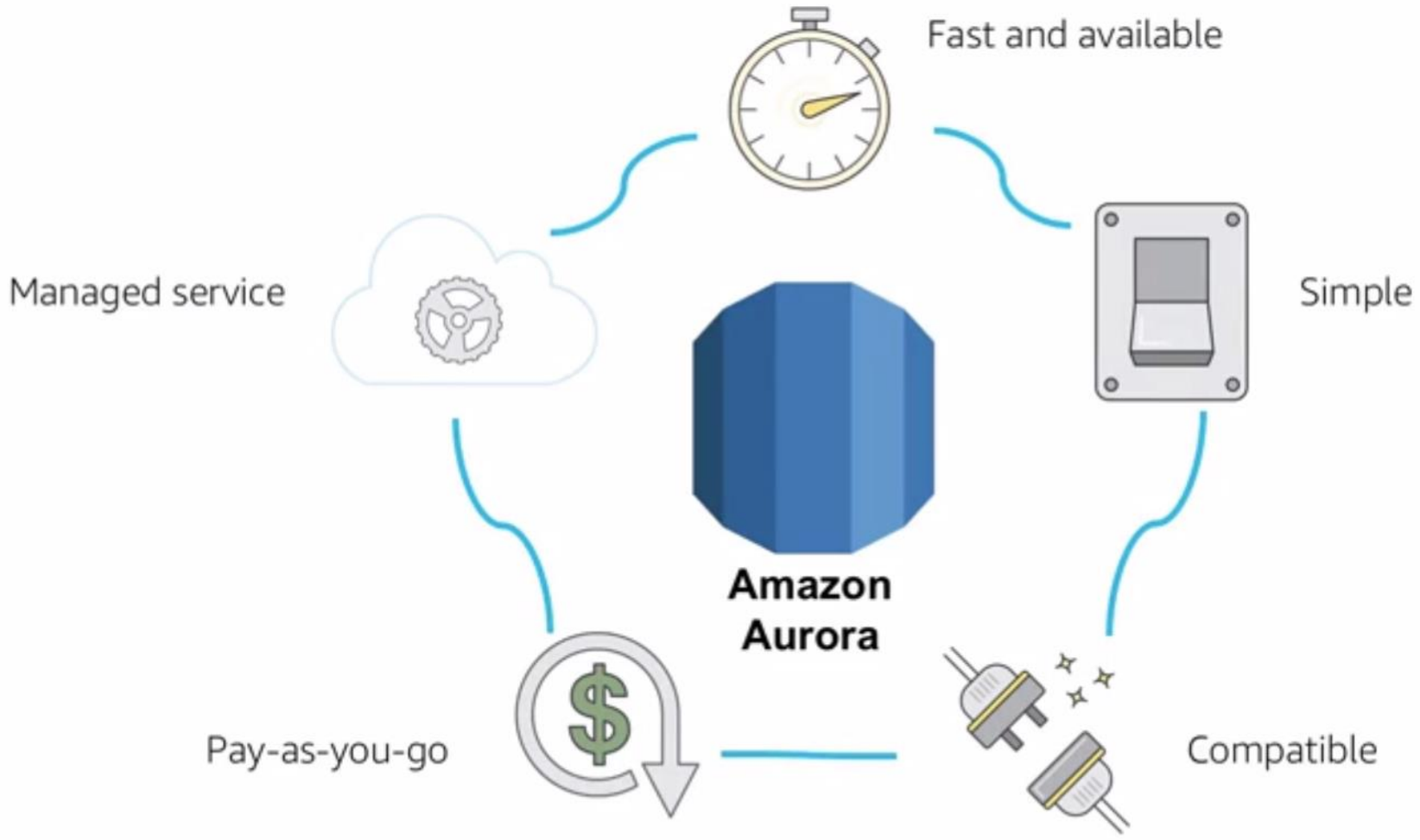
- Relational vs non-relational 'NoSQL' databases
  - High-level Comparison
  - NoSQL data models
- AWS Database Services
  - RDS
  - Dynamo DB
  - Redshift data warehouse
  - **Aurora**

Amazon Aurora

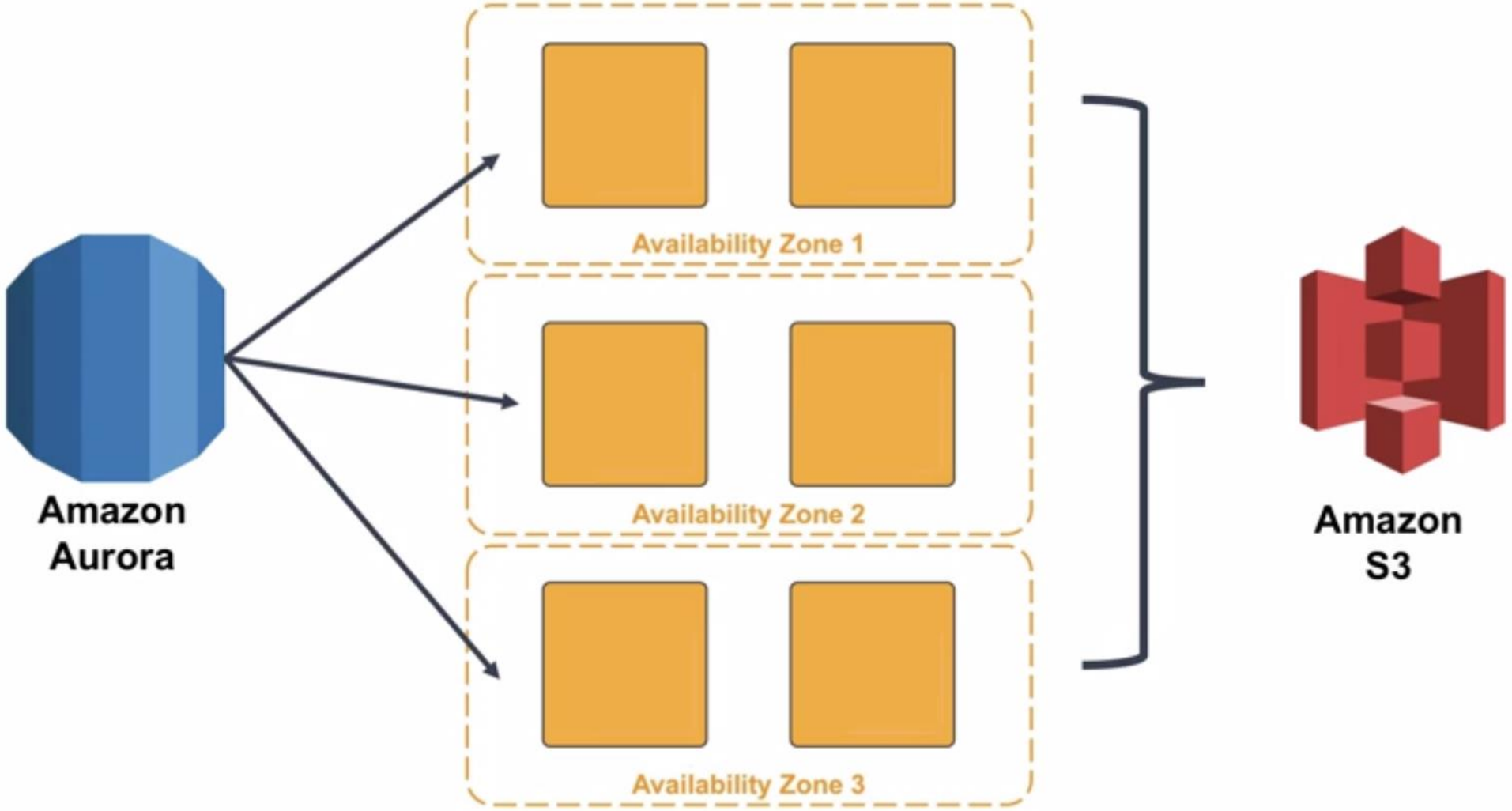
Amazon Aurora

Amazon Aurora

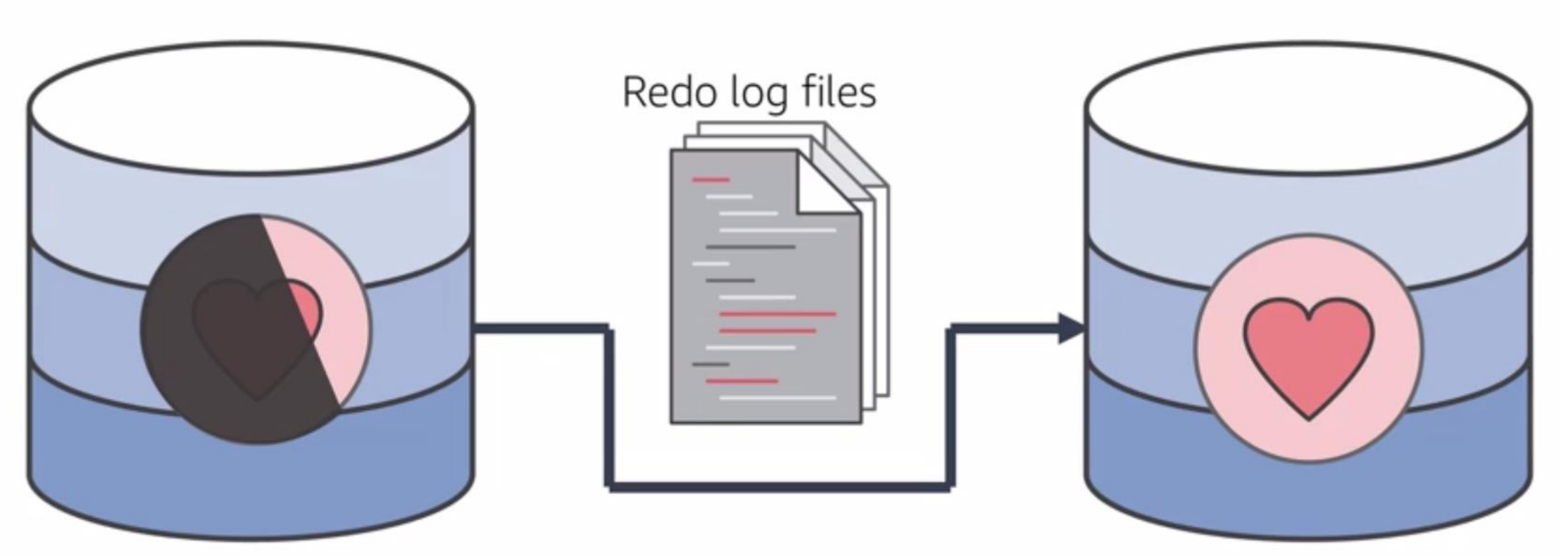
Amazon Aurora Service Benefits



High Availability



Resilient Design



In Review

- High performance and scalability
- High availability and durability
- Multiple levels of security
- Compatible with MySQL and PostgreSQL
- Fully managed