Database

Module overview

Sections

1. Database layer considerations
2. Amazon RDS
3. Amazon DynamoDB
4. Database security controls
5. Migrating data into AWS databases

Database considerations: Scalability

Scalability

Total storage requirements

Object size and type

Durability

How much throughput is needed?

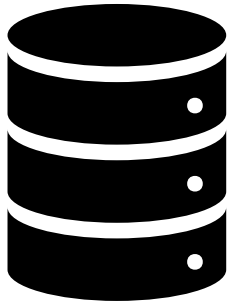
Will the chosen solution be able to scale up later, if needed?

Database considerations: Storage requirements

- Scalability
- **Total storage requirements**
- Object size and type
- Durability

How large does the database need to be?

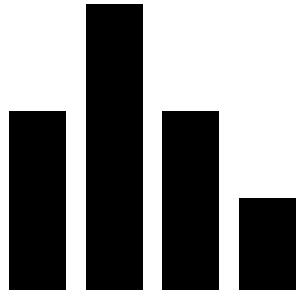
Will it need to store GB, TB, or petabytes of data?



Database considerations: Object size and type

- Scalability
- Total storage requirements
- **Object size and type**
- Durability

Do you need to store simple data structures, large data objects, or both?



Database considerations: Durability

- Scalability
- Total storage requirements
- Object size and type
- **Durability**

What level of **data durability**, **data availability**, and **recoverability** is required?

Do regulatory obligations apply?

Database types

Now that you reviewed key considerations, consider the two categories of database options available:

Relational

Traditional examples:

Microsoft SQL Server
Oracle Database
MySQL

Non-Relational

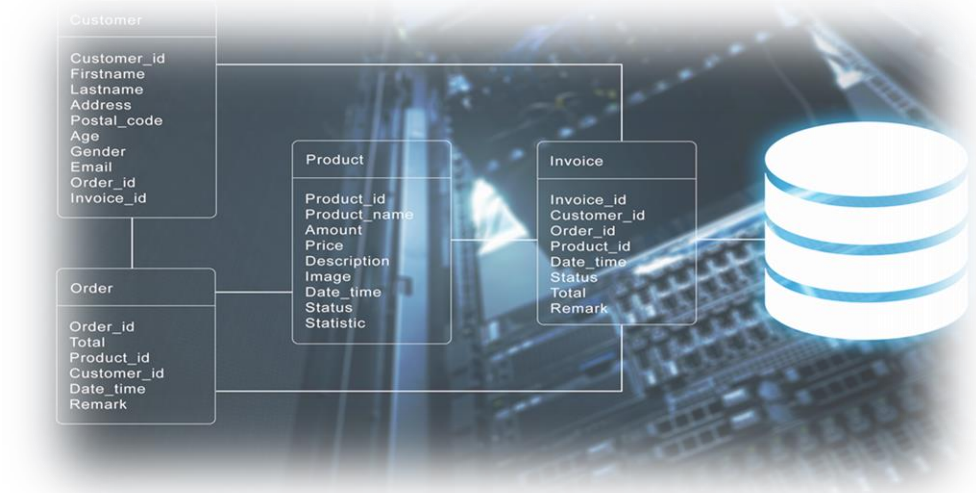
Traditional examples:

MongoDB
Cassandra
Redis

Relational database type

Benefits:

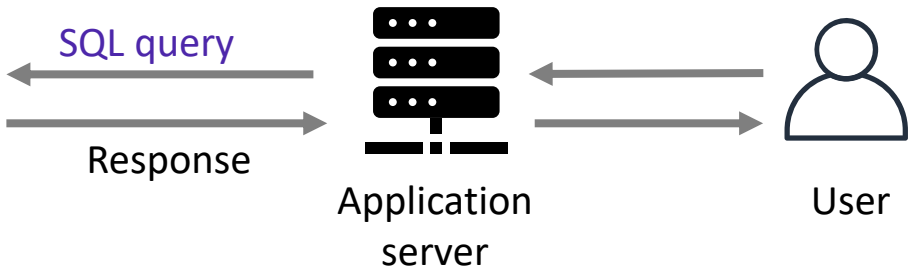
- Ease of use
- Data integrity
- Reduced data storage
- Common language (structured query language, or SQL)



Relational database management system (RDBMS)

Relational is ideal when you:

- Need strict schema rules, ACID compliance, and data quality enforcement
- Do not need extreme read/write capacity
- Do not need extreme performance
  - An RDBMS can be the best, lowest-effort solution



Non-relational database type

Benefits

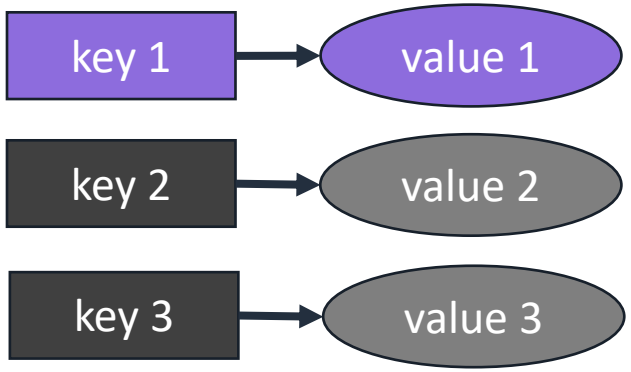
- Flexibility
- Scalability
- High performance
- Highly functional APIs

Non-relational is ideal when:

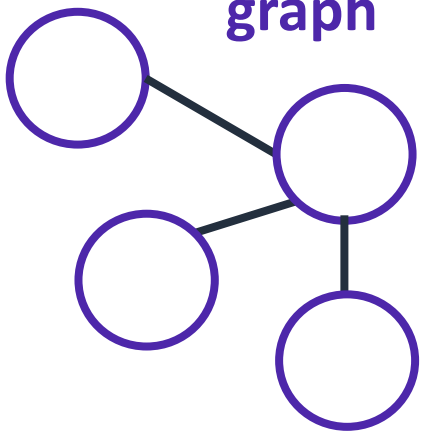
- Database must scale horizontally to handle massive data volume
- Data does not lend itself well to traditional schemas
- Read/write rates exceed what can be economically supported through traditional RDBMS

Example models

key-value



graph



Amazon database options

More database options exist—these options are common examples

Relational databases

Amazon RDS

Amazon Redshift

Amazon Aurora

Non-relational databases

Amazon DynamoDB

Amazon ElastiCache

Amazon Neptune

Focus in this module

key takeaways

- When you choose a database, consider scalability, storage requirements, the type and size of objects to be stored, and durability requirements
- Relational databases have strict schema rules, provide data integrity, and support SQL
- Non-relational databases scale horizontally, provide higher scalability and flexibility, and work well for semistructured and unstructured data

Amazon RDS

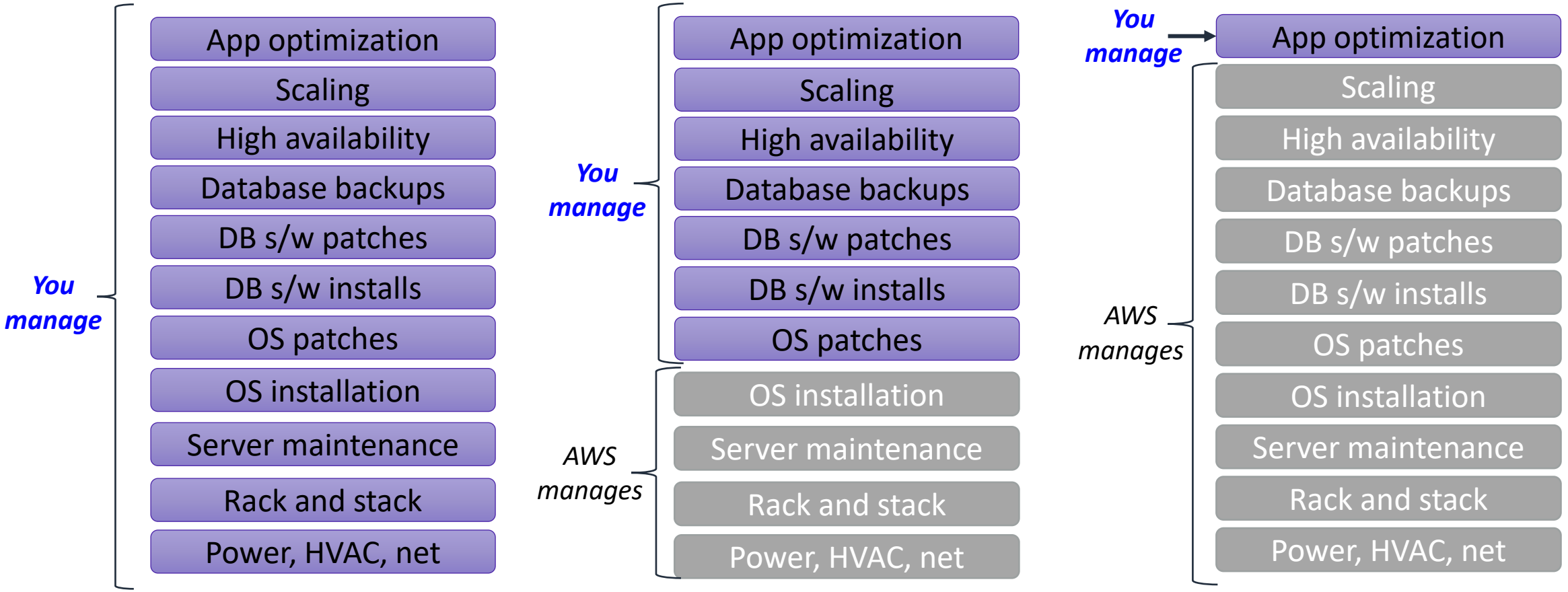
Amazon RDS

Relational

Amazon
RDS

Amazon RDS is a fully managed relational database service.

Advantage of managed AWS database services

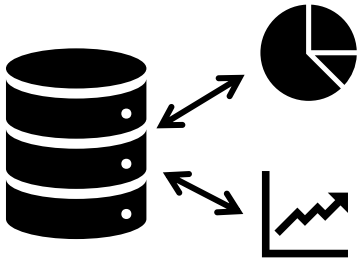


Host database **on-premises**

Host database in **Amazon EC2**

Host database in a **managed AWS database service**

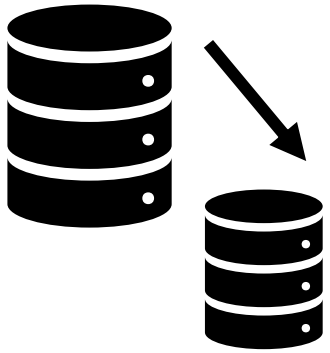
Amazon RDS characteristics



Access pattern

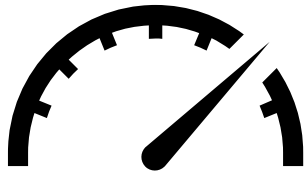
Transactional

Light analytics



Data size

Low-TB range



Performance

Mid to high throughput

Low latency

Business use cases

Transactional

OLAP

Amazon RDS: Uses and database types

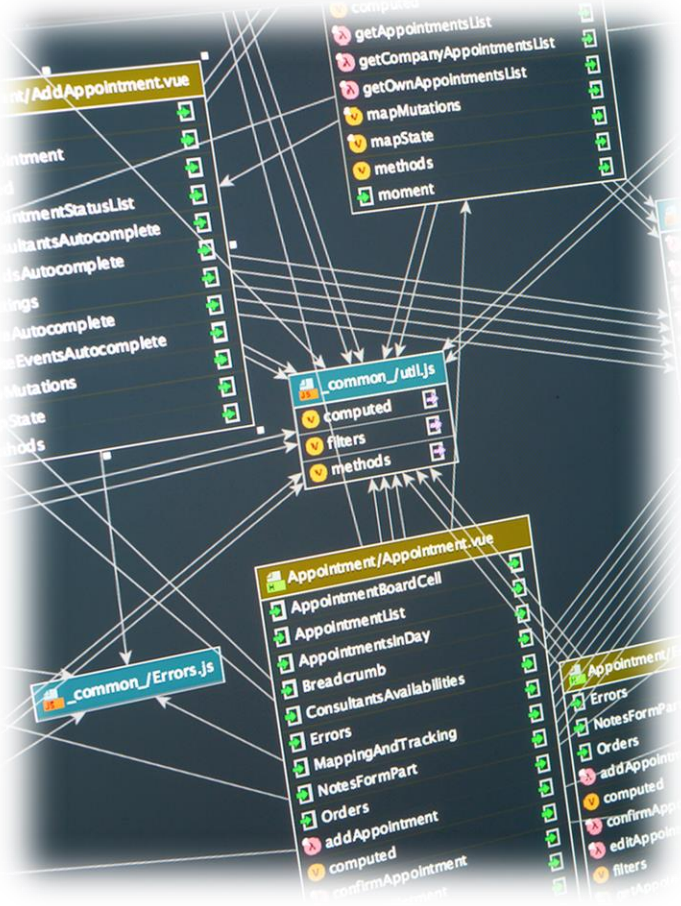
Amazon RDS

Works well for applications that:

- Have more complex data
- Need to combine and join datasets
- Need enforced syntax rules

Six database types supported:

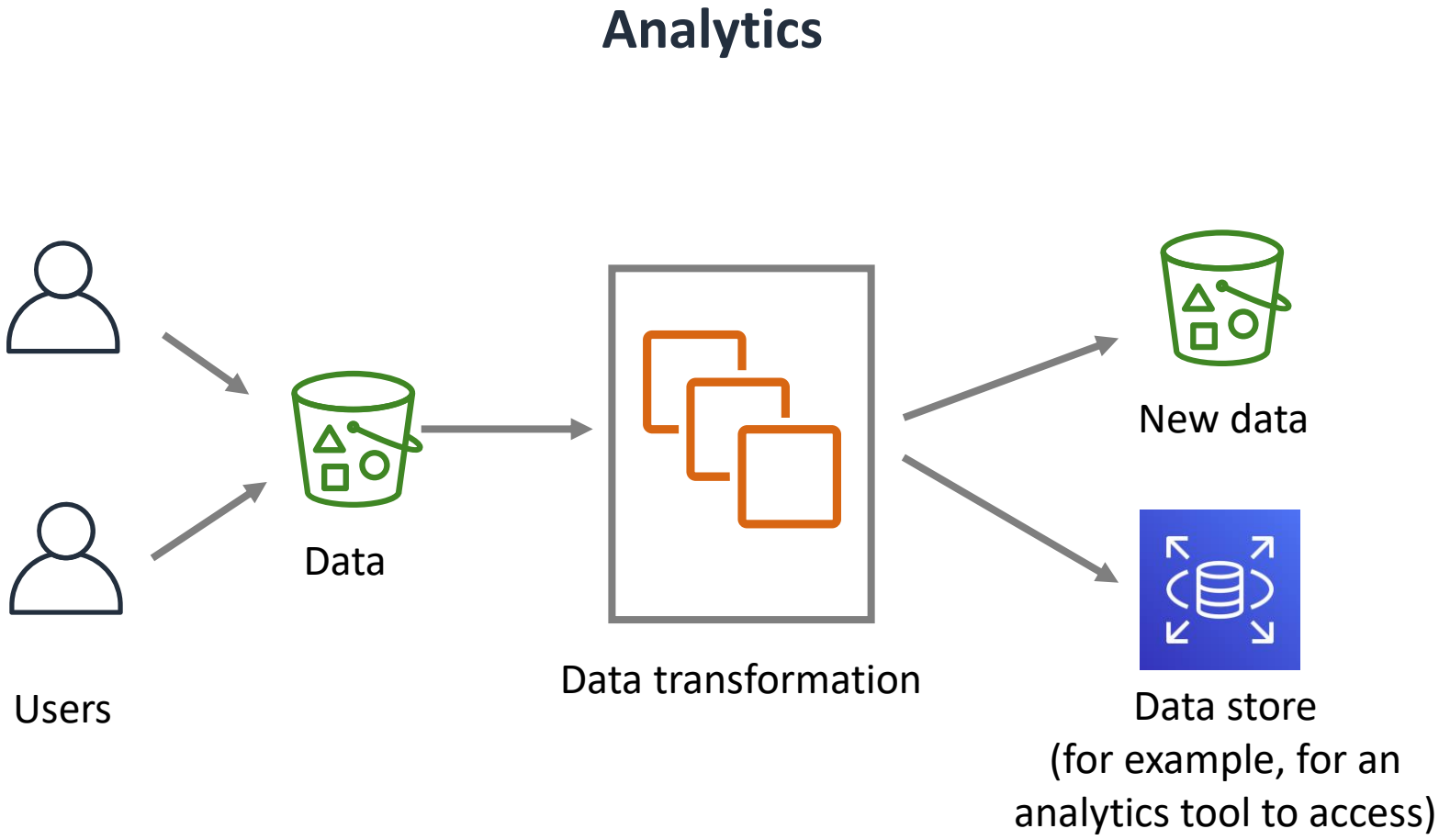
- Microsoft SQL Server
- Oracle
- MySQL
- PostgreSQL
- Aurora
- MariaDB



Database instance sizing

	T family	M family	R family
Type	Burstable instances	General-purpose instances	Memory-optimized instances
Sizing	1 vCPU/1 GB RAM to 8 vCPU 32 GB RAM	2 vCPU/8 GB RAM to 96 vCPU 384 GB RAM	2 vCPU/16 GB RAM to 96 vCPU 768 GB RAM
Networking	Moderate performance	High performance	High performance
Ideal Workload	Smaller or variable	CPU-intensive	Query-intensive, high connection counts
Highlights	T3 can burst above baseline for extra charge	M5 offers up to 96 vCPU	R5 offers up to 96 vCPU 768 GiB RAM

Amazon RDS: Example use case



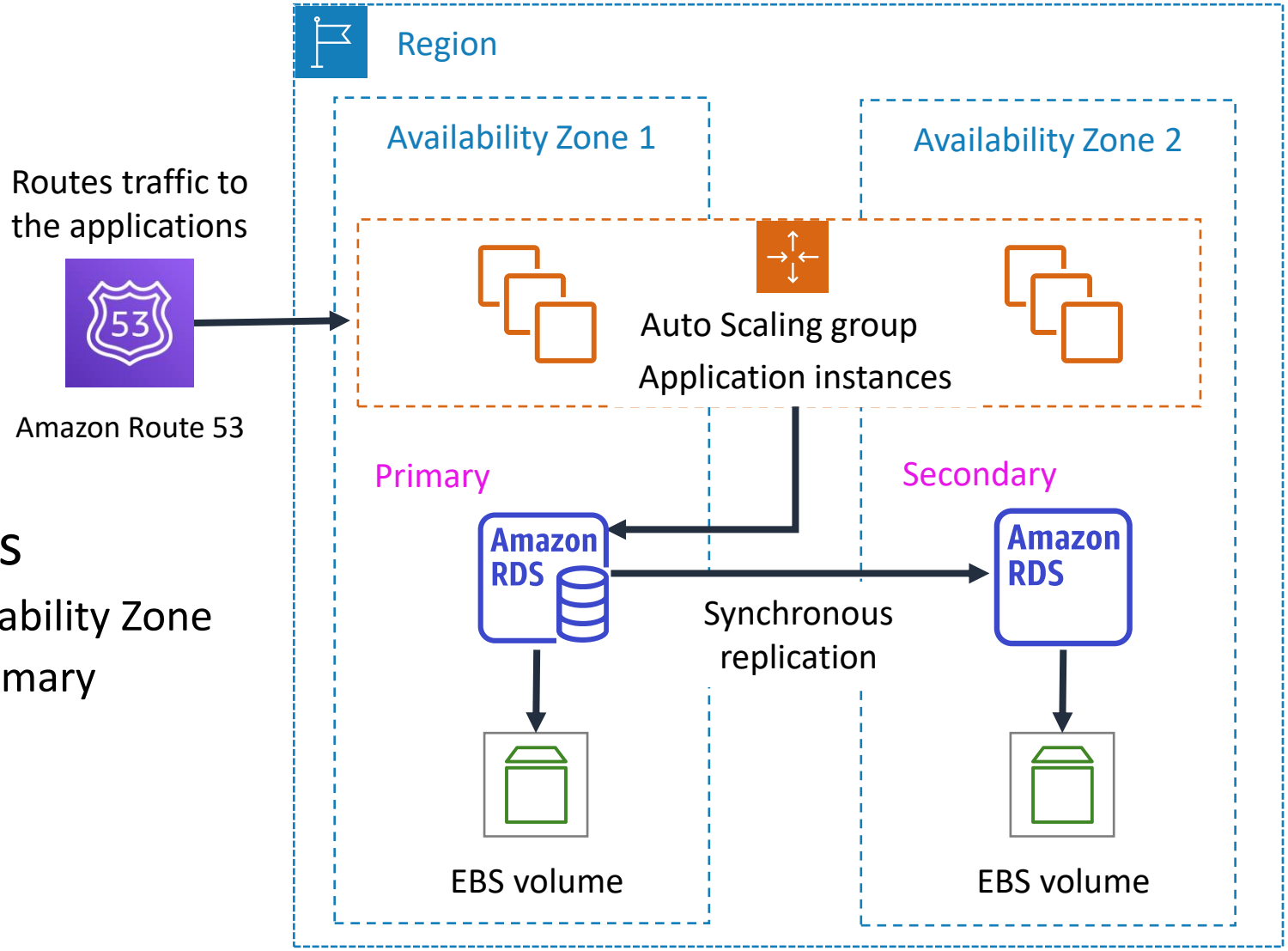
Multi-AZ deployment for high availability

Benefits

- Enhanced durability
- Increased availability
- Fail over to standby occurs automatically

Automated failover conditions

- Loss of availability in primary Availability Zone
- Loss of network connectivity to primary
- Compute unit failure on primary
- Storage failure on primary



Read replicas for performance

Benefits

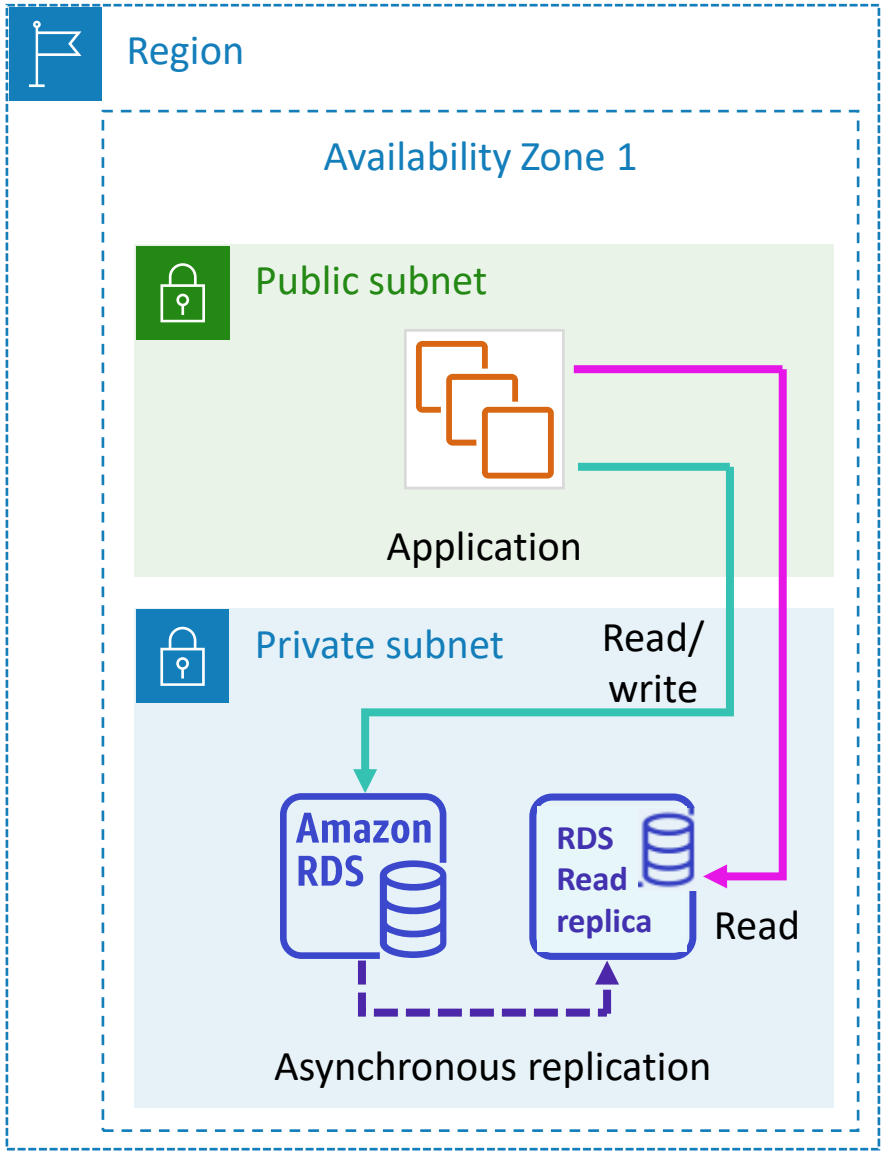
- Enhanced performance
- Increased availability
- Designed for security

Supported by

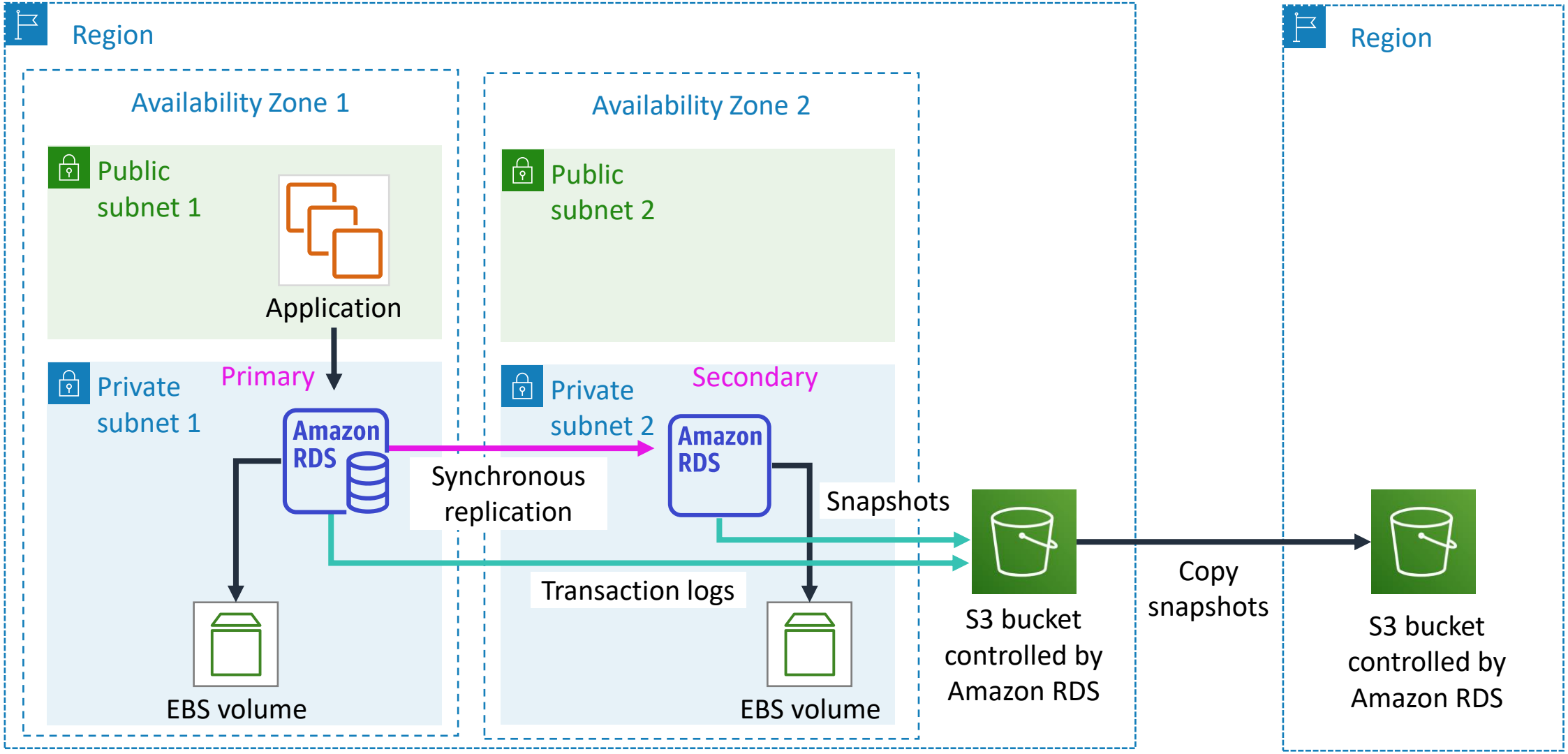
- MySQL
- MariaDB
- PostgreSQL
- Oracle

Limits

- Five read replicas per primary
- For strict read-after-write consistency, read from the primary



Amazon RDS backup solution



Amazon Aurora

Amazon Aurora

Amazon Aurora is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine.

- Used for online transactional processing (OLTP)
- Provides up to five times the throughput of MySQL*
- Provides up to three times the throughput of PostgreSQL*
- Replicates data six ways across three Availability Zones
- Requires little change to your existing application

* Benchmarking details are available for MySQL and. PostgreSQL.

Amazon Redshift

Amazon Redshift

Amazon Redshift is a data warehousing service.

- Is used for online analytics processing (OLAP)
- Stores very large datasets
  - Store highly structured, frequently accessed data in Amazon Redshift
  - Can also store exabytes of structured, semistructured, and unstructured data in Amazon S3

key takeaways

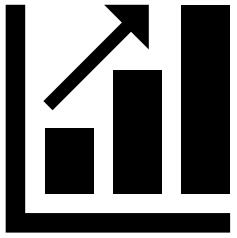
- Managed AWS database services handle administration tasks so you can focus on your applications
- Amazon RDS supports Microsoft SQL Server, Oracle, MySQL, PostgreSQL, Aurora, and MariaDB
- Amazon RDS Multi-AZ deployments provide high availability with automatic failover
- You can have up to five read replicas per primary database to improve Amazon RDS performance
- Amazon Aurora is a fully managed, MySQL- and PostgreSQL-compatible, relational database engine
- Amazon Redshift is a relational database offering for data warehousing

Amazon DynamoDB

Amazon DynamoDB

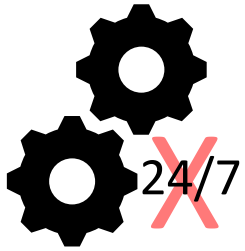
Amazon DynamoDB

A fully managed non-relational key-value and document database service.



Performance at any scale

Extreme horizontal scaling capability



Serverless

Event-driven programming (serverless computing)

Enterprise-ready

Encryption, access controls, backups

Amazon DynamoDB characteristics

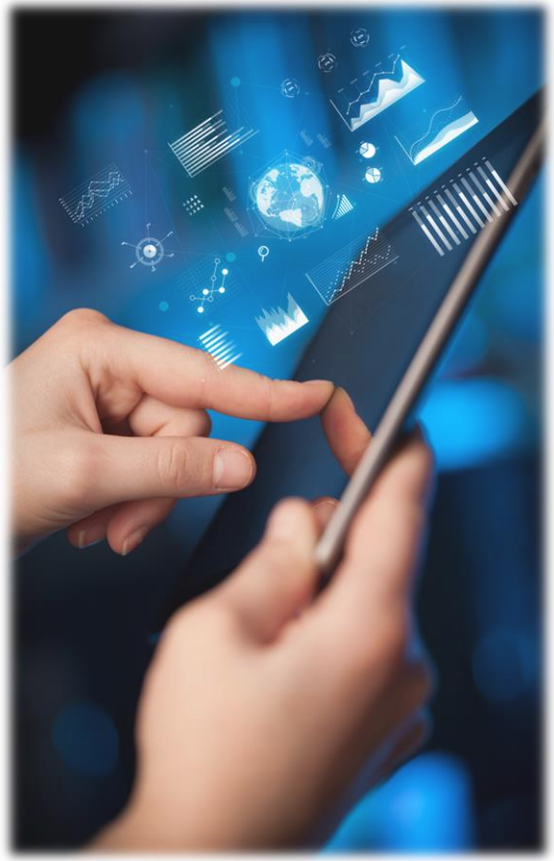
Amazon DynamoDB

Works well for applications that:

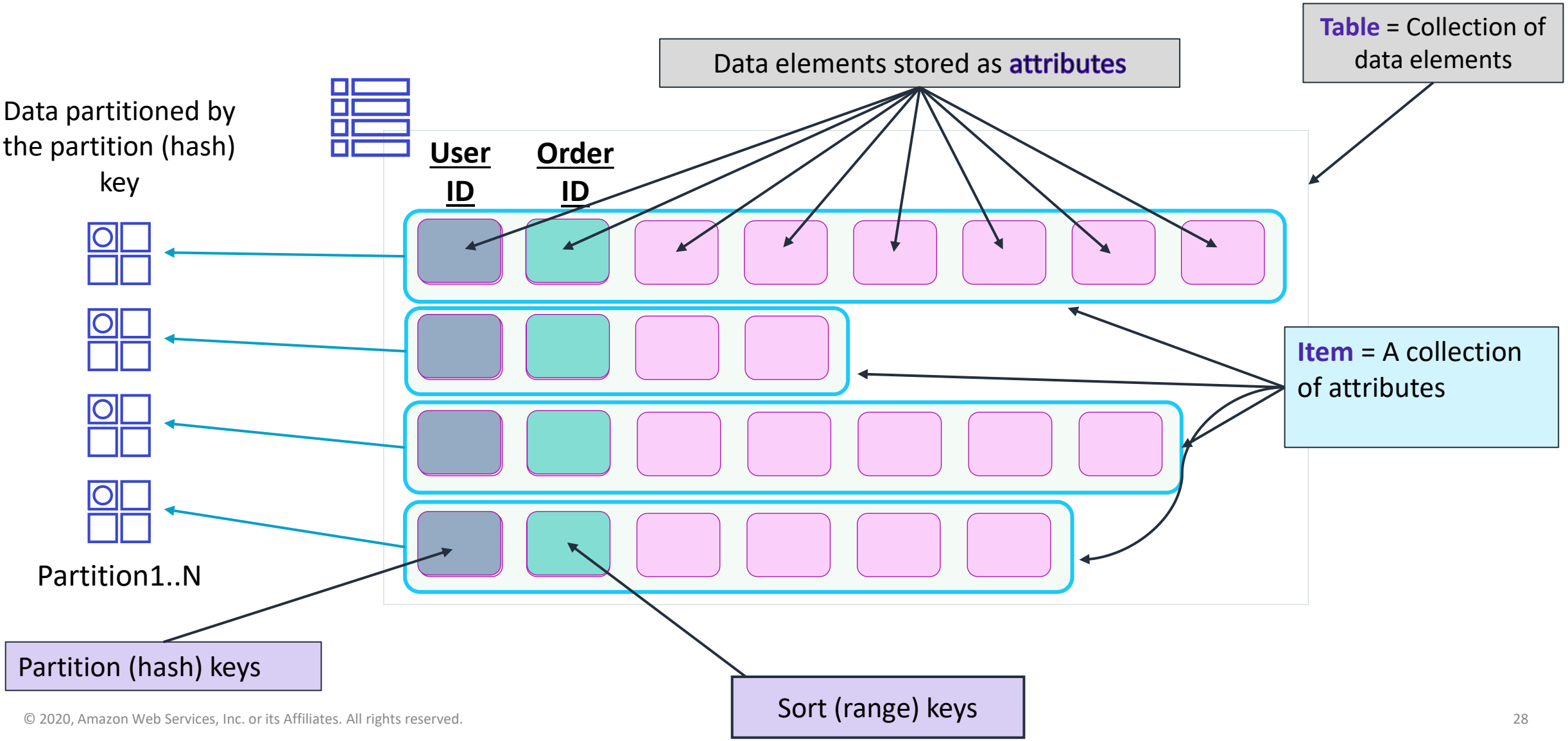
- Have simple high-volume data (high-TB range)
- Must scale quickly
- Don't need complex joins
- Require ultra-high throughput and low latency

Key features

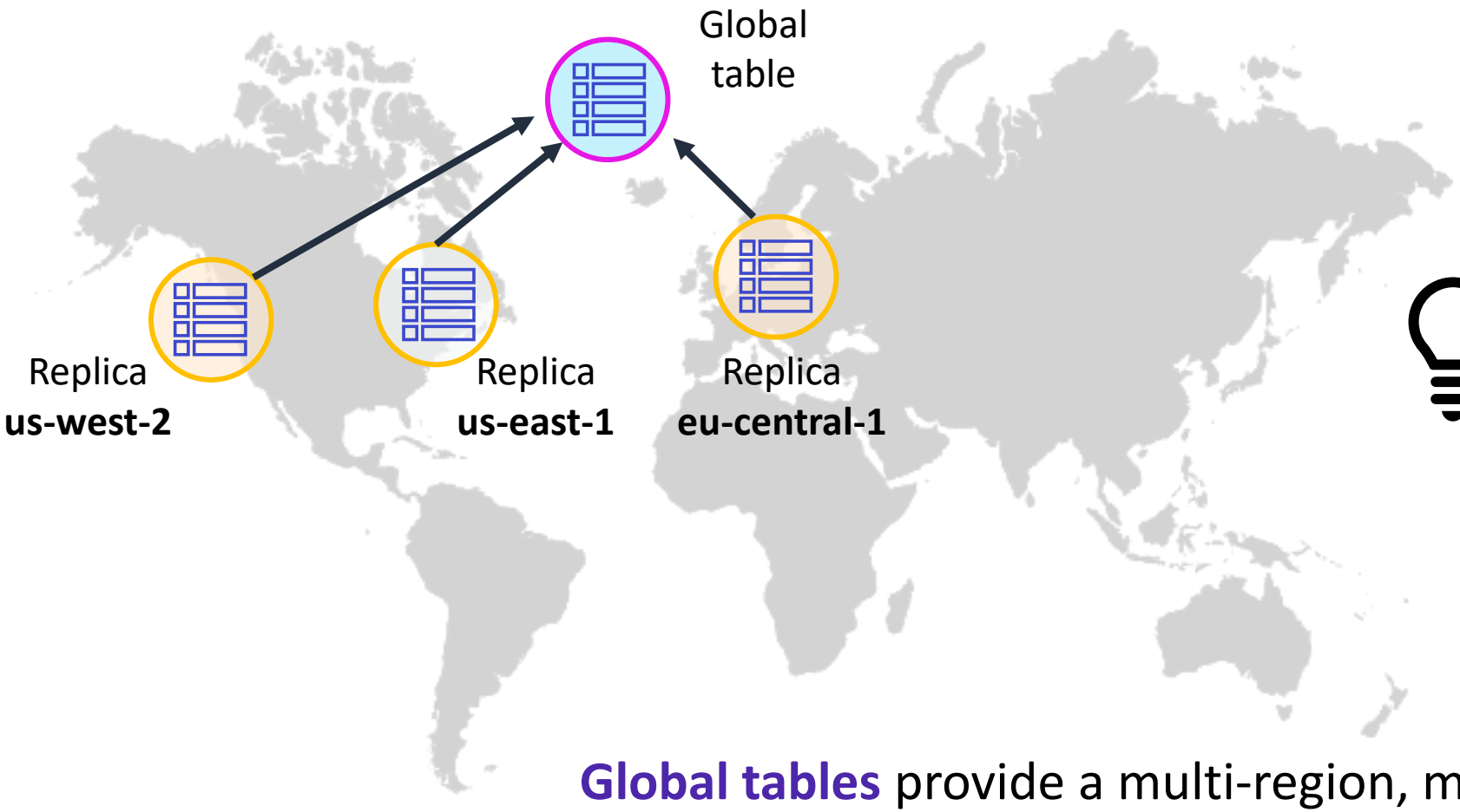
- NoSQL tables
- Items can have differing attributes
- In-memory caching
- Support for peaks of more than 20 million requests per second



Amazon DynamoDB data model



Amazon DynamoDB global tables

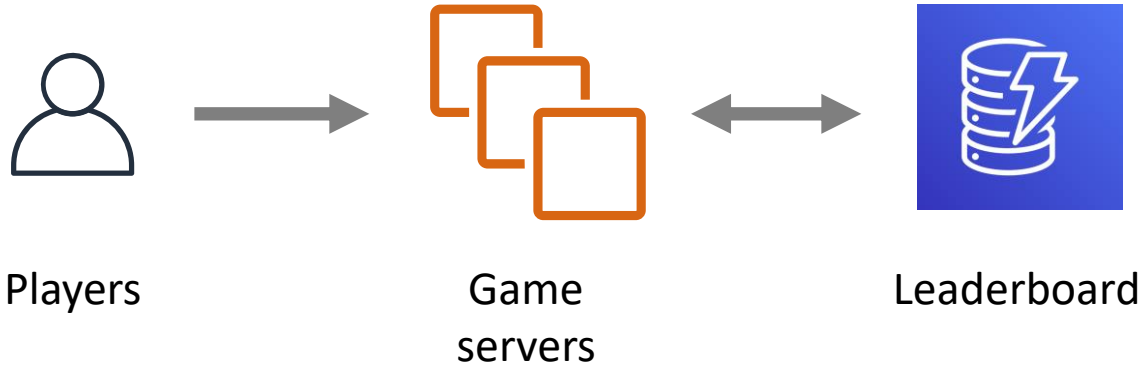


Fast local read/write performance for global applications.

Global tables provide a multi-region, multi-master database.

Amazon DynamoDB use case 1

Leaderboards and Scoring

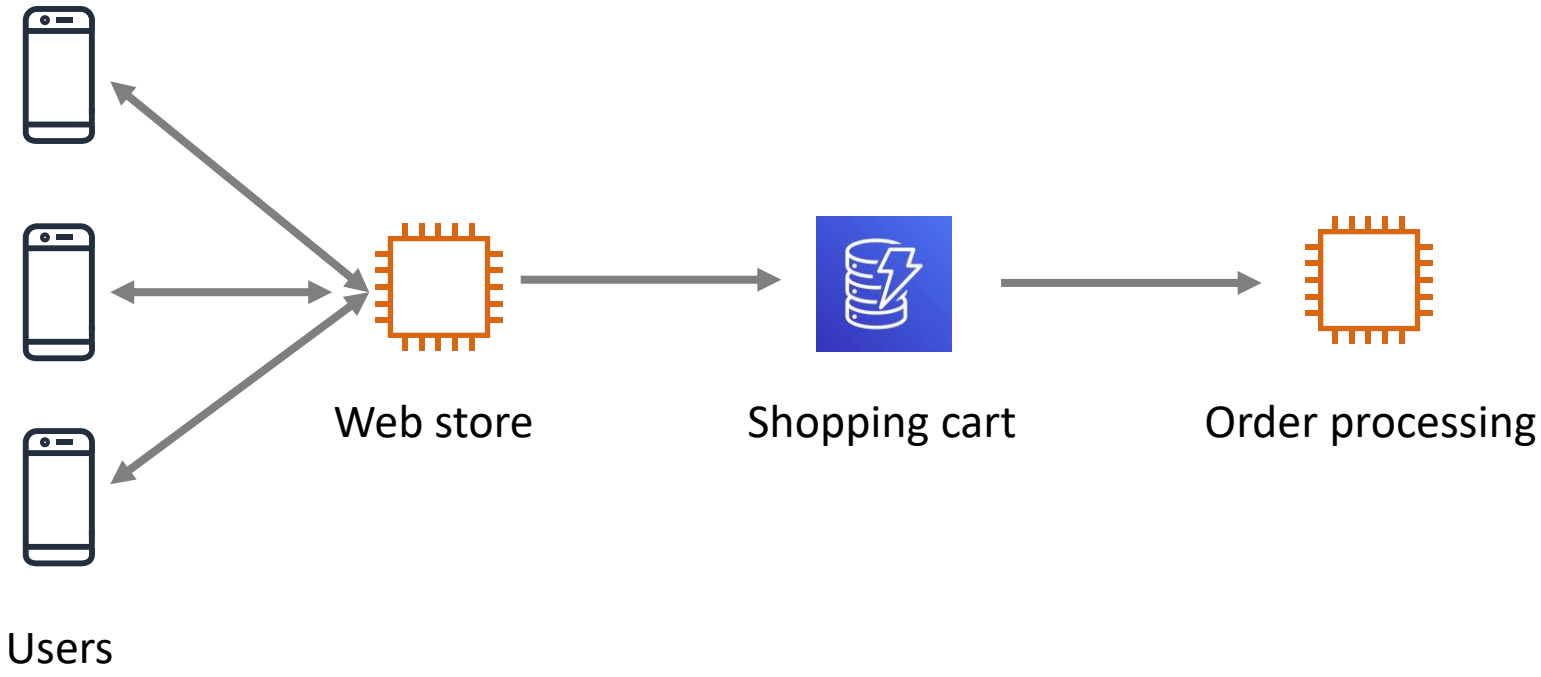


GameScores

UserId	GameTitle	TopScore	TopScoreDateTime	Wins	Losses	
"101"	"Galaxy Invaders"	5842	"2015-09-15:17:24:31"	21	72	...
"101"	"Meteor Blasters"	1000	"2015-10-22:23:18:01"	12	3	...
"101"	"Starship X"	24	"2015-08-31:13:14:21"	4	9	...
"102"	"Alien Adventure"	192	"2015-07-12:11:07:56"	32	192	...
"102"	"Galaxy Invaders"	0	"2015-09-18:07:33:42"	0	5	...
"103"	"Attack Ships"	3	"2015-10-19:01:13:24"	1	8	...
"103"	"Galaxy Invaders"	2317	"2015-09-11:06:53:00"	40	3	...
"103"	"Meteor Blasters"	723	"2015-10-19:01:13:24"	22	12	...
"103"	"Starship X"	42	"2015-07-11:06:53:00"	4	19	...
...	...	...	...	...	...	

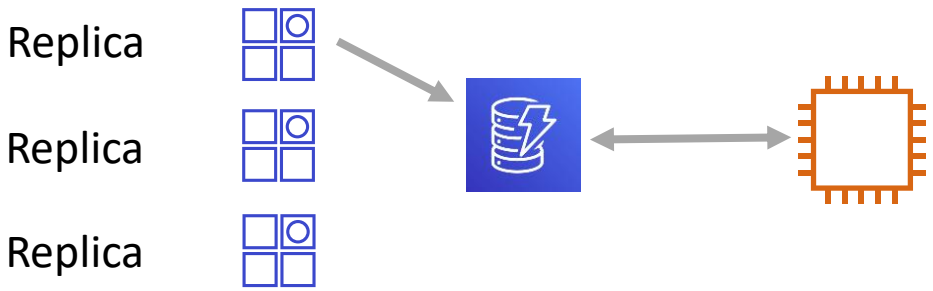
Amazon DynamoDB use case 2

Temporary Data (Online Cart)



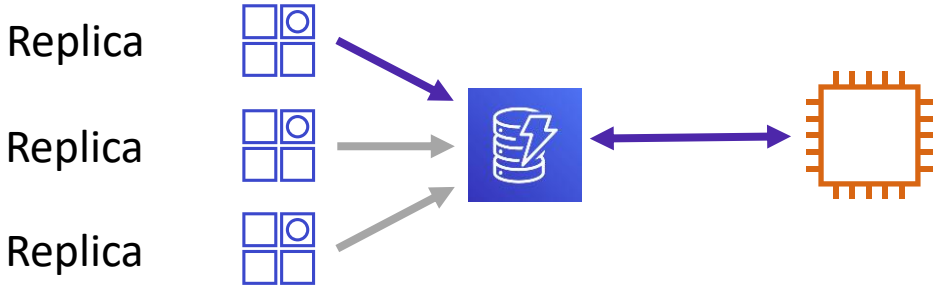
Amazon DynamoDB consistency options

Eventually consistent



The default setting. All copies of data usually reach consistency within 1 second.

Strongly consistent



This feature is optional. Use for applications that require all reads to return a result that reflects all writes before the read.

key takeaways

- Amazon DynamoDB is a fully managed non-relational key-value and document NoSQL database service.
- DynamoDB is serverless, provides extreme horizontal scaling and low latency.
- DynamoDB global tables ensure that data is replicated to multiple Regions.
- DynamoDB provides eventual consistency by default (in general, it is fully consistent for reads 1 second after the write). Strong consistency is also an option.

Database security controls

aws academy

Securing Amazon RDS databases

Recommendations

- Run the RDS instance in a virtual private cloud (VPC)
  - Provides service isolation and IP firewall protection
- Use AWS Identity and Access Management (IAM) policies for authentication and access
  - Permissions determine who is allowed to manage Amazon RDS resources
- Use security groups to control what IP addresses or EC2 instances can connect to your databases
  - By default, network access is disabled
- Use Secure Sockets Layer (SSL) for encryption in transit
- Use Amazon RDS encryption on DB instances and snapshots to secure data at rest
- Use the security features of your DB engine to control who can log in to the databases on a DB instance
- Configure event notifications to alert you when important Amazon RDS events occur

Amazon RDS

Securing Amazon DynamoDB

Recommendations

- Use **IAM roles** to authenticate access
- Use **IAM policies**
  - To define fine-grain access permissions to use DynamoDB APIs
  - Define access at the table, item, or attribute level
  - Follow the **principle of granting least privilege**
- Configure **VPC endpoints**
  - Prevents connection traffic from traversing the open internet
  - VPC endpoint policies allow you to control and limit API access to a DynamoDB table
- Consider **client-side encryption**
  - Encrypt data as close as possible to its origin

Amazon DynamoDB

Security provided by default

- **Encryption at rest** of all user data stored in tables, indexes, streams, and backups
- **Encryption in transit –** All communications to and from DynamoDB and other AWS resources use HTTPS

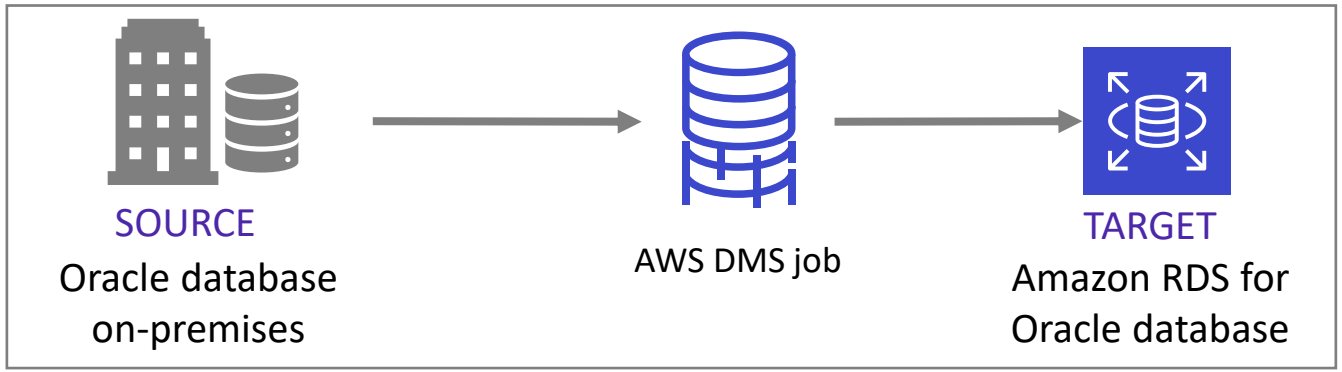
Migrating data into AWS databases

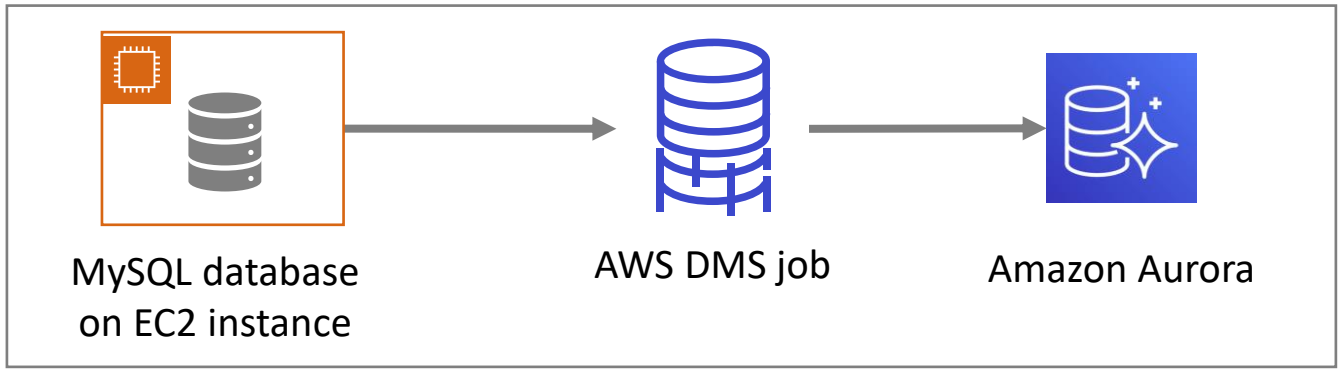
AWS Database Migration Service

AWS Database Migration Service (AWS DMS)

- Use to migrate to and from most commercial and open source databases
- Migrate between databases on Amazon EC2, Amazon RDS, Amazon S3, and on-premises

Example homogenous migration





Example heterogeneous migration

AWS DMS key features

- Perform one-time migrations
- Or, accomplish continuous data replication
  - Example: Configure continuous data replication of an on-premises database to an RDS instance
- AWS Schema Conversion Tool (AWS SCT) supports changing the database engine between source and target
- Typical migration major steps:
  1. Create a target database
  2. Migrate the database schema
  3. Set up the data replication process
  4. Initiate the data transfer, and confirm completion
  5. Switch production to the new database (for one-time migrations)

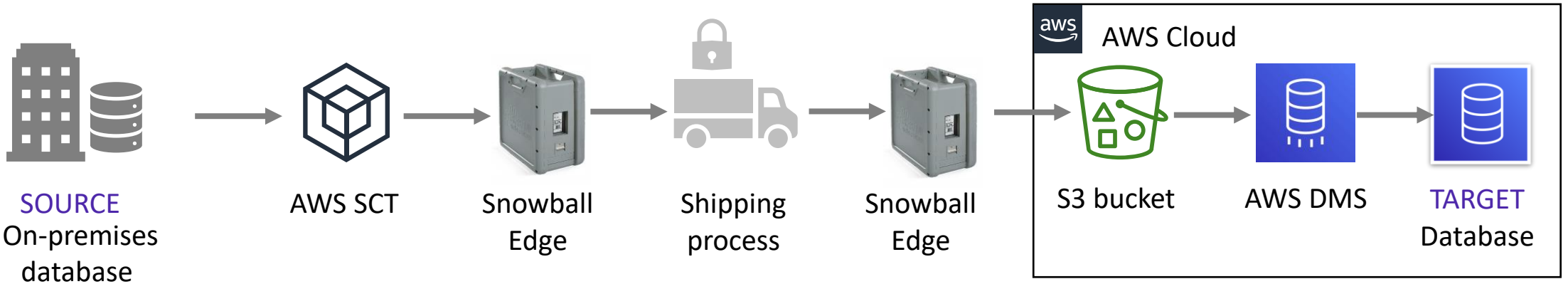
Using AWS Snowball Edge with AWS DMS

When migrating data is not practical:

- Database is too large
- Connection is too slow
- You have privacy and security concerns

Use **AWS Snowball Edge**

- Multi-terabyte transfer without using the internet



Module summary

In summary, in this module, you learned how to:

- Compare database types
- Differentiate between managed versus unmanaged services
- Explain when to use Amazon Relational Database Service (Amazon RDS)
- Explain when to use Amazon DynamoDB
- Describe available database security controls
- Describe how to migrate data into Amazon Web Services (AWS) databases
- Deploy a database server

Additional resources

- AWS Databases – Resource page
- Amazon RDS Getting Started Guide
- Best Practices for Amazon RDS
- Amazon RDS FAQs
- Amazon DynamoDB Developer Guide
- Amazon DynamoDB FAQs