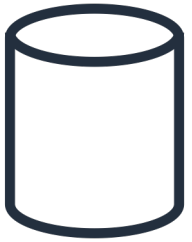
RDS

Databases Intro



- Storing data on disk (EFS, EBS, EC2 Instance Store, S3) can have its limits
- Sometimes, you want to store data in a database…
- You can structure the data
- You build indexes to efficiently query / search through the data
- You define relationships between your datasets

- Databases are optimized for a purpose and come with different features, shapes and constraints

Relational Databases

- Looks just like Excel spreadsheets, with links between them!
- Can use the SQL language to perform queries / lookups

Students

Student ID	Dept ID	Name	Email
1	M01	Joe Miller	joe@abc.com
2	B01	Sarah T	sarah@abc.com

Departments

Dept ID	SPOC	Email	Phone
M01	Kelly Jones	kelly@abc.com	+1234567890
B01	Satish Kumar	satish@abc.com	+1234567891

Subjects

Student ID	Subject
1	Physics
1	Chemistry
1	Math
2	History
2	Geography
2	Economics

NoSQL Databases

- NoSQL = non-SQL = non relational databases
- NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications.
- Benefits:
  - Flexibility: easy to evolve data model
  - Scalability: designed to scale-out by using distributed clusters
  - High-performance: optimized for a specific data model
  - Highly functional: types optimized for the data model

- Examples: Key-value, document, graph, in-memory, search databases

NoSQL data example: JSON

- JSON = JavaScript Object Notation
- JSON is a common form of data that fits into a NoSQL model
- Data can be nested
- Fields can change over time
- Support for new types: arrays, etc…

```
{
    "name":
    "John", "age":
    30,
    "cars": [
        "Ford",
        "BMW",
        "Fiat"
    ],
    "address": {
        "type": "house",
        "number": 23,
        "street": "Dream
        Road"
    }
}
```

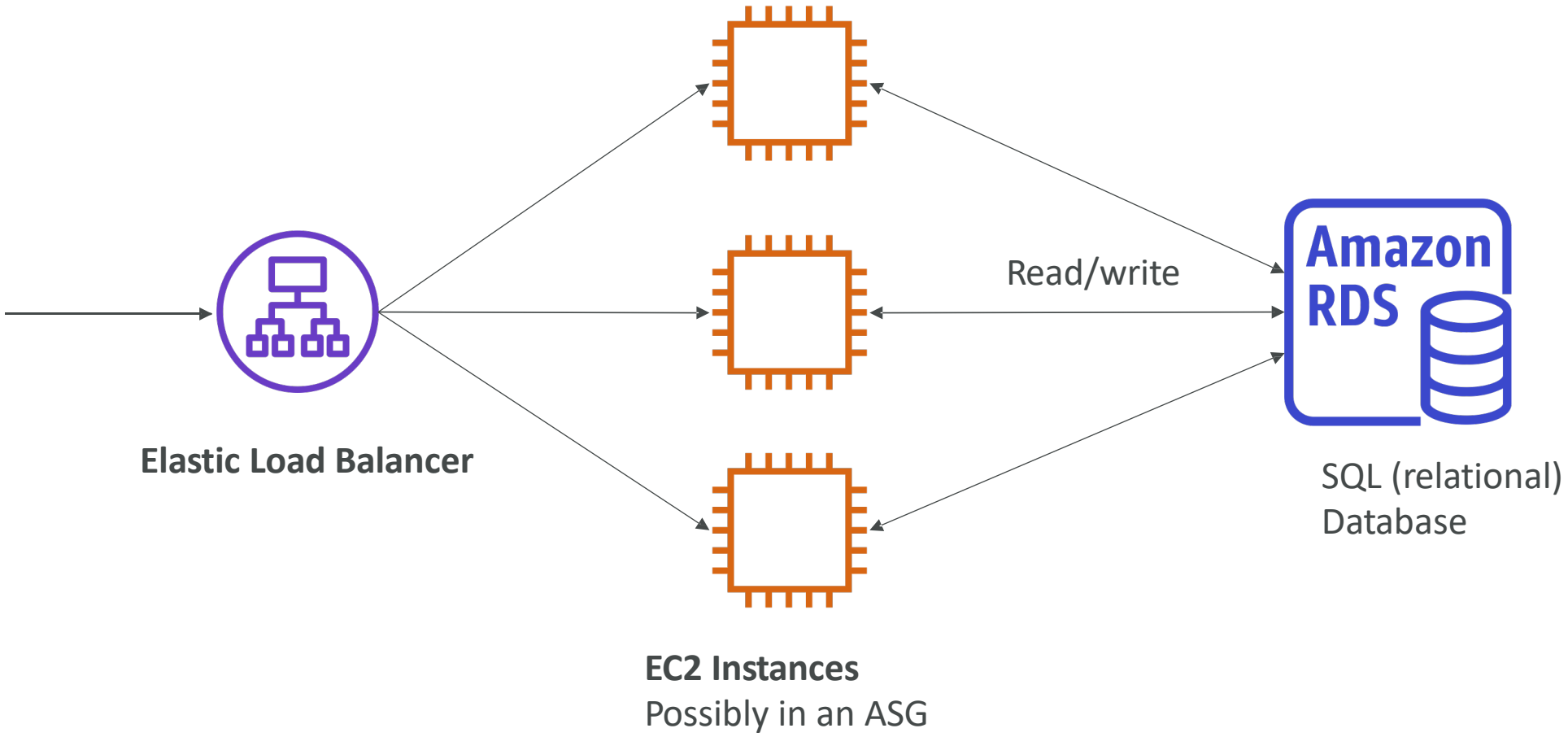
AWS RDS Overview

- RDS stands for Relational Database Service
- It’s a managed DB service for DB use SQL as a query language.
- It allows you to create databases in the cloud that are managed by AWS
  - Postgres
  - MySQL
  - MariaDB
  - Oracle
  - Microsoft SQL Server
  - Aurora (AWS Proprietary database)

Advantage over using RDS versus deploying DB on EC2

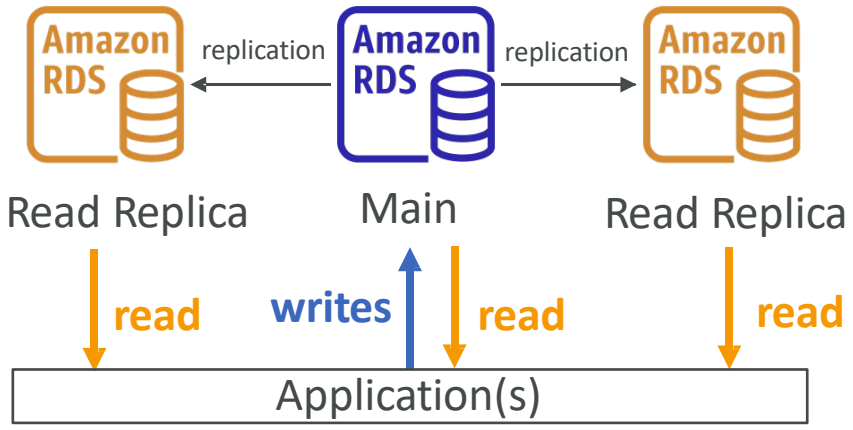
- RDS is a managed service:
  - Automated provisioning, OS patching
  - Continuous backups and restore to specific timestamp (Point in Time Restore)!
  - Monitoring dashboards
  - Read replicas for improved read performance
  - Multi AZ setup for DR (Disaster Recovery)
  - Maintenance windows for upgrades
  - Scaling capability (vertical and horizontal)
  - Storage backed by EBS (gp2 or io1)
- BUT you can’t SSH into your instances

RDS Solution Architecture

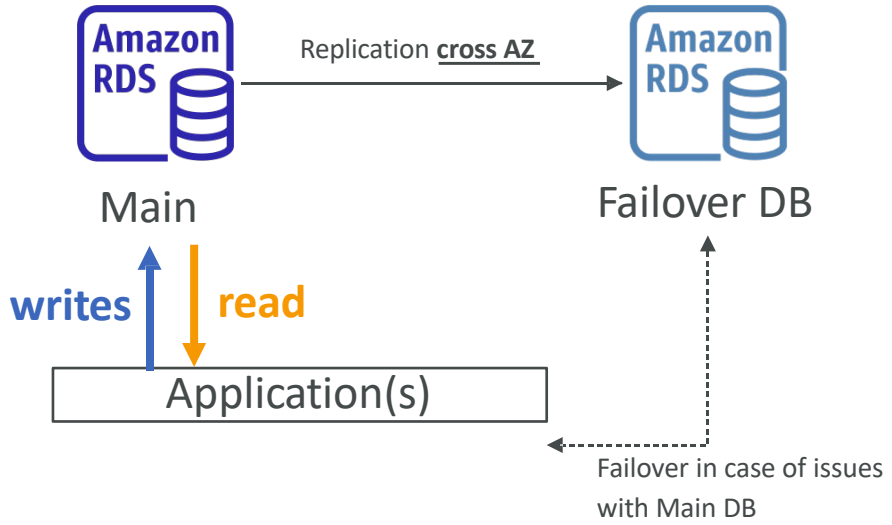


RDS Deployments: Read Replicas, Multi-AZ

- Read Replicas:
  - Scale the read workload of your DB
  - Can create up to 5 Read Replicas
  - Data is only written to the main DB

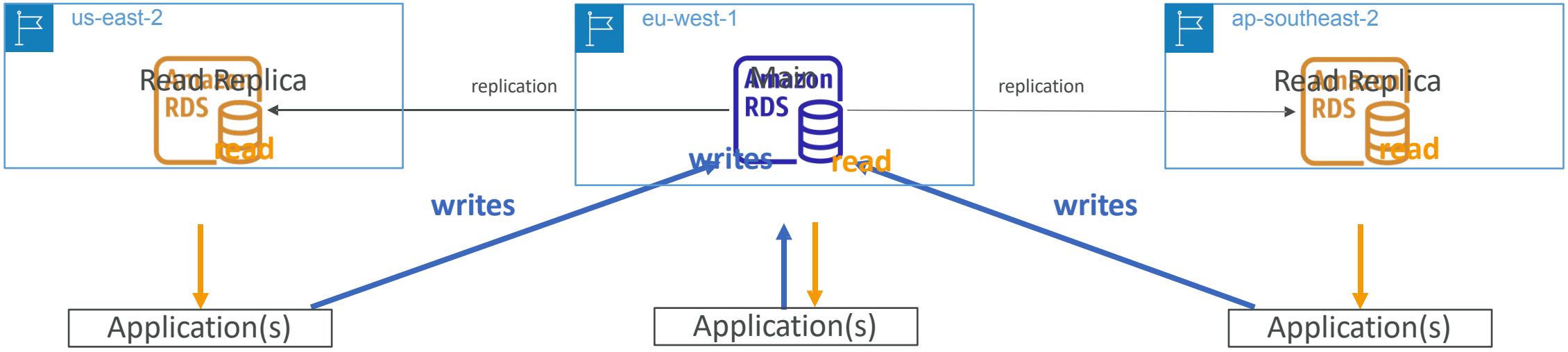


- Multi-AZ:
  - Failover in case of AZ outage (high availability)
  - Data is only read/written to the main database
  - Can only have 1 other AZ as failover



RDS Deployments: Multi-Region

- Multi-Region (Read Replicas)
  - Disaster recovery in case of region issue
  - Local performance for global reads
  - Replication cost



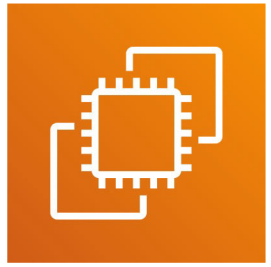
Lambda

What's serverless?

- Serverless is a new paradigm in which the developers don't have to manage servers anymore…
- They just deploy code
- They just deploy… functions !
- Initially… Serverless == FaaS (Function as a Service)
- Serverless was pioneered by AWS Lambda but now also includes anything that's managed:"databases, messaging, storage, etc."
- Serverless does not mean there are no servers…

  it means you just don't manage / provision / see them

Why AWS Lambda



Amazon EC2

- Virtual Servers in the Cloud
- Limited by RAM and CPU
- Continuously running
- Scaling means intervention to add / remove servers

Amazon Lambda

- Virtual functions – no servers to manage!
- Limited by time - short executions
- Run on-demand
- Scaling is automated!

Benefits of AWS Lambda

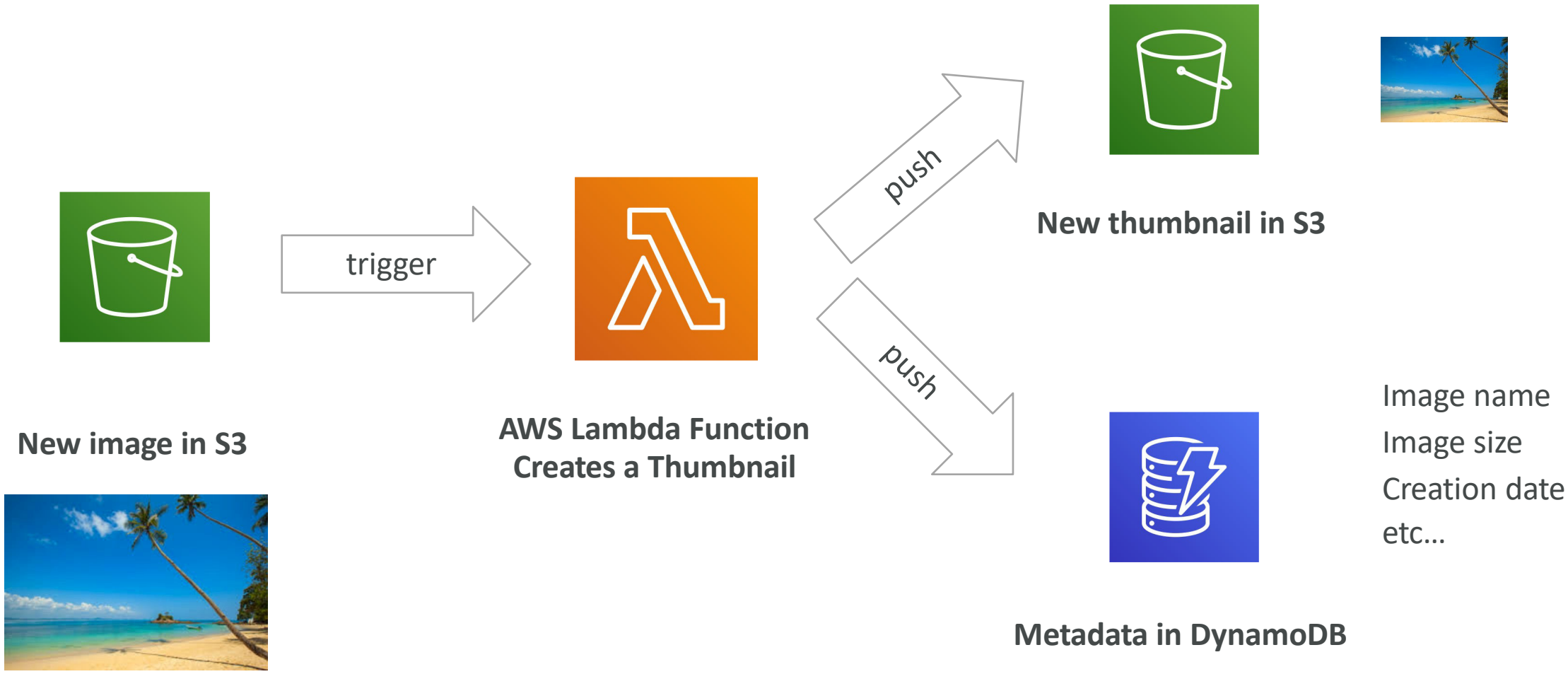
- Easy Pricing:
  - Pay per request and compute time
  - Free tier of 1,000,000 AWS Lambda requests and 400,000 GBs of compute time
- Integrated with the whole AWS suite of services
- Event-Driven: functions get invoked by AWS when needed
- Integrated with many programming languages
- Easy monitoring through AWS CloudWatch
- Easy to get more resources per functions (up to 10GB of RAM!)
- Increasing RAM will also improve CPU and network!

AWS Lambda language support

- Node.js (JavaScript)
- Python
- Java (Java 8 compatible)
- C# (.NET Core)
- Golang
- C# / Powershell
- Ruby
- Custom Runtime API (community supported, example Rust)

- Lambda Container Image
  - The container image must implement the Lambda Runtime API
  - ECS / Fargate is preferred for running arbitrary Docker images

Example: Serverless Thumbnail creation



Example: Serverless CRON Job

AWS Lambda Pricing: example

- You can find overall pricing information here: https://aws.amazon.com/lambda/pricing/
- Pay per calls:
  - First 1,000,000 requests are free
  - $0.20 per 1 million requests thereafter ($0.0000002 per request)
- Pay per duration: (in increment of 1 ms)
  - 400,000 GB-seconds of compute time per month for FREE
  - == 400,000 seconds if function is 1GB RAM
  - == 3,200,000 seconds if function is 128 MB RAM
  - After that $1.00 for 600,000 GB-seconds
- It is usually ver y cheap to run AWS Lambda so it's ver y popular