DAY-III

An Introduction to

By P.S.R.Patnaik

Microsoft Specialist: Architecting Microsoft Azure Solutions
Microsoft, License F312-3640,
AWS Certified Solutions Architect - Associate
Amazon Web Services, License 639BWPV1JFRE11CR

AWS LAMBDA

AWS LAMBDA

Introduction to Lambda

Ref. Link

AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.

AWS LAMBDA

Important Components of Lambda

Ref. Link

AWS Lambda can automatically run code in response to multiple events, such as HTTP requests via Amazon API Gateway, modifications to objects in Amazon S3 buckets, table updates in Amazon DynamoDB, and state transitions in AWS Step Functions.

AWS LAMBDA

Creating simple function in Lambda

Ref. Link

Sign in to the AWS Management Console and open the AWS Lambda console.

Choose Create a function under the Get Started section to proceed.

On the Create function page, chose a function

Leave the Policy templates field blank.

Under your new function-name page, note function name

AWS LAMBDA

Creating and Managing triggers in Lambda

Ref. Link

A Lambda@Edge trigger is one combination of CloudFront distribution, cache behavior, and event that causes a function to execute. You can specify one or more CloudFront triggers that cause the function to run. For example, you can create a trigger that causes the function to execute when CloudFront receives a request from a viewer for a specific cache behavior you set up for your distribution.

AWS LAMBDA

Sample Code run using Lambda

Ref. Link

- On the yourfunction page, choose Test.
- In the Configure test event page, choose Create new test event
- Choose Create and then choose Test.
- AWS Lambda executes your function on your behalf.
- Upon successful execution, view results in the console.

AWS LAMBDA
Limitations & Best Practice of Lambda

Ref. Link

The following are recommended best practices for using AWS Lambda:

- Function Code
- Function Configuration
- Alarming and Metrics
- Stream Event Invokes
- Async Invokes
- Lambda VPC

AWS RDS

AWS RDS

Introduction to Database - Its Components

Ref. Link

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while automating time-consuming administration tasks such as hardware provisioning, database setup, patching and backups. It frees you to focus on your applications so you can give them the fast performance, high availability, security and compatibility they need.

AWS RDS

Database Services provided by AWS

Ref. Link

Amazon RDS is available on several database instance types - optimized for memory, performance or I/O - and provides you with six familiar database engines to choose from, including Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server. You can use the AWS Database Migration Service to easily migrate or replicate your existing databases to Amazon RDS.

AWS RDS
Introduction to RDS

Ref. Link

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

AWS RDS
Components of RDS

Ref. Link

- Remote Desktop Connection Broker
- Remote Desktop Gateway
- Remote Desktop Licensing
- Remote Desktop Session Host
- Remote Desktop Virtualization Host
- Remote Desktop Web Access

AWS RDS
DB engines provided by RDS

Ref. Link

- MariaDB on Amazon RDS
- Microsoft SQL Server on Amazon RDS
- MySQL on Amazon RDS
- Oracle on Amazon RDS
- PostgreSQL on Amazon RDS

AWS RDS

Snapshots and Back-up in RDS

Ref. Link

Amazon RDS creates and saves automated backups of your DB instance. Amazon RDS creates a storage volume snapshot of your DB instance, backing up the entire DB instance and not just individual databases.

Amazon RDS creates automated backups of your DB instance during the backup window of your DB instance. Amazon RDS saves the automated backups of your DB instance according to the backup retention period that you specify.and you can recover your database to any point in time during the backup retention period.

AWS RDS
Read Replicas in RDS

Ref. Link

Amazon RDS Read Replicas provide enhanced performance and durability for database (DB) instances. This feature makes it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. You can create one or more replicas of a given source DB Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput. Read replicas can also be promoted when needed to become standalone DB instances.

AWS RDS

Creating and connecting to a RDS database

Ref. Link

Step 1: Create a MySQL DB Instance

Step 2: Download a SQL Client

Step 3: Connect to the MySQL Database

Step 4: Delete the DB Instance

AWS RDS
RDS Security Groups

Ref. Link

Security groups control the access that traffic has in and out of a DB instance. Three types of security groups are used with Amazon RDS: DB security groups, VPC security groups, and Amazon EC2 security groups. In simple terms, these work as follows:

- A DB security group controls access to EC2-Classic DB instances that are not in a VPC.
- A VPC security group controls access to DB instances and EC2 instances inside a VPC.
- An EC2 security group controls access to an EC2 instance.

AWS RDS
Pricing in RDS

Ref. Link

Amazon RDS is free to try. Pay only for what you use. There is no minimum fee. You can pay for Amazon RDS using On-Demand or Reserved Instances. Estimate your monthly bill using the AWS Simple Monthly Calculator.

Amazon RDS provides a selection of instance types optimized to fit different relational database use cases. Select one of the Amazon RDS database engines below to view pricing.

AWS RDS

Limitations and Best Practice - RDS

Ref. Link

- Amazon RDS Basic Operational Guidelines
- DB Instance RAM Recommendations
- Amazon RDS Security Best Practices
- Using Enhanced Monitoring to Identify Operating System Issues
- Using Metrics to Identify Performance Issues
- Best Practices for Working with MySQL Storage Engines
- Best Practices for Working with MariaDB Storage Engines
- Best Practices for Working with Oracle
- Best Practices for Working with PostgreSQL
- Best Practices for Working with SQL Server
- Working with DB Parameter Groups

AWS DynamoDB

AWS DynamoDB
Introduction to NoSQL Database

Ref. Link

NoSQL databases are purpose built for specific data models and have flexible schemas for building modern applications. NoSQL databases are widely recognized for their ease of development, functionality, and performance at scale. They use a variety of data models, including document, graph, key-value, in-memory, and search.

AWS DynamoDB
Difference between SQL and NoSQL

Ref. Link

	Relational databases	NoSQL databases
Optimal workloads	Relational databases are designed for transactional and strongly consistent online transaction processing (OLTP) applications and are good for online analytical processing (OLAP).	NoSQL key-value, document, graph, and in-memory databases are designed for OLTP for a number of data access patterns that include low-latency applications. NoSQL search databases are designed for analytics over semi-structured data.
Data model	The relational model normalizes data into tables that are composed of rows and columns. A schema strictly defines the tables, rows, columns, indexes, relationships between tables, and other database elements. The database enforces the referential integrity in relationships between tables.	NoSQL databases provide a variety of data models that includes document, graph, key-value, in-memory, and search.

AWS DynamoDB
Difference between SQL and NoSQL

Ref. Link

	Relational databases	NoSQL databases
ACID properties	Relational databases provide atomicity, consistency, isolation, and durability (ACID) properties: • Atomicity requires a transaction to execute completely or not at all. • Consistency requires that when a transaction has been committed, the data must conform to the database schema. • Isolation requires that concurrent transactions execute separately from each other. Durability requires the ability to recover from an unexpected system failure or power outage to the last known state.	NoSQL databases often make tradeoffs by relaxing some of the ACID properties of relational databases for a more flexible data model that can scale horizontally. This makes NoSQL databases an excellent choice for high throughput, low-latency use cases that need to scale horizontally beyond the limitations of a single instance.

AWS DynamoDB
Difference between SQL and NoSQL

Ref. Link

	Relational databases	NoSQL databases
Performance	Performance is generally dependent on the disk subsystem. The optimization of queries, indexes, and table structure is often required to achieve peak performance.	Performance is generally a function of the underlying hardware cluster size, network latency, and the calling application.
Scale	Relational databases typically scale up by increasing the compute capabilities of the hardware or scale-out by adding replicas for read-only workloads.	NoSQL databases typically are partitionable because key-value access patterns are able to scale out by using distributed architecture to increase throughput that provides consistent performance at near boundless scale.

AWS DynamoDB
Components of DynamoDB

Ref. Link

In DynamoDB, tables, items, and attributes are the core components that you work with. A table is a collection of items, and each item is a collection of attributes.

- Tables, Items, and Attributes
- Primary Key
- Secondary Indexes
- DynamoDB Streams

AWS DynamoDB
AutoScaling in DynamoDB

Ref. Link

Even if you're not around, DynamoDB Auto Scaling will be monitoring your tables and indexes to automatically adjust throughput in response to changes in application traffic. This can make it easier to administer your DynamoDB data, help you maximize availability for your applications, and help you reduce your DynamoDB costs.

AWS DynamoDB
DynamoDB Streams

Ref. Link

DynamoDB Streams provides a time ordered sequence of item level changes in any DynamoDB table. The changes are de-duplicated and stored for 24 hours. This capability enables you to extend the power of DynamoDB with cross-region replication, continuous analytics with Redshift integration, change notifications, and many other scenarios.

AWS DynamoDB
Indexing in DynamoDB

Ref. Link

Indexes give you access to alternate query patterns, and can speed up queries. This section compares and contrasts index creation and usage in SQL and DynamoDB.

Whether you are using a relational database or DynamoDB, you should be judicious with index creation. Whenever a write occurs on a table, all of the table's indexes must be updated. In a write-heavy environment with large tables, this can consume large amounts of system resources. In a read-only or read-mostly environment, this is not as much of a concern.

AWS DynamoDB
Data Distribution in DynamoDB

Ref. Link

DynamoDB stores data in partitions. A partition is an allocation of storage for a table, backed by solid-state drives (SSDs) and automatically replicated across multiple Availability Zones within an AWS Region. Partition management is handled entirely by DynamoDB, you never have to manage partitions yourself.

When you create a table, the initial status of the table is CREATING. During this phase, DynamoDB allocates sufficient partitions to the table so that it can handle your provisioned throughput requirements.

AWS DynamoDB
Backup and Monitoring in DynamoDB

Ref. Link

When you create an on-demand backup, a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes.

All backups in DynamoDB work without consuming any provisioned throughput on the table. DynamoDB backups do not guarantee causal consistency across items; however, the skew between updates in a backup is usually much less than a second.

AWS DynamoDB
Creating Table and loading data into DynamoDB

Ref. Link

Step 1: Create Example Tables

Step 2: Load Data into Tables

Step 3: Query the Data

Step 4: (Optional) Clean up

AWS DynamoDB
Pricing in DynamoDB

Ref. Link

Pay only for the resources DynamoDB provisions to achieve your target read and write capacity. Afterward, DynamoDB will auto scale your capacity based on usage. Optionally, you can directly specify read and write capacity if you prefer to manually manage table throughput. Estimate your monthly bill using the AWS Simple Monthly Calculator.

AWS DynamoDB
Best Practices - DynamoDB

Ref. Link

- NoSQL Design for DynamoDB
  - Differences Between Relational Data Design and NoSQL
  - Two Key Concepts for NoSQL Design
  - Approaching NoSQL Design
- Best Practices for Designing and Using Partition Keys Effectively
  - Using Burst Capacity Effectively
  - Understanding DynamoDB Adaptive Capacity
  - Designing Partition Keys to Distribute Your Workload Evenly
  - Using Write Sharding to Distribute Workloads EvenlyDistributing Write Activity Efficiently During Data Upload
- Best Practices for Using Sort Keys to Organize Data
  - Using Sort Keys for Version Control

AWS EB

AWS EB
Introduction to Elastic Beanstalk

Ref. Link

A

AWS EB
Important Concepts in EB

Ref. Link

A

AWS EB
Managing Environments in EB

Ref. Link

A

AWS EB
Managing Applications in EB

Ref. Link

A

AWS EB
EB Platforms

Ref. Link

A

AWS EB

Getting started with EB

Ref. Link

A

AWS EB
Pricing in EB

Ref. Link

A

AWS API Gateway

AWS API Gateway

Gateway Introduction to API Gateway

Ref. Link

A

AWS API Gateway
How API Gateway Works

Ref. Link

A

AWS API Gateway

Why Should we use API Gateway

Ref. Link

A

AWS API Gateway
Pricing in API Gateway

Ref. Link

A

AWS API Gateway
Use Case Discussion

Ref. Link

A

DAY-II
THE END

Ref. Link

END