Getting Started with MicroService on AWS

Sanchit Jain
9th April, Saturday
7:00 PM to 8:00 PM

Speakers

Sanchit Jain

Lead Architect - AWS at Quantiphi

AWS APN Ambassador

FOLLOW ME

Overview of AWS Lambda

What is serverless?

What is Serverless?

a cloud-native platform

for

short-running, stateless computation

and

event-driven applications

which

scales up and down instantly and automatically

and

charges for actual usage at a millisecond granularity

Greater Agility

Less Overhead

Better Focus

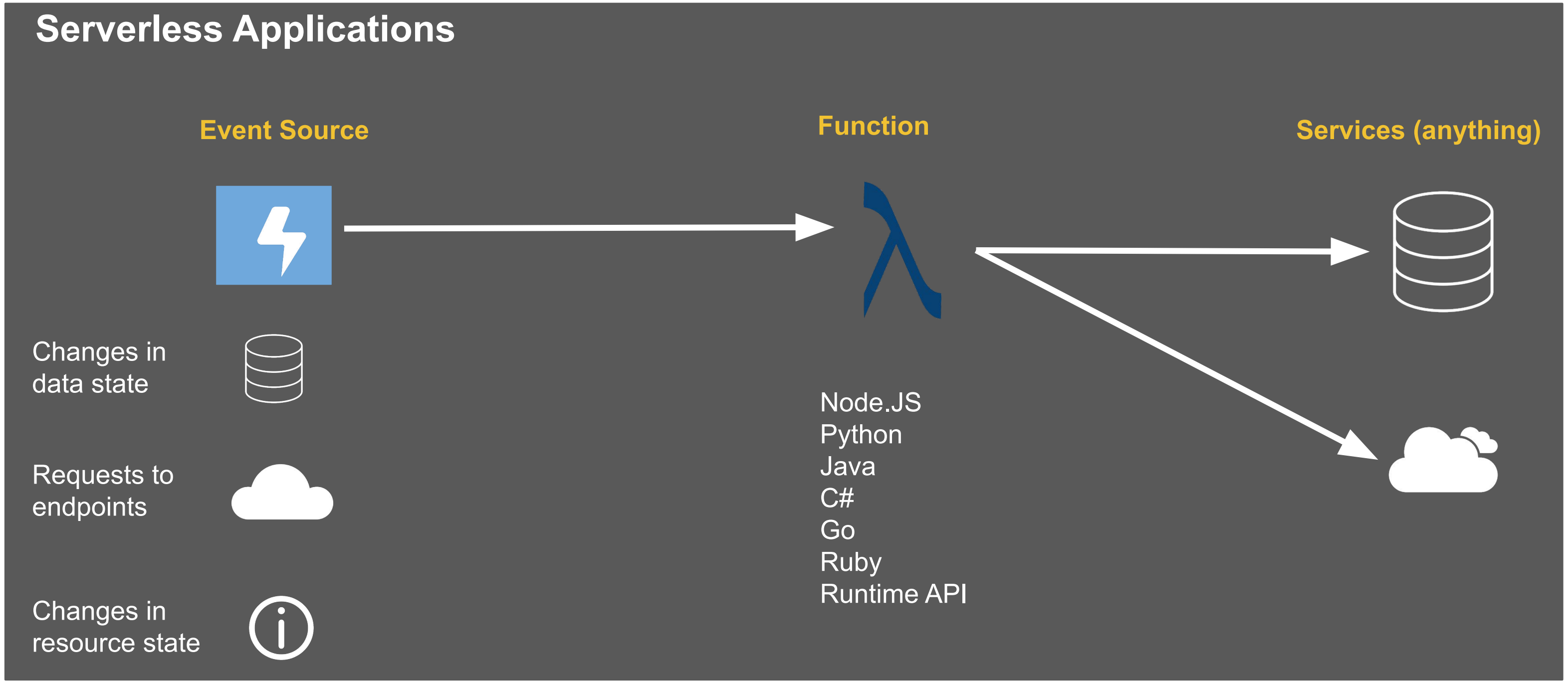
Increased Scale

More Flexibility

Faster Time To Market

What triggers code execution?

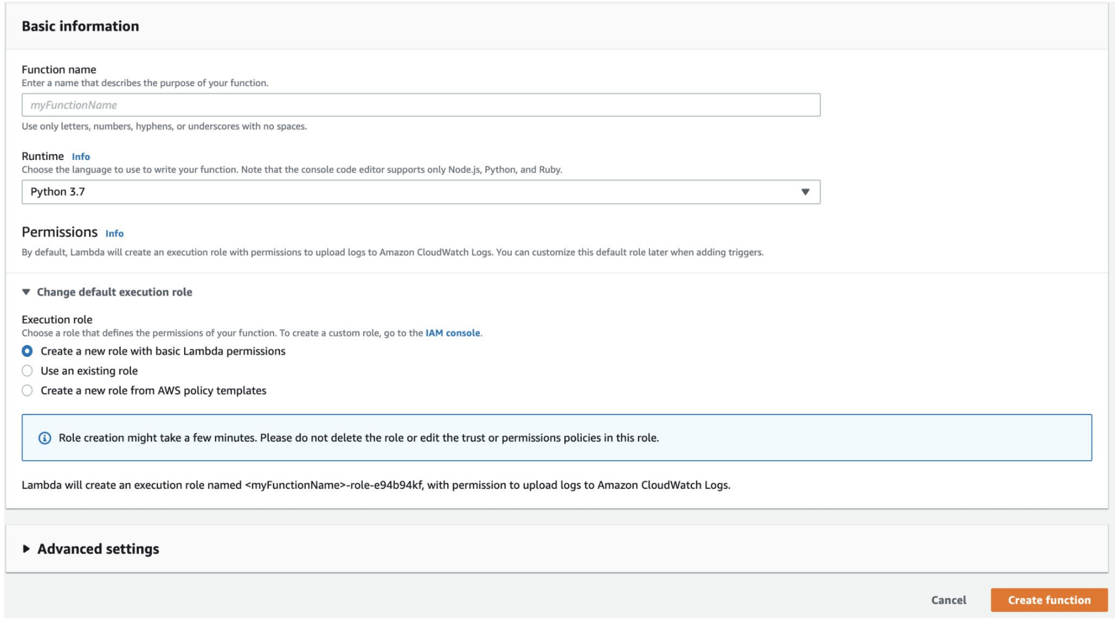
- Runs code in response to events
- Event-programming model



DEMO

Create function

- Navigate to the AWS Lambda service
- On the Lambda console, Click Create function
- In the Create function, enter demo-lambda as the function name, runtime as Python 3.7
- In the execution role section, select the role cloudformation-iam-policy--xxxxx from the drop-down list
- Click Create function



Basic information

Function name
Enter a name that describes the purpose of your function.
myFunctionName
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.7

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ **Change default execution role**

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

- [✓ Create a new role with basic Lambda permissions]
- Use an existing role
- Create a new role from AWS policy templates

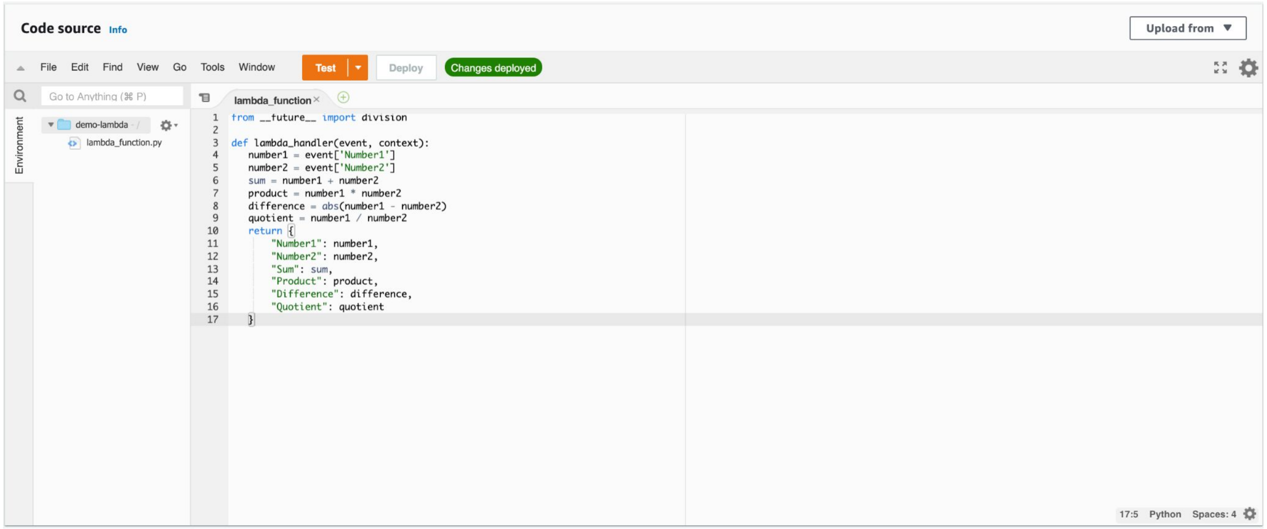
ⓘ Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named <myFunctionName>-role-e94b94kf, with permission to upload logs to Amazon CloudWatch Logs.

▶ **Advanced settings**

Configure function

- Download the code from this link and paste it into code source
- Click Deploy, then click Test
- Copy the message { "Number1": 10, "Number2": 20 }, and paste it
- Click Test again, and capture the output



```
from __future__ import division

def lambda_handler(event, context):
    number1 = event['Number1']
    number2 = event['Number2']
    sum = number1 + number2
    product = number1 * number2
    difference = abs(number1 - number2)
    quotient = number1 / number2
    return {
        "Number1": number1,
        "Number2": number2,
        "Sum": sum,
        "Product": product,
        "Difference": difference,
        "Quotient": quotient
    }
```

Lambda Code Walkthrough

Anatomy Of A Lambda Function

Handler() Function

Function to be executed upon invocation

Event Object

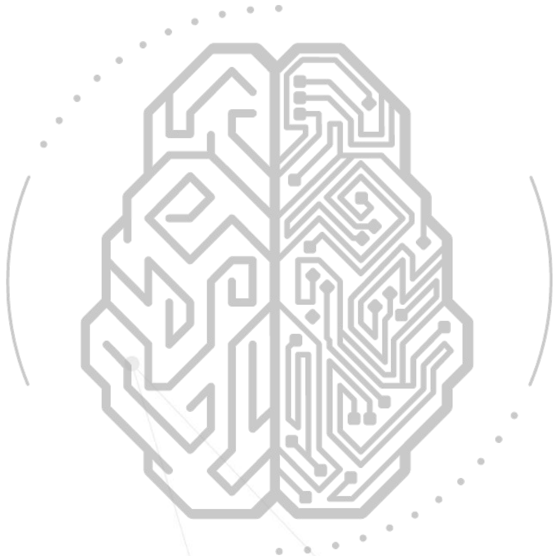
Data sent during Lambda Function invocation

Context Object

Methods available to interact with runtime information (request ID, log group, more)

```
def lambda_handler(event, context):
    return {
        “Hello World!”
 }
```

Overview of AWS API Gateway

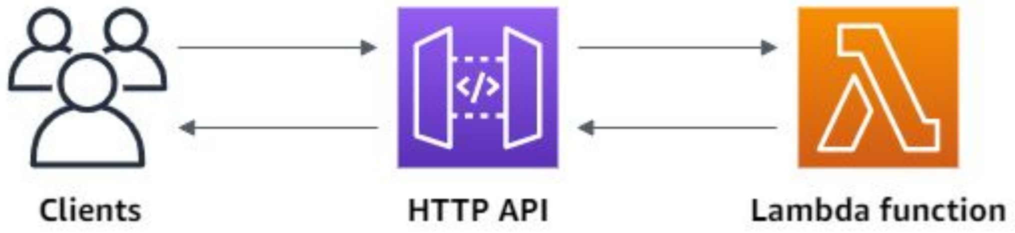


What is AWS API Gateway?

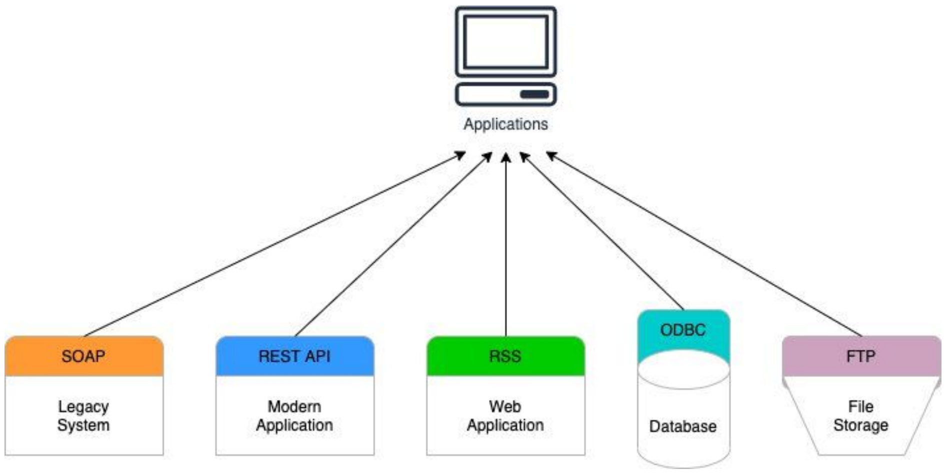
- Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale.

- API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud

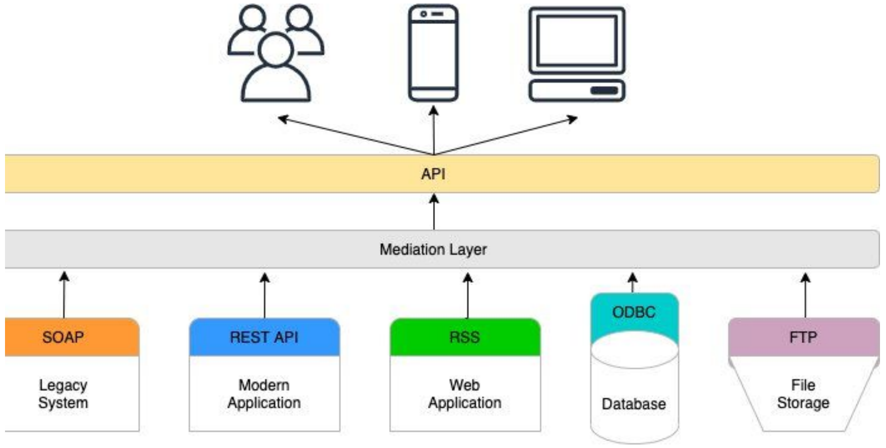
- API Gateway creates RESTful APIs that:
  - Are HTTP-based.
  - Enable stateless client-server communication.
  - Implement standard HTTP methods such as GET, POST, PUT, PATCH, and DELETE.



Overview



Traditional way

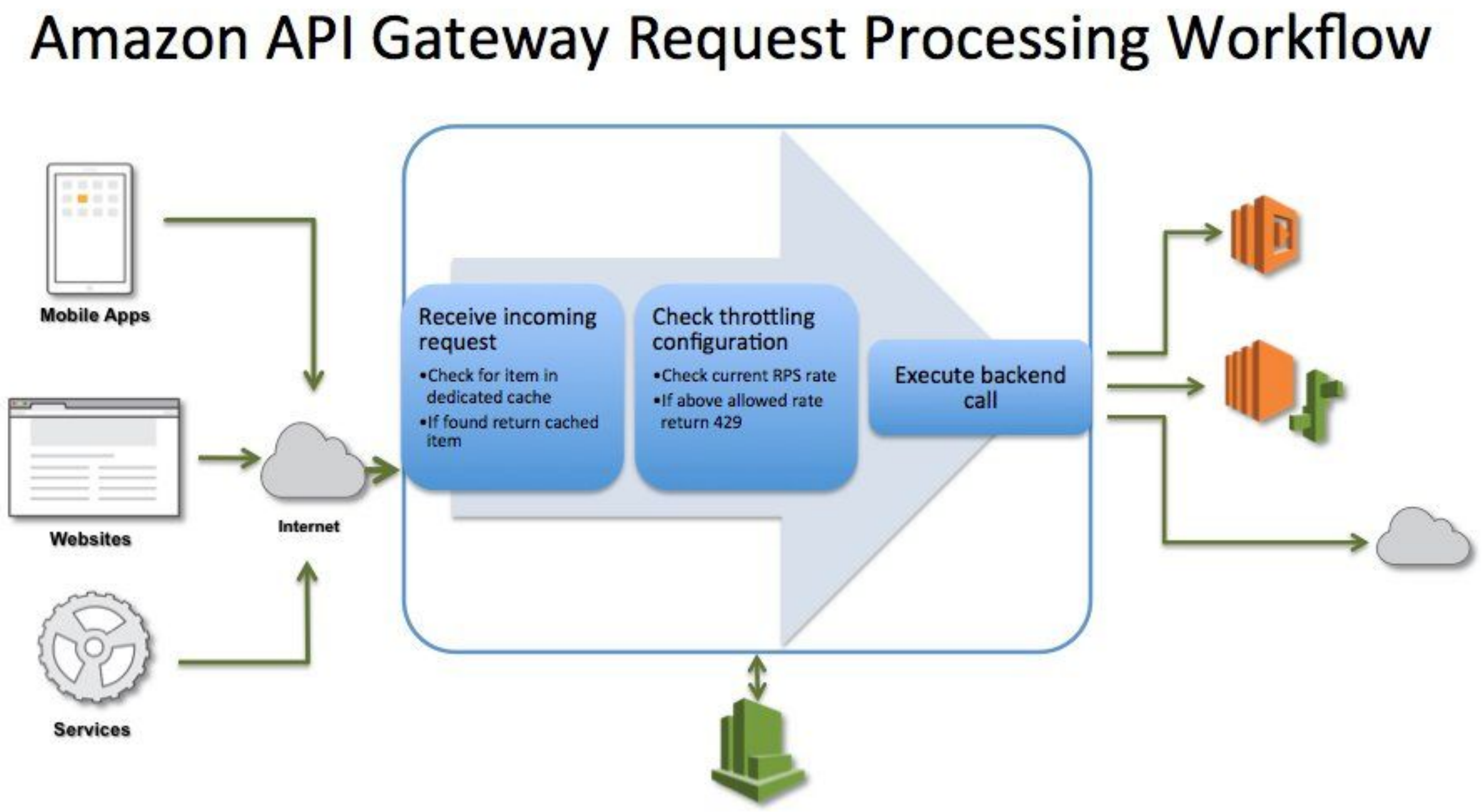


REST Based System

Security Features

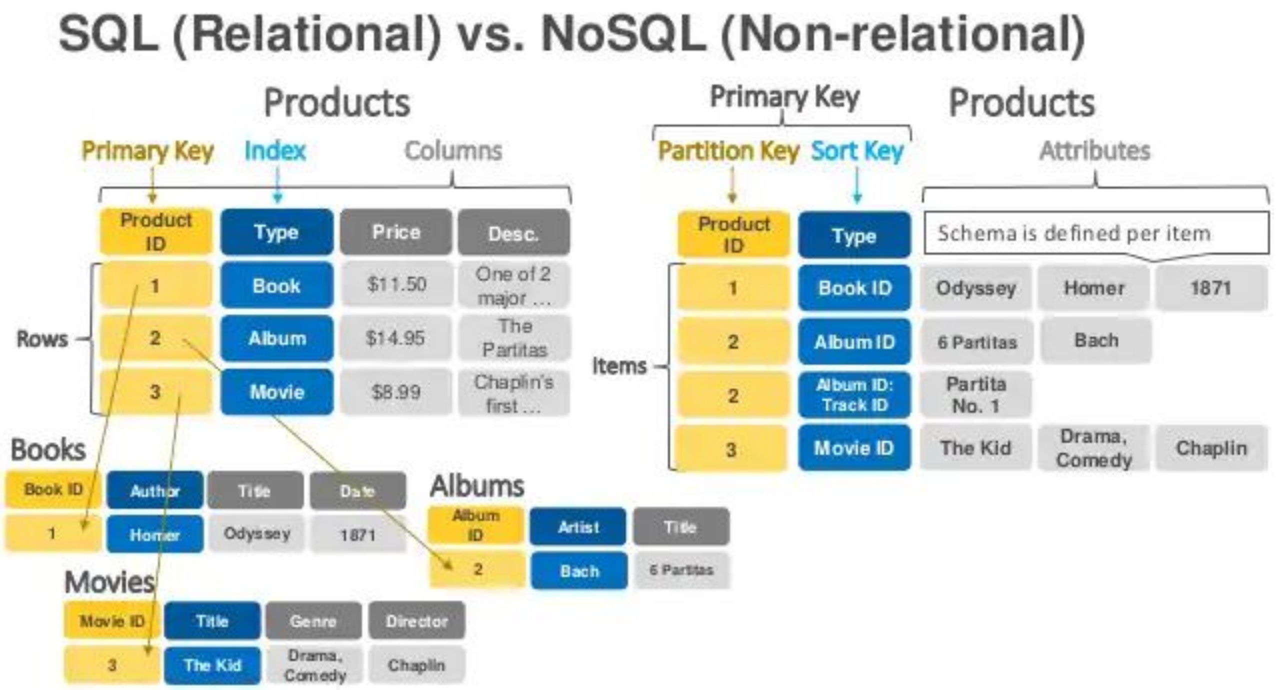
1. Authentication and Authorization
    - Resource policies
    - Standard AWS IAM roles and policies
    - IAM tags
    - Endpoint policies for interface VPC endpoints
    - Lambda authorizers
    - Amazon Cognito user pools
2. Access control
    - Cross-origin resource sharing (CORS)
    - Client-side SSL certificates
    - AWS WAF
3. Tracking and Limiting usage
    - Usage plans
    - API keys

Design Pattern


Amazon API Gateway Request Processing Workflow
Mobile Apps
Websites
Services
Internet
Receive incoming request
•Check for item in dedicated cache
•If found return cached item
Check throttling configuration
•Check current RPS rate
•If above allowed rate return 429
Execute backend call

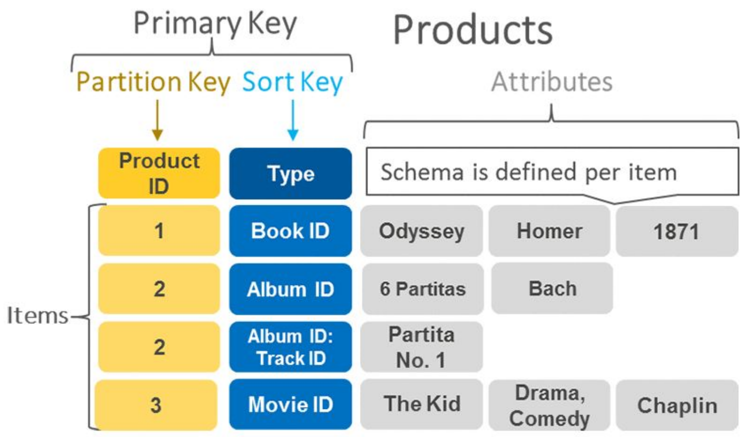
Overview of AWS DynamoDB

SQL (Relational) vs. NoSQL (Non-Relational)



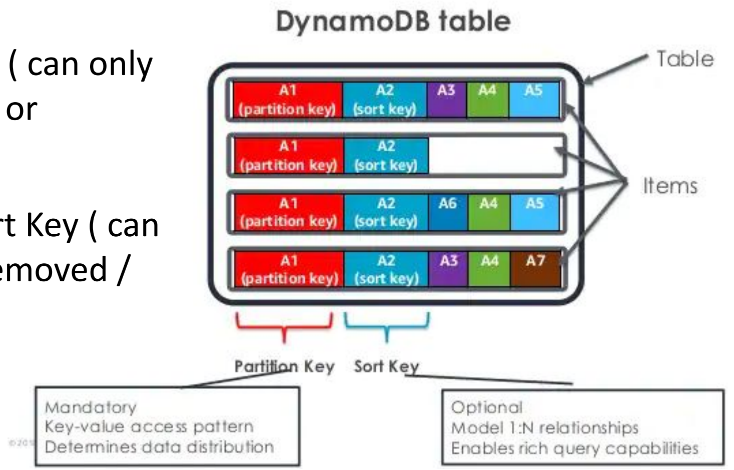
What is DynamoDB?

- Fully managed, fast NoSQL key-value database service
- No hardware provisioning, setup and configuration, replication, software patching, or cluster scaling required
- Allows to delete expired items from tables automatically to help reduce storage usage using TTL



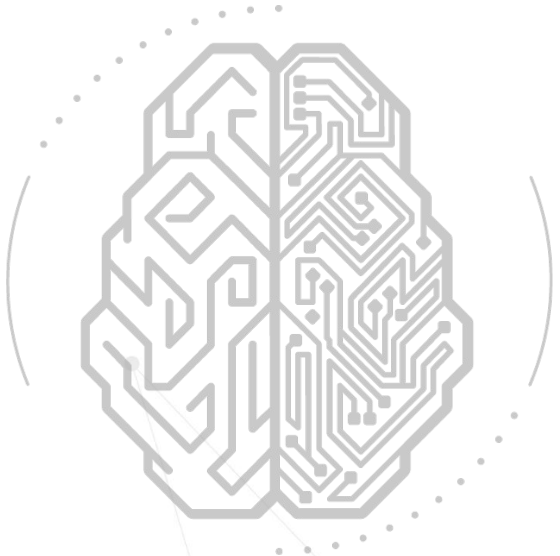
Overview

- DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second
- Primary Key - Two types of primary key
  - Partition Key (Hash Key) will help determine the physical location of data.
  - Composite key: Partition Key (Hash Key) & Sort Key (Range key – e.g date)
- Secondary Indexes - Secondary indexes allow you to perform queries on attributes that are not part of the table’s primary key
- Local Secondary Index - Same Partition Key + Different Sort Key (can only be created while creating the table, cannot be added/removed or modified later)
- Global Secondary Index - Different Partition Key + Different Sort Key (can be created during the table creation or can be added later or removed / modified later)



DEMO

Demo - CloudNative Web Application



THANK YOU