MENTORMATE™
DevCamp

# Serverless

**Yulia Tenincheva**

Senior Cloud Engineer, MentorMate

## Serverless Architecture

MENTORMATE™
DevCamp

How should my app withstand a server **failing**?

How can I tell if a server has been **compromised**?

How can I increase **utilization** of my servers?

Which **OS** should my servers run?

How much remaining **capacity** do my servers have?

How should I implement dynamic **configuration changes** on my servers

When should I decide to **scale up** my servers?

**What size** servers are right for my budget?

How will I keep my server OS **patched**?

How can I control **access from** my servers?

Which packages should be baked into my **server images**?

# Servers

(AAHHHHHHHHH!!)

How will new code be **deployed** to my servers?

How will the application handle server **hardware failure**?

How many users create **too much load** for my servers?

What size server is right for my **performance**?

Which users should have **access to** my servers?

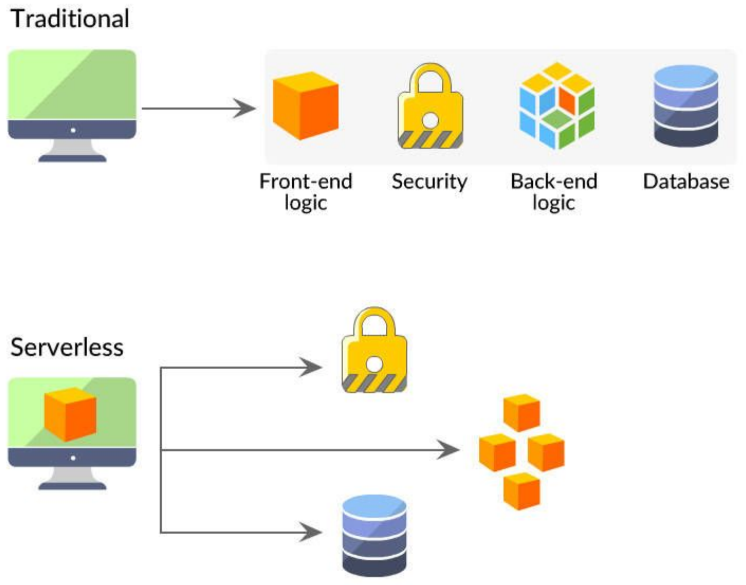
Should I **tune OS settings** to optimize my application?

When should I decide to **scale out** my servers?

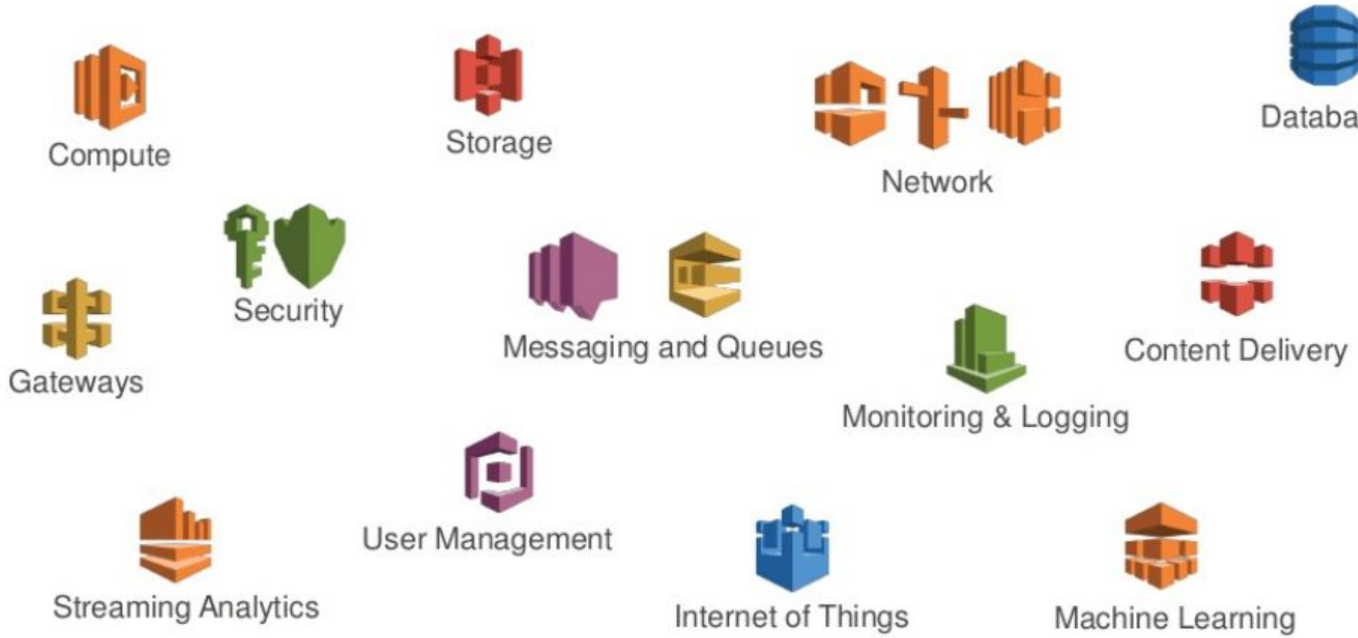
**How many** servers should I budget for?

# Architect to be Serverless

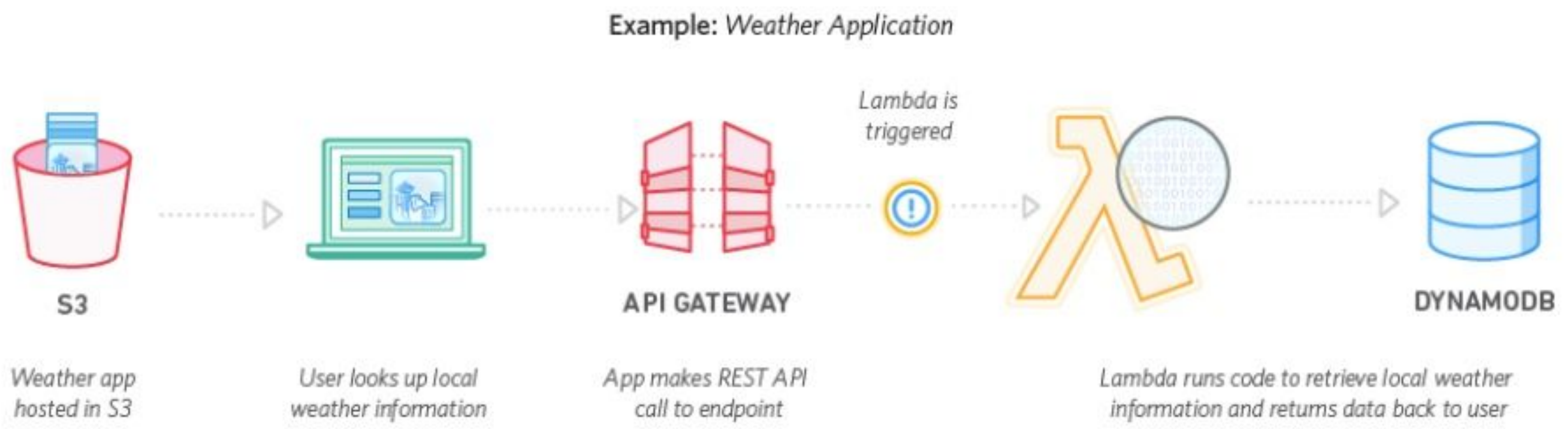
- Fully Managed
  - No provisioning
  - Zero administration
  - High availability
- Developer Productivity
  - Focus on the code that matters
  - Innovate rapidly
  - Reduce time to market
- Continuous Scaling
  - Automatically
  - Scale up and scale down



# Many Serverless Options on AWS



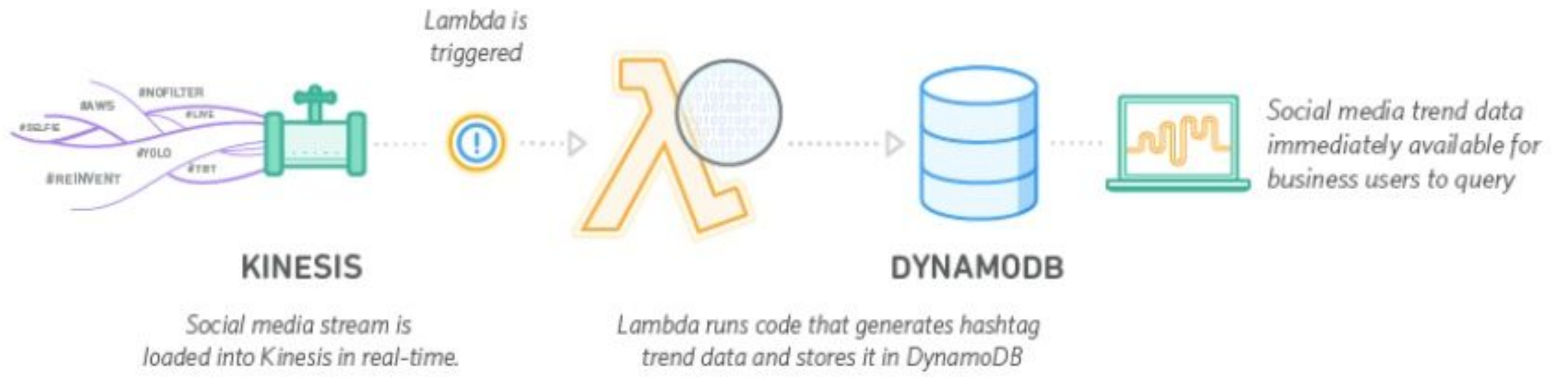
# Example of Serverless Architecture


Example: Weather Application
S3
Weather app hosted in S3
User looks up local weather information
API GATEWAY
App makes REST API call to endpoint
Lambda is triggered
DYNAMODB
Lambda runs code to retrieve local weather information and returns data back to user

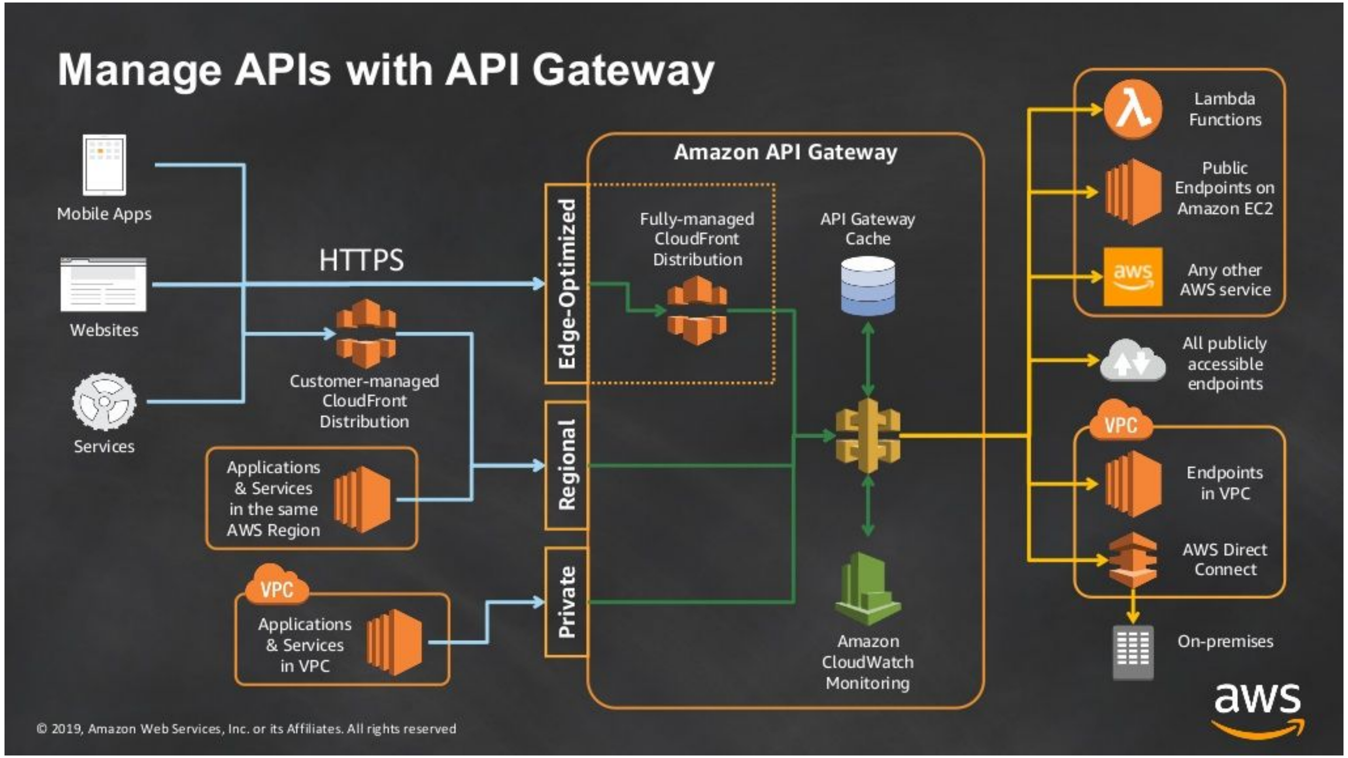
MENTORMATE™
DevCamp

# Example of Serverless Architecture

**Example:** *Analysis of Streaming Social Media Data*


Lambda is triggered
#NOFILTER
#AWS
#SELFIE
#LIVE
#YOLO
#REINVENT
#TBT
KINESIS
Social media stream is loaded into Kinesis in real-time.
Lambda runs code that generates hashtag trend data and stores it in DynamoDB
DYNAMODB
Social media trend data immediately available for business users to query

# Example of Serverless Architecture


Manage APIs with API Gateway
Mobile Apps
Websites
Services
HTTPS
Customer-managed CloudFront Distribution
Applications & Services in the same AWS Region
VPC
Applications & Services in VPC
Edge-Optimized
Regional
Private
Amazon API Gateway
Fully-managed CloudFront Distribution
API Gateway Cache
Amazon CloudWatch Monitoring
Lambda Functions
Public Endpoints on Amazon EC2
aws
Any other AWS service
All publicly accessible endpoints
VPC
Endpoints in VPC
AWS Direct Connect
On-premises
aws

MENTORMATE™ DevCamp

## Serverless Building Blocks

MENTORMATE™
DevCamp

**Database**

Amazon DynamoDB

**Gateways**

Amazon API Gateway

**Security**

AWS IAM

AWS KMS

**Messaging and Queues**

Amazon SQS

Amazon SNS

**Compute**

AWS Lambda

**Storage**

Amazon S3

**Network**

Amazon VPC

Amazon Route 53

Elastic Load Balancing

**Content Delivery**

Amazon CloudFront

**Streaming Analytics**

Amazon Kinesis

**User Management**

Amazon Cognito

**Internet of Things**

AWS IoT

**Monitoring & Logging**

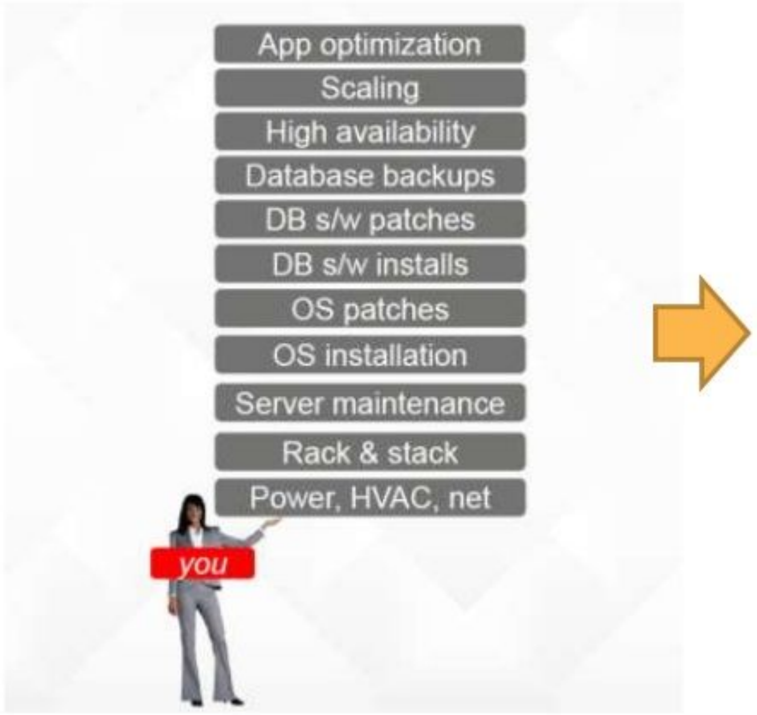
Amazon CloudWatch

**Machine Learning**

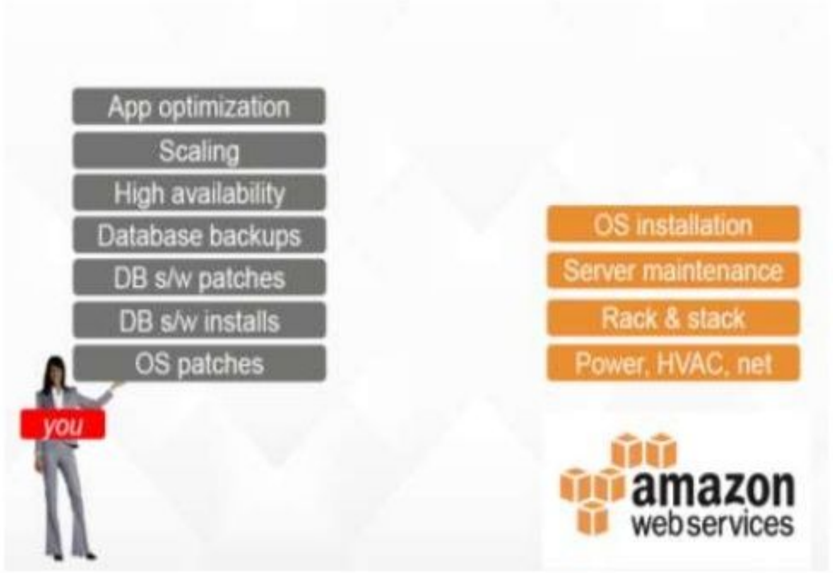
Amazon Machine Learning

## Serverless Database ?

MENTORMATE™
DevCamp



DB hosted on-premises



DB hosted on Amazon EC2

MENTORMATE™ DevCamp

## DynamoDB Benefits

- Fully managed
- Fast, consistent performance
- Highly scalable
- Flexible
- Event-driven programming
- Fine-grained access control

# Duolingo Scales to Store Over 31 Billion Items Using DynamoDB

Using AWS, we can handle traffic spikes that expand up to seven times the amount of normal traffic.

**Severin Hacker**
CTO, Duolingo

duolingo.

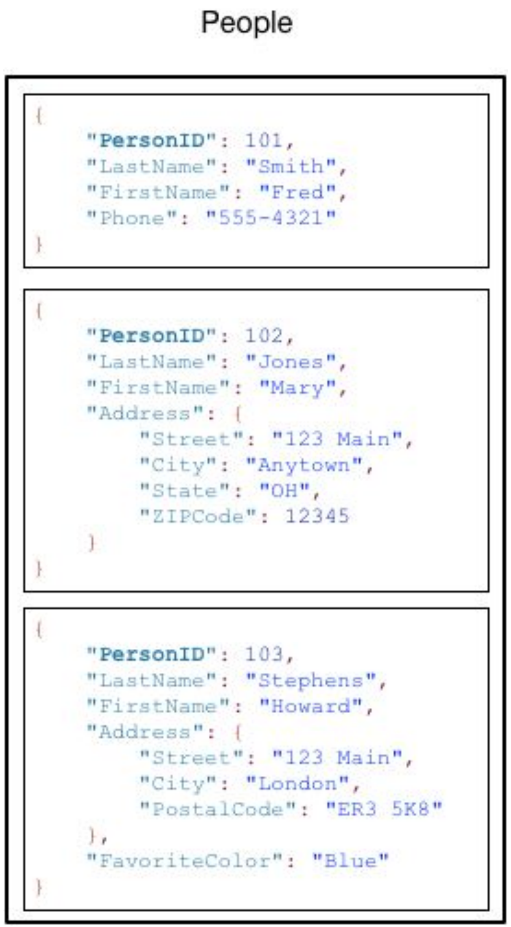
Duolingo is a free language learning service where users help translate the web and rate translations.

- Duolingo stores data about each user to be able to generate personalized lessons.
- The MySQL database couldn't keep up with Duolingo's rate of growth
- By using the scalable database service, data store capacity increased from 100 million to more than four billion items
- Duolingo has the capacity to scale to support over 8 million active users

Source: [This case study]

# What is DynamoDB?

- NoSQL database **tables** as a service
- Store as many **items** as you want
- Items may have different **attributes**
- **Low-latency** queries
- Scalable read/write **throughput**

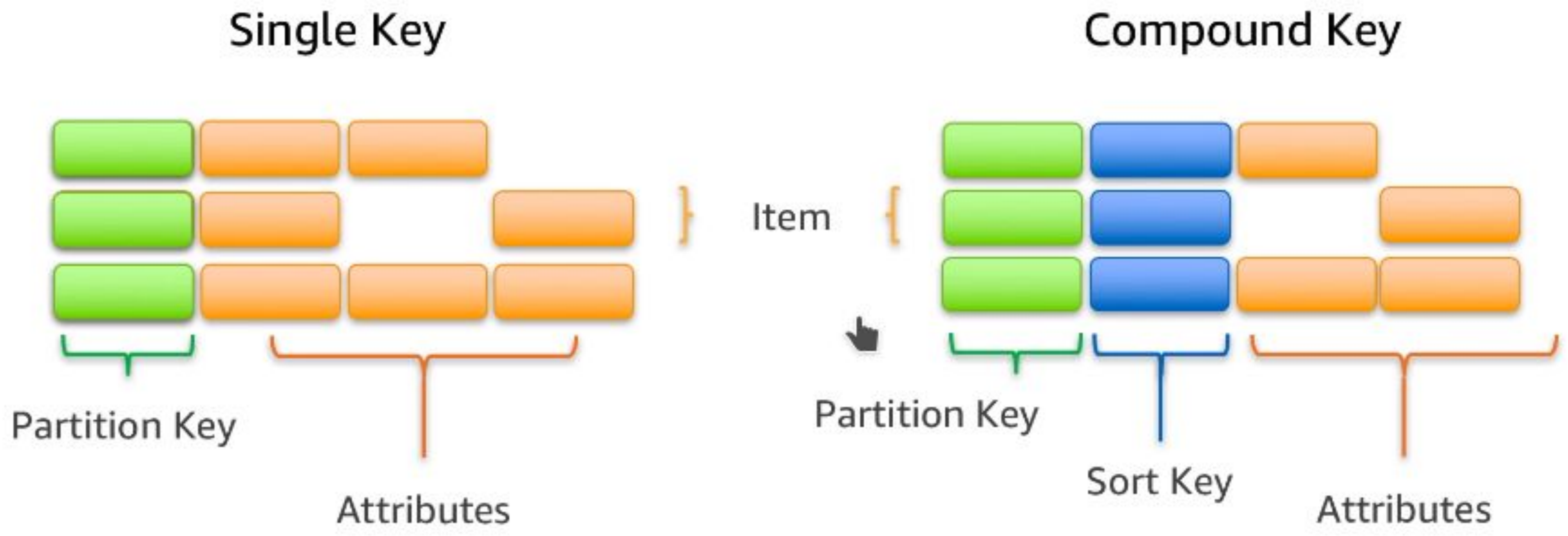


```
{
    "PersonID": 101,
    "LastName": "Smith",
    "FirstName": "Fred",
    "Phone": "555-4321"
}
```

```
{
    "PersonID": 102,
    "LastName": "Jones",
    "FirstName": "Mary",
    "Address": {
        "Street": "123 Main",
        "City": "Anytown",
        "State": "OH",
        "ZIPCode": 12345
    }
}
```

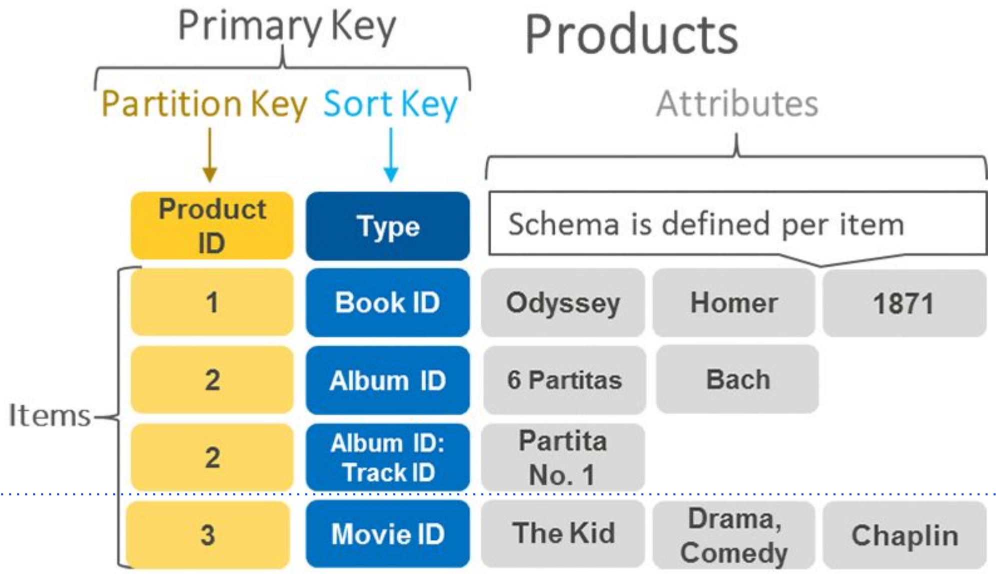
```
{
    "PersonID": 103,
    "LastName": "Stephens",
    "FirstName": "Howard",
    "Address": {
        "Street": "123 Main",
        "City": "London",
        "PostalCode": "ER3 5K8"
    },
    "FavoriteColor": "Blue"
}
```

# Items in a table must have a key



## Partition key

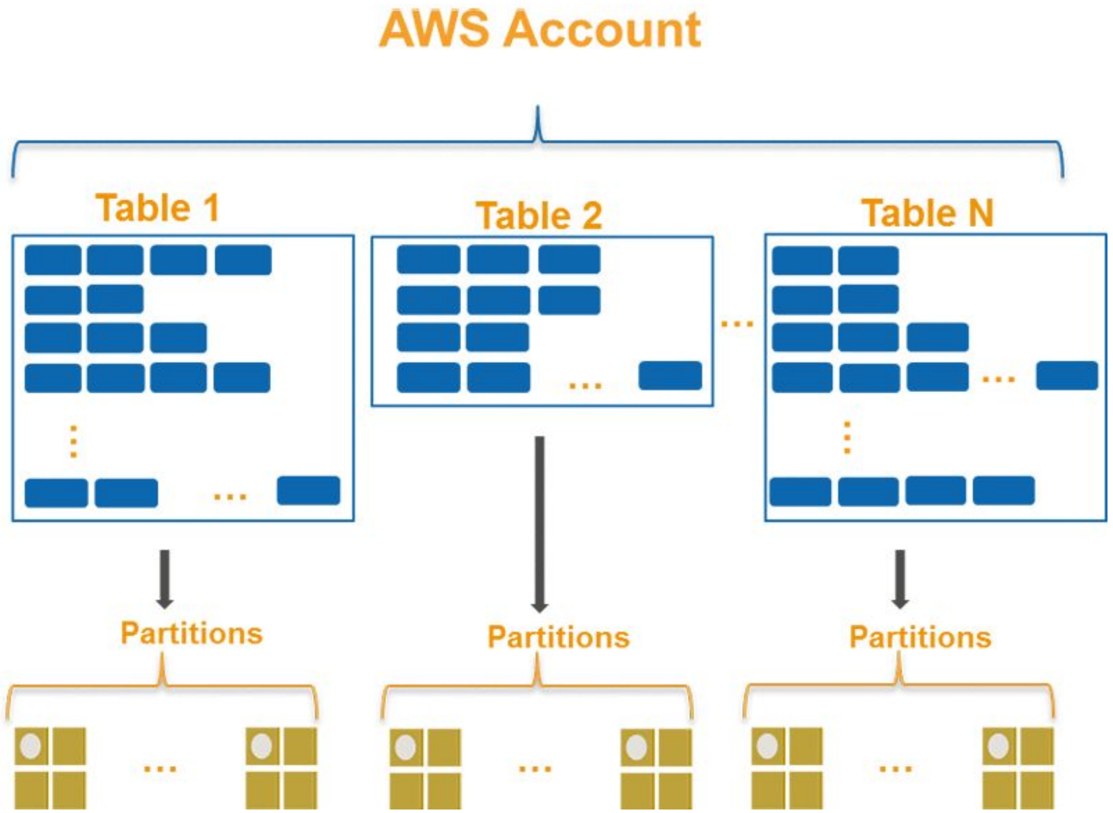
- **Recommendations** for partition keys:
    - Use high-cardinality attributes - **distinct** values for each item
    - Use composite attributes



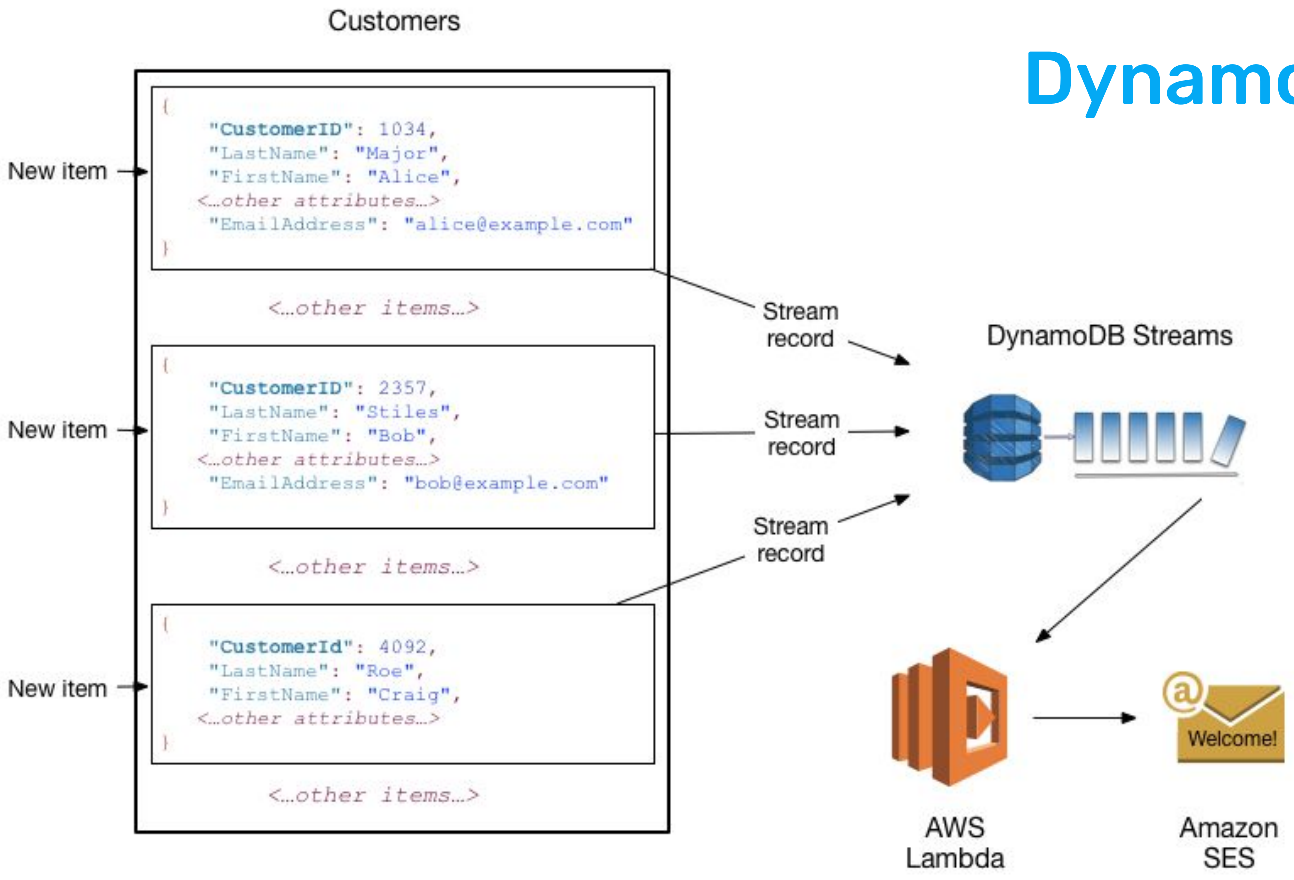
## Partitioning

- Items are **distributed** across 10-GB storage units, called **partitions** (physical storage internal to DynamoDB)
- Using low-cardinality attributes as the partition key and order_date as the sort key greatly increases the likelihood of **hot partition issues**.

  For example, if one product is more popular, then the reads and writes for that key is high, resulting in throttling issues.



# DynamoDB Streams



```
{
    "CustomerID": 1034,
    "LastName": "Major",
    "FirstName": "Alice",
  <...other attributes...>
    "EmailAddress": "alice@example.com"
}
```

```
{
    "CustomerID": 2357,
    "LastName": "Stiles",
    "FirstName": "Bob",
  <...other attributes...>
    "EmailAddress": "bob@example.com"
}
```

```
{
    "CustomerId": 4092,
    "LastName": "Roe",
    "FirstName": "Craig",
  <...other attributes...>
}
```

MENTORMATE™ DevCamp

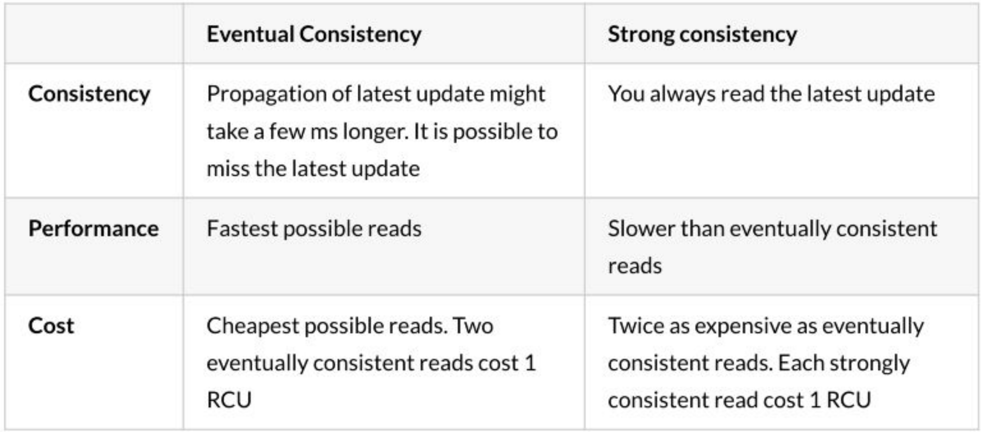
## Query vs Scan

| Scan | Query |
|---|---|
| No need to specify any key criteria | Need to specify Partition Key mandatorily |
| Navigates through all the items in a table | Navigates through all the items in a partition |
| Maximum limit of 1 MB per page scanned | Maximum limit of 1 MB per page queried |
| FilterExpression operation can be used to narrow down the results, post scan | Sort key can be specified to narrow down the results of query. In addition, FilterExpression operation can also be used to narrow down the results, post query |

**Avoid full table scans!**

# Other core concepts

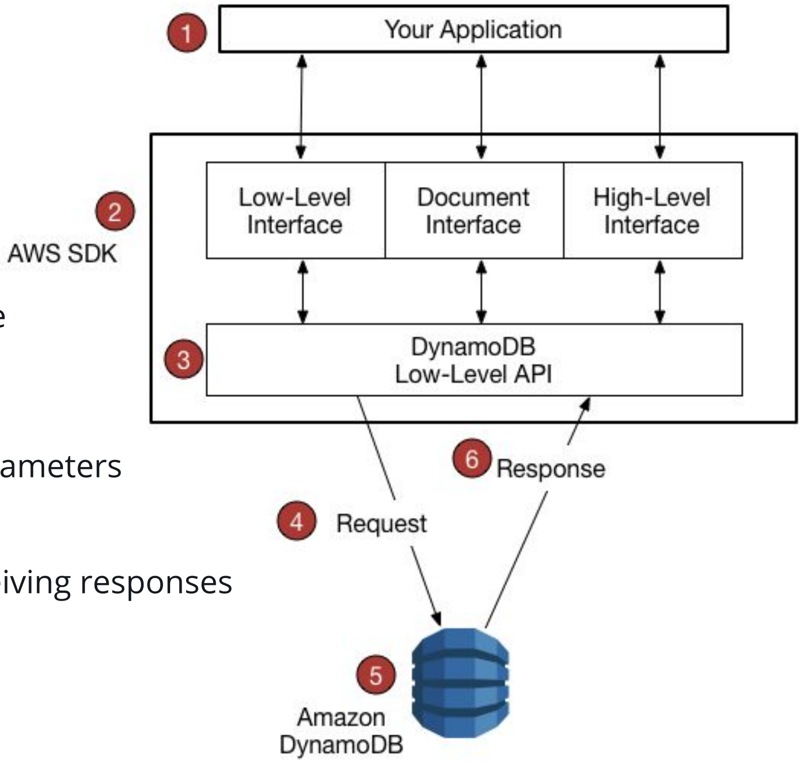
- **Secondary indexes** guide, explained
- **Strong** vs **Eventual** consistency
- Read/Write Capacity Mode guide



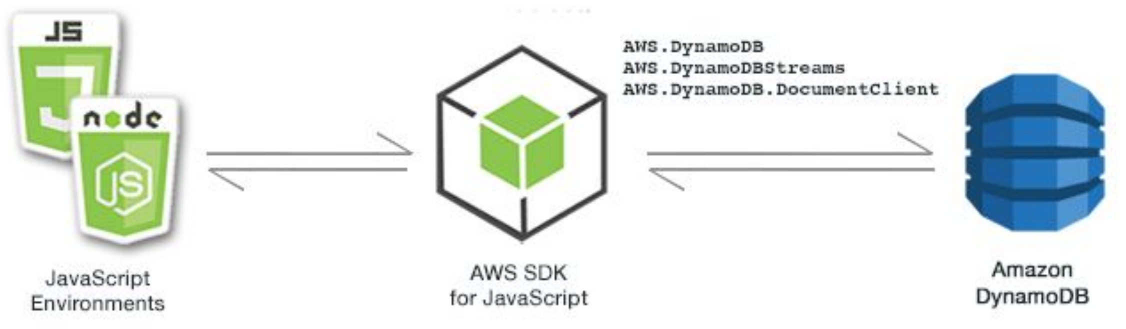
| | Eventual Consistency | Strong consistency |
|---|---|---|
| **Consistency** | Propagation of latest update might take a few ms longer. It is possible to miss the latest update | You always read the latest update |
| **Performance** | Fastest possible reads | Slower than eventually consistent reads |
| **Cost** | Cheapest possible reads. Two eventually consistent reads cost 1 RCU | Twice as expensive as eventually consistent reads. Each strongly consistent read cost 1 RCU |

# AWS SDK for DynamoDB

- How it works?
  - The AWS SDK constructs/sends HTTP(S) requests for use with the low-level DynamoDB API
  - DynamoDB executes the request. Returns an HTTP code
- AWS SDKs provides important services:
  - Formatting HTTP(S) requests and serializing request parameters
  - Generating a cryptographic signature for each request
  - Forwarding requests to a DynamoDB endpoint and receiving responses from DynamoDB.
  - Extracting the results from those responses
  - Implementing basic retry logic in case of errors



## Getting Started



- https://www.dynamodbguide.com/ - Learn DynamoDB
- Setting Up DynamoDB - To learn how to set up DynamoDB (the downloadable version or the web service).
- Working with DynamoDB - Tables, Items, Queries, Scans, and Indexes
- JavaScript SDK Documentation & Examples

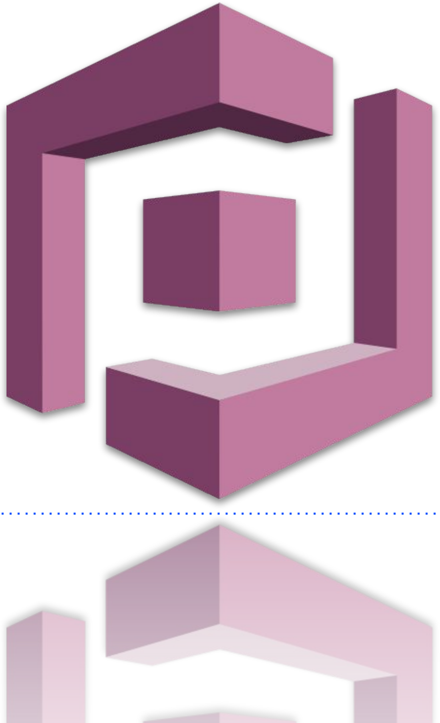
## DynamoDB DEMO

MENTORMATE™
DevCamp

## AWS Cognito

## Overview

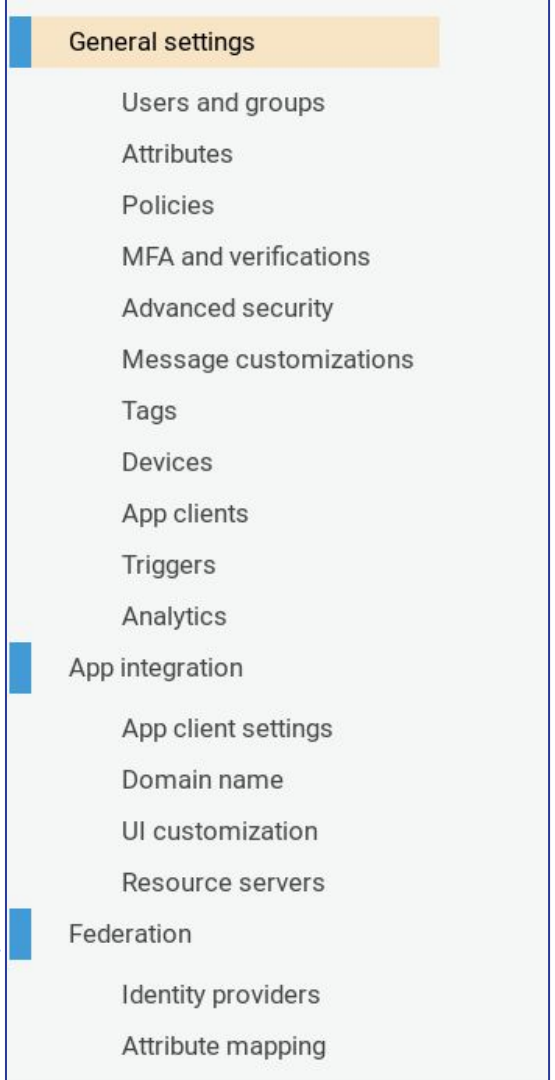
- Secure and scalable user directory
- Social and enterprise identity federation
- Standards-based authentication
- Security for your apps and users
- Access control for AWS resources
- Easy integration with your app
- Can be used as a standalone IdP



MENTORMATE™ DevCamp

## User Pools

- Sign-up and sign-in services, social sign in
- A built-in, customizable web UI
- Forgot password flow, email or phone number verification, MFA
- User directory management and user profiles
- Security features
- Customized workflows and user migration through AWS Lambda triggers.



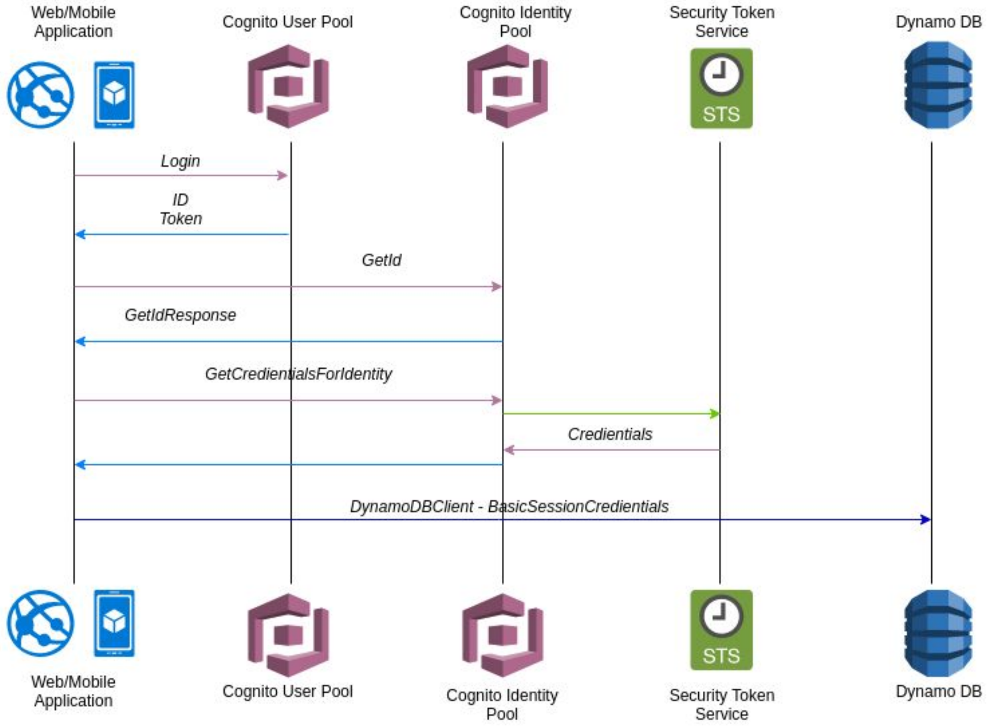
MENTORMATE™ DevCamp

## Admin Capabilities

- Create and manage User pools
- Define custom attributes
- Require Submission of Attribute Data
- Set per-app permissions
- Set up Password Policies
- Search users
- Manage users

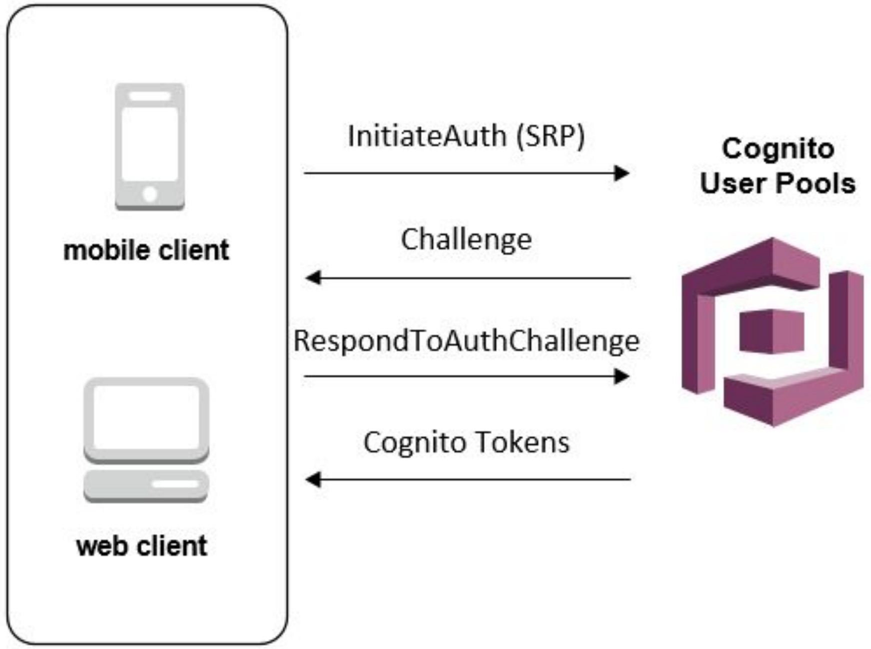
# Federated Identities

- Public Providers (Amazon, Google, Facebook)
- Amazon Cognito User Pools
- Open ID Connect & SAML Identity
- Developer Authenticated Identities



MENTORMATE™ DevCamp

## Simple Authentication



- Getting Started with User Pools
- SRP Explained
- Understanding Cognito for Serverless
- NPM Module for Node.js

## Cognito Demo

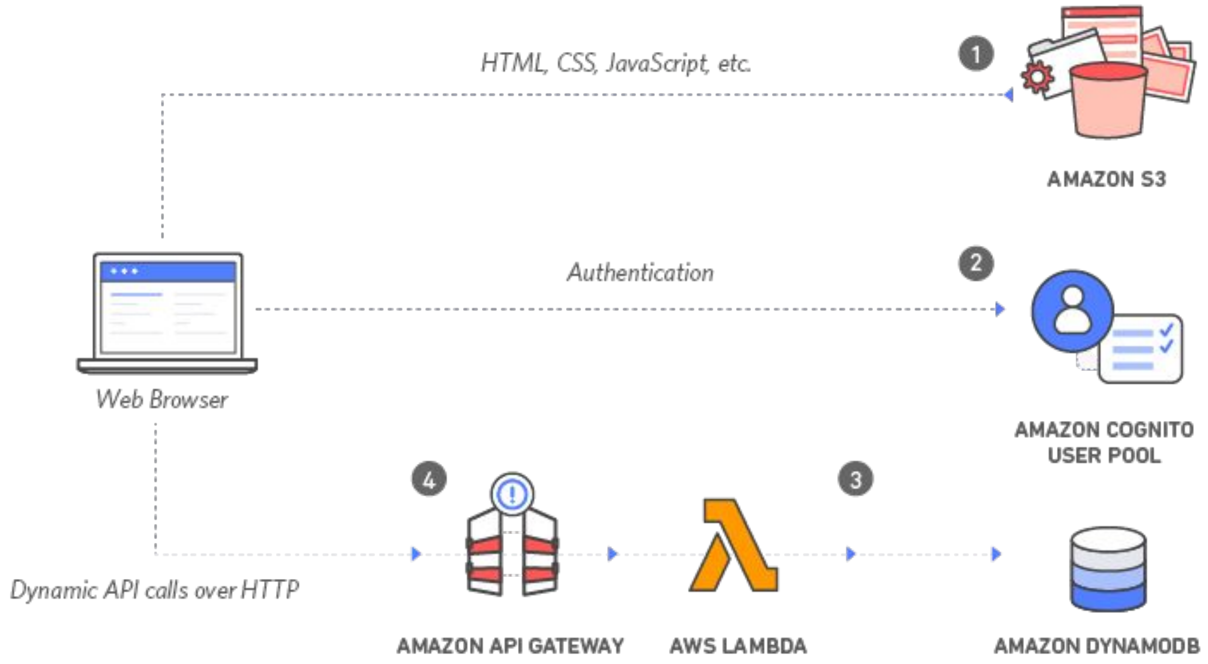
MENTORMATE™
DevCamp

## Integrate Cognito with API Gateway

MENTORMATE™
DevCamp

## Wire up

- Wild Rydes Tutorial



# Homework

1. **DynamoDB Practice**

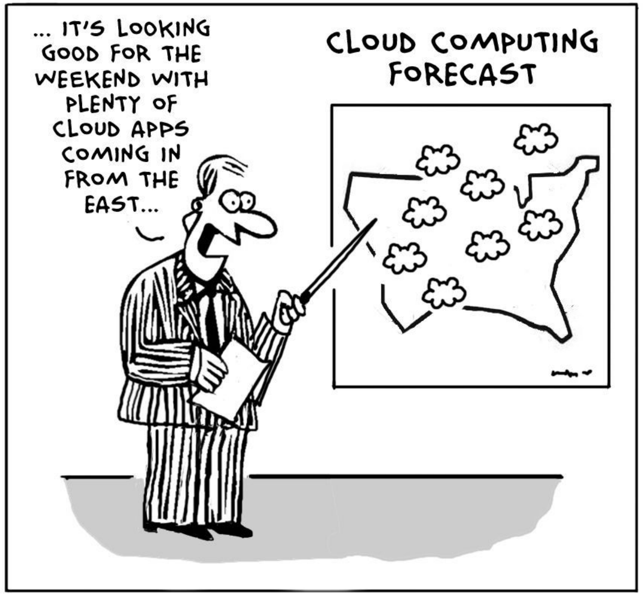
Using AWS-SDK & JavaScript, create a DynamoDB Table for hobbies/skills. You can use a shared table with another Trainee ;) Decide what your partition/sort keys should look-like.

Example schema:
Name - *String*
Description - *String*
Practitioner - Your name
Since - *Date*
Rating - *Integer from 1 to 10*
Expertise - *ENUM ('novice', 'advanced beginner', 'competent', 'proficient', 'expert')*

Create a simple REST API for managing these hobbies (CRUD operations).

Create endpoints that return data via more advanced queries, for example:

- Show practitioner with the most hobbies
- Get top skill per practitioner (user)
- List top 5 favorite hobbies / activities / skills
- … be creative!


... IT'S LOOKING GOOD FOR THE WEEKEND WITH PLENTY OF CLOUD APPS COMING IN FROM THE EAST...
CLOUD COMPUTING FORECAST