AWS DynamoDB

- https://cloudacademy.com/amazon-web-services/working-with-amazon-dynamodb-course/what-is-dynamodb.html
- https://acloud.guru/learn/aws-dynamodb
- Nosql DB
- All management will be held by AWS even backup
- We just need to setup a tables
- Charge total amount of throughput that configure for table + storage
- Use key value stored
- Flexible schema for column
- All data will be replicated in 3 AZ with in region.
- In case of outage Dynamodb route to different AZ
- Fast performance
- Infinity scalable
- Data is eventually consistent
- Limited data types
- Max item size: 400KB
- 10 indexes per table
- Performance limited to provisioned throughout level
- ---
- Use API calls like PUT & GET
- Eventually consistent (take time to replicate data to all 3 AZ)
- Always hosted on AWS. hybrid scenario is not fit with this DB
- Document store
- Transparently scalable.
- ---
- ----------Using DynamoDB in Your Application----------
- Interacting with DB: UI OR programmably
- Also using Custom software that use API.
- 13 different method of dynamoDB API, scaled with 3 categories.
- + Managing Tables (List, describe, create,update, delete)
- + Reading Data (Get Item, Batch Item, Query, Scan)
- + Modifying Data (Put Item, update Item, Delete Item, Batch written Item)
- It has a HTTP Endpoint, that why we refer as web service
- ---
- ----------**Create DynamoDB Tables**----------
- ------Creating a 1st Table---------
- Creating table practically
- Create table name: Orders. Primary key shows the individual records in the table
- Primary keys have 2 parts: Partition key & sort key

- Primary key: Partition key <order ID>: e.g Number (Use user_ID for high score game)
- 5sec Read & write capacity per second (Default)
- Increase read/write will incur a cost
- Table status: Active means ready to use
- -------Creating a table with a composite Primary key----------
- Partition key would be the username
- Sort key would be the timestamp (Game title)
- Other example of online shopping

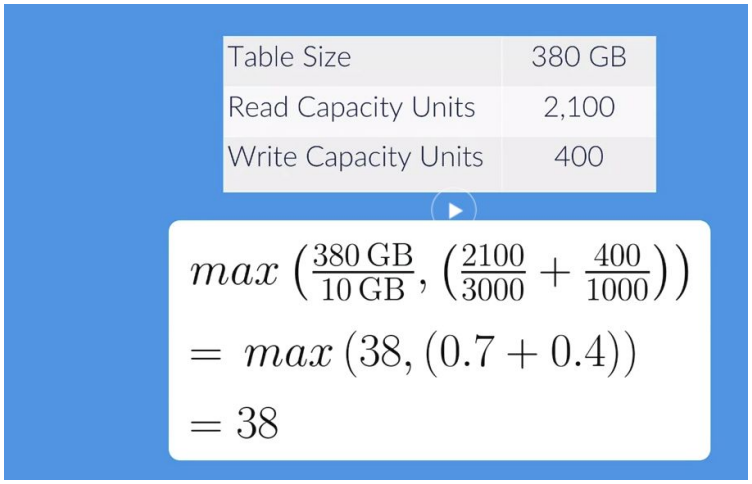
+ Partition_Key: Order_ID
+ Partition_Key & Sort_Key are inside the “Primary Key”
+ Sort key: Line_Number
+ Lab: Create Primary key (Order_ID) , create sort key (Line_Number)

---------**Understanding Provisioned throughput**----------

- You need to inform amazon how much capacity you reserved
- Amazon will charged Read & write capacity units you allocated
- 1 read capacity (RCU) will let you know

★ One item
★ Upto 4KB in size
★ With strong consistency
★ Each second
★ Eventually consistent reads cost half as much
★ 1 write capacity (WCU) will let you store
★ One item
★ Upto 1KB in size
★ Each second

- Lab: Metrics tab (will show capacity)
- Capacity tab: if increased it will show “provisioned capacity is updating’
- ----------Read & Write Data----------
- Split into many partitions
- Lab: How to add a key value. How to modify existing key values. Remove attributes
- We can set the data types for each attributes
- Use “Map” attribute value for nested hierarchy
- DynamoDB is schemaless, so the only key will be show is “ Partition Key”.
- ---------- **Manipulating data using API** ---------developer lab
- Use python to write code
- Use boto3 to write code
- Required 2 parameters: “Name of table” & primary key to load the data
- ---------- **Understanding Queries & Scans** ----------
- A query searches the table for **single partition key**
- Results can be ordered by sort key
- Eventually consistent data
- Scan can be filtered on any attributes
- Scan searches across all partition keys

- Scans are always eventually consistent.
- Scans can be run in parallel
- ---------- **Queries & Scans in AWS Console** ----------
- Lab: Search by attributes & its value
- ---------- **Understanding Secondary indexes** ----------
- 2 types of secondary indexes
- Global secondary indexes: Let your query across the entire table
- GSI can be created with the table or later.
- Throughput is provisioned separately for each GSI
- Limit of 5 GSI per table
- Local Secondary Indexes: Let your query within a single partition key
- LSI must be created when the table is created
- Throughput is shared with the main table
- Limit of 5LSI per table
- ------ **Creating Secondary Indexes** ----------
- By default the index we create is GSI
- Lab:
- ---------- **Querying Secondary Indexes** ----------
- Lab: scan GSI & LSI by selecting options
- Use increase & Decrease based..
- ---------- **Introduction to partitioning** ----------
- 

- It has 38 Partitions
- If 10.1 It means ~11 Partitions
- Increase at 2 x n =
- Partition relies on Right & Left.
- Amazon doesn't use Partition keys with AWS Customers
- Each partition can hold 10GB of data.
- ----- **Balancing Partition in Large Tables** ----------
- Partition can be go on burst capacity
- DB support optimistic locking

- Use conditional writes for consistency.
-