# UNCLEAR LAB (objectives) - INCOMPLETE

# 

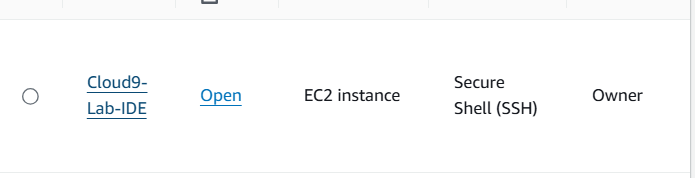
# Lab 1 - Implement Fleet and Trip Data Management using Amazon DynamoDB Tables, Indexes, and Change Streams

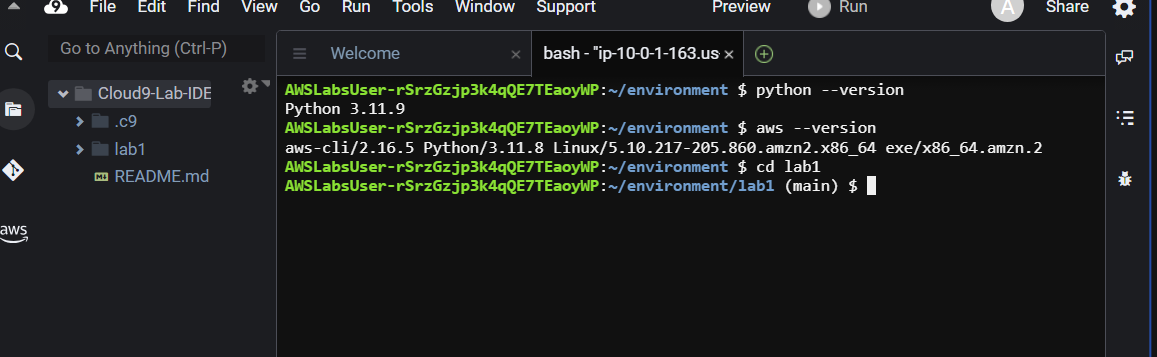
1. Design a modern application on an AWS NoSQL database, DynamoDB.
2. Implement DynamoDB data APIs to manage application data in DynamoDB tables.
3. Implement an event-driven architecture by using DynamoDB Streams as an event source for AWS Lambda.

Event-Driven Architecture: You'll set up DynamoDB such that it captures changes (events) happening in DynamoDB tables in real-time. These events are then used as triggers to perform actions

**Task 1: Review the development environment**

1.1 go to cloud9, and open

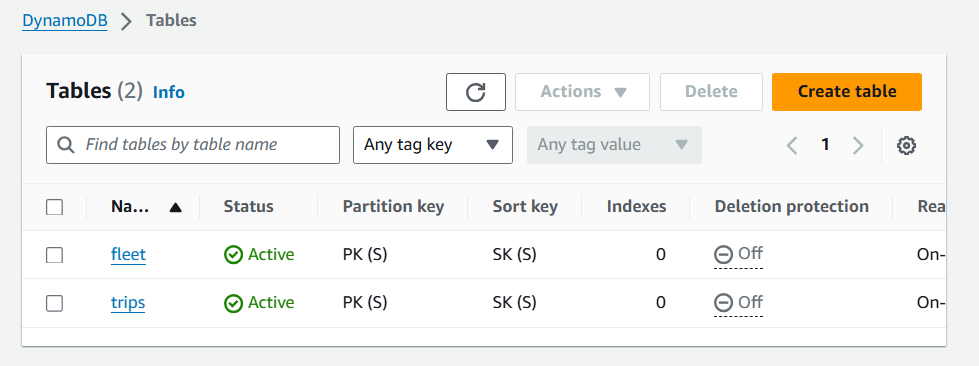
****



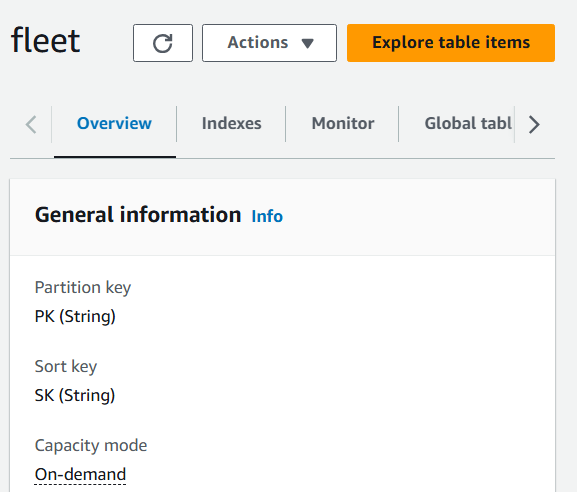
**Task 2: Review the DynamoDB tables**

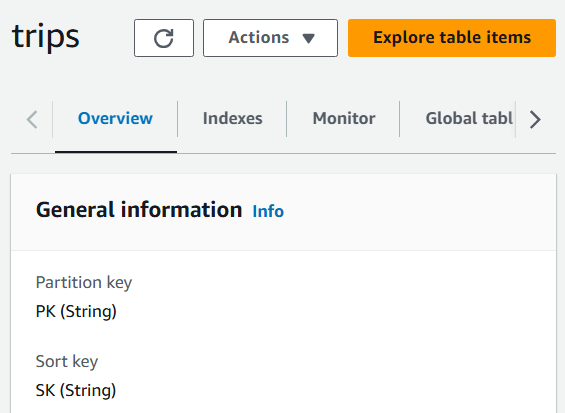


2.1 open dynamoDB, and see the tables



2.2 open the fleet table



2.3 

**Task 3: Implement access patterns**

implement a high-performing and an effective solution for the defined access pattern to find battery-powered assets with a LOW\_BATTERY status.

The fleet application has been updated to add the GSI1\_PK attribute to LOW\_BATTERY when the bike reports a low battery condition. The attribute is deleted when the bike is charged.

GSIs are created by defining a partition key and optionally a sort key

3.1 This AWS CLI command scans the DynamoDB table "fleet", filters items where the GSI1\_PK attribute equals "LOW\_BATTERY", and returns the count of matching items while tracking the consumed capacity.

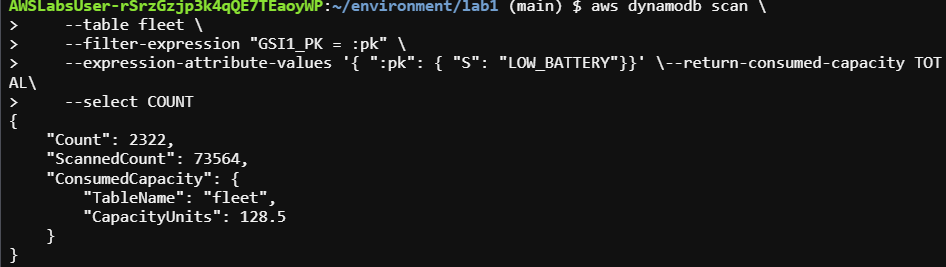
aws dynamodb scan \

--table fleet \

--filter-expression "GSI1\_PK = :pk" \

--expression-attribute-values '{ ":pk": { "S": "LOW\_BATTERY"}}' \--return-consumed-capacity TOTAL\

--select COUNT



Finding the bikes with a status of LOW\_BATTERY requires a scan operation on the fleet table and a filter on the status attribute. This is not an effective method

3.2 optimize finding low battery assets

3.3 in cloud9, make a new file

[

{

"Create": {

"IndexName": "GSI1",

"KeySchema": [

{

"AttributeName": "GSI1\_PK",

"KeyType": "HASH"

},

{

"AttributeName": "SK",

"KeyType": "RANGE"

}

],

"Projection": {

"ProjectionType": "ALL"

}

}

}

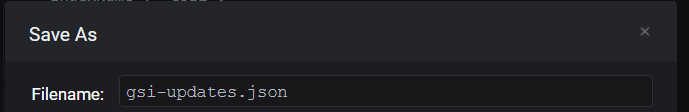
]

GSI1\_PK: This attribute is used as the hash key (partition key) for the index.

SK: This attribute is used as the range key (sort key) for the index.

this JSON snippet defines an index named "GSI1" on the DynamoDB table "fleet", with "GSI1\_PK" as the hash key and "SK" as the range key. It includes all attributes from the base table in the index.

3.4 save



3.5 run

aws dynamodb update-table \

--table-name fleet \

--attribute-definitions \

AttributeName=GSI1\_PK,AttributeType=S \

AttributeName=SK,AttributeType=S \

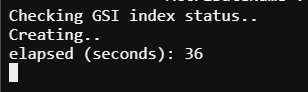
--global-secondary-index-updates file://gsi-updates.json

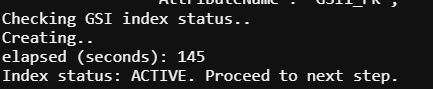
The command updates the DynamoDB table "fleet" by adding a new global secondary index (GSI) named "GSI1". This index is defined with "GSI1\_PK" as the hash key and "SK" as the range key, stored in gsi-file.

3.6 check status of GSI

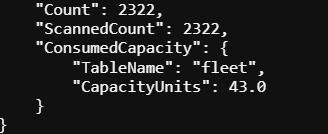
chmod +x \*.sh && chmod +x \*.py

./gsi\_status.sh





3.7 query for bikes with a LOW\_BATTERY status, using GSI



Reduced capacityunits needed



We do similar task of creating GSI for trips table too