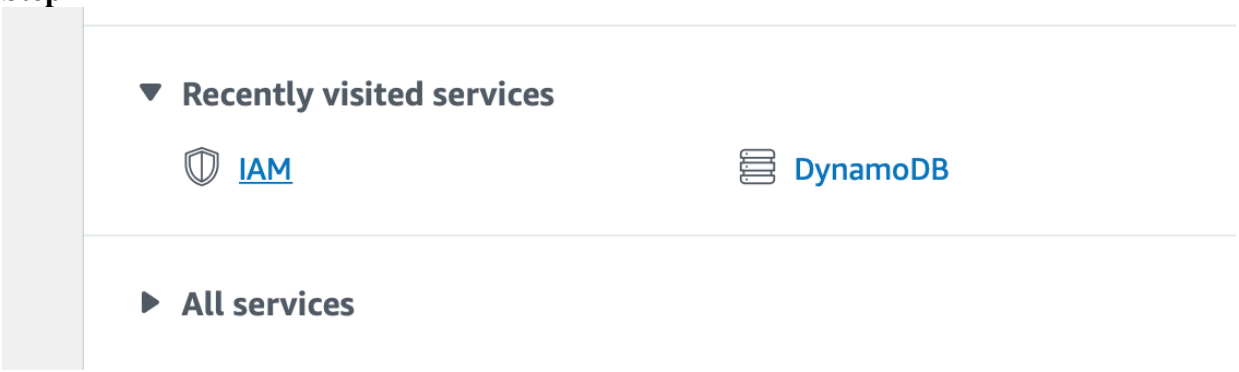
Uploading IoT Sensor Data to DynamoDB using Python and Design Pattern

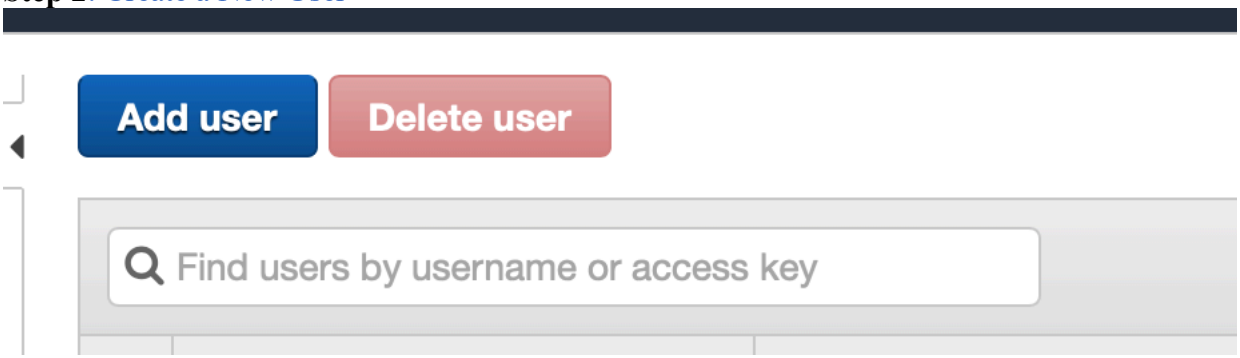
Aim: Upload IoT Sensor Data to AWS DynamoDB

Objective: In this Lab we will learn how to upload sensor data to DynamoDB using Python Bopto3 and Facade Design Pattern

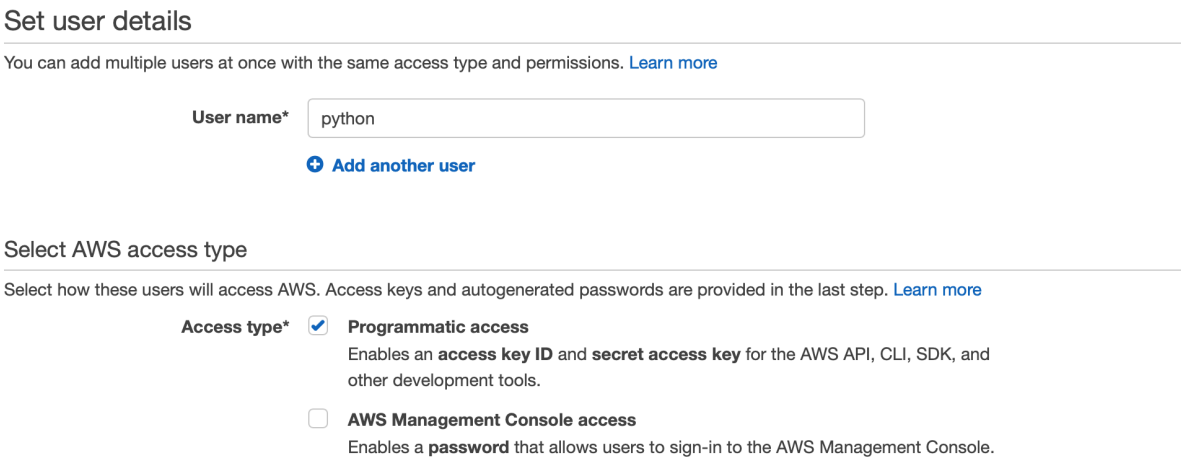
Step 1: Create a Free Tier Account on AWS and. Create a User



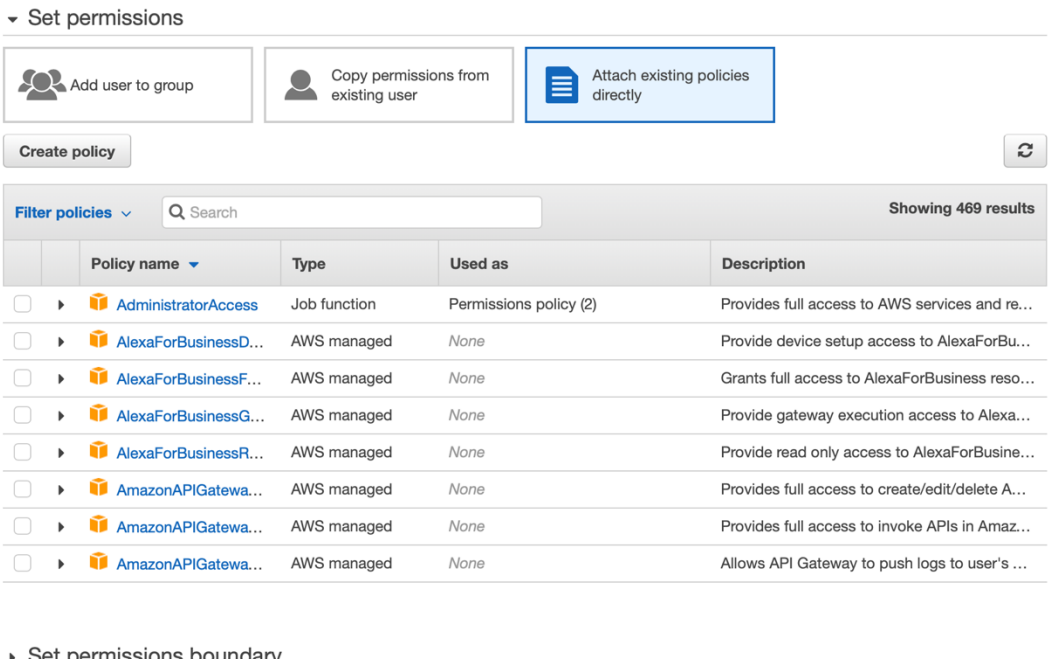
Step 2: Create a New User



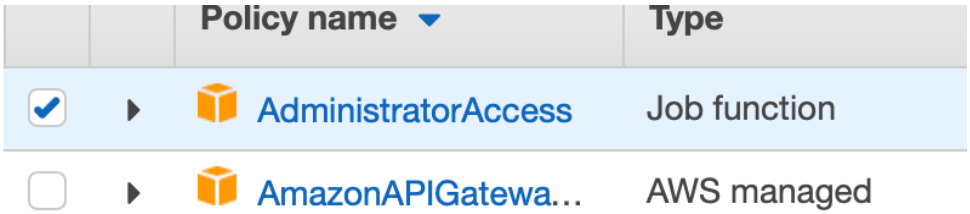
Step 3: Add username as Python and make sure access type is as Programmatic



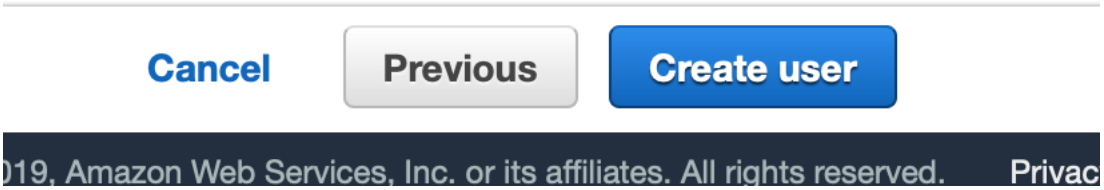
Step 4: Create a Policy Make sure to Give Admin Access



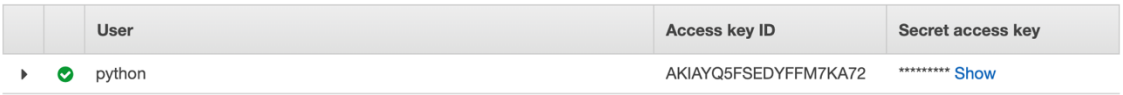
Click on Administrative Access



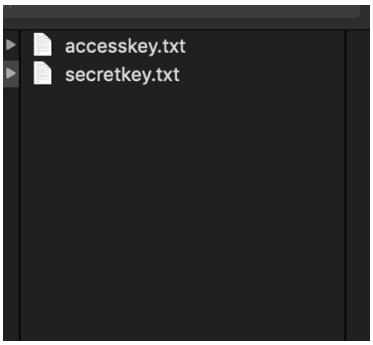
Step 5: Create the user



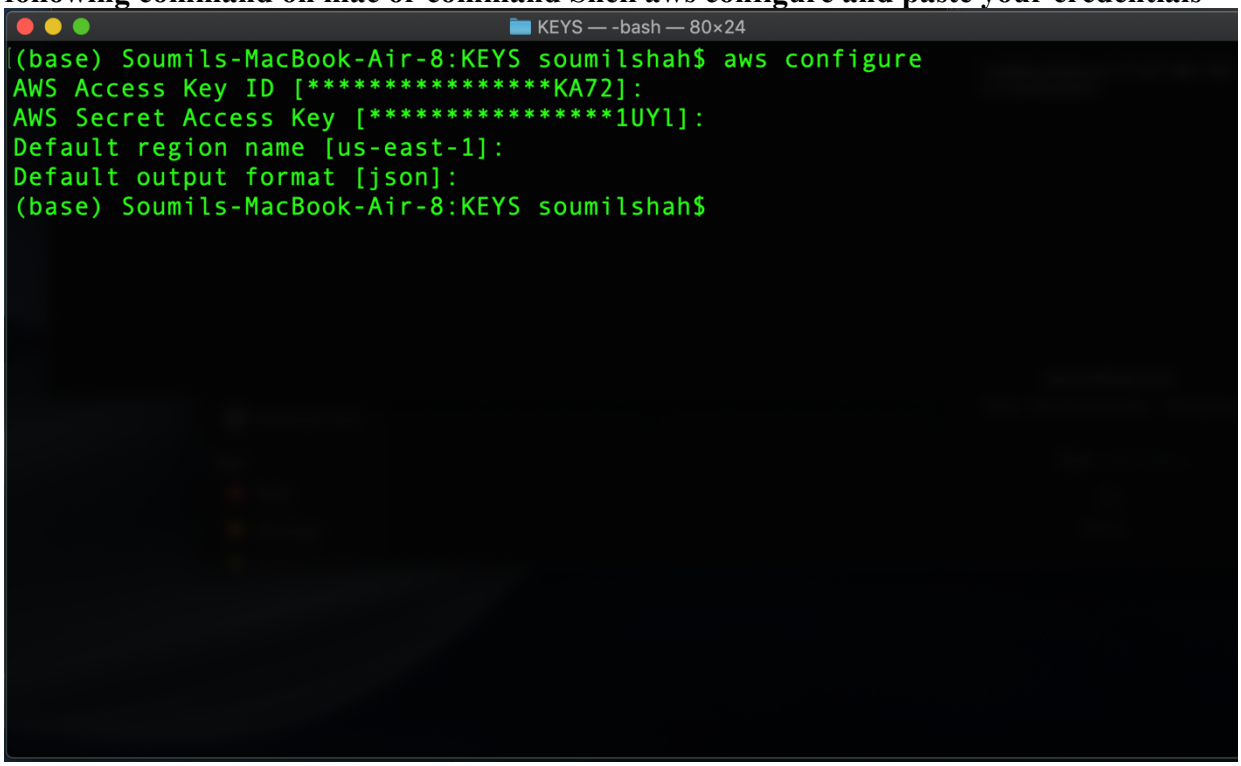
Step 7: Copy the Secrete Key and Access Key



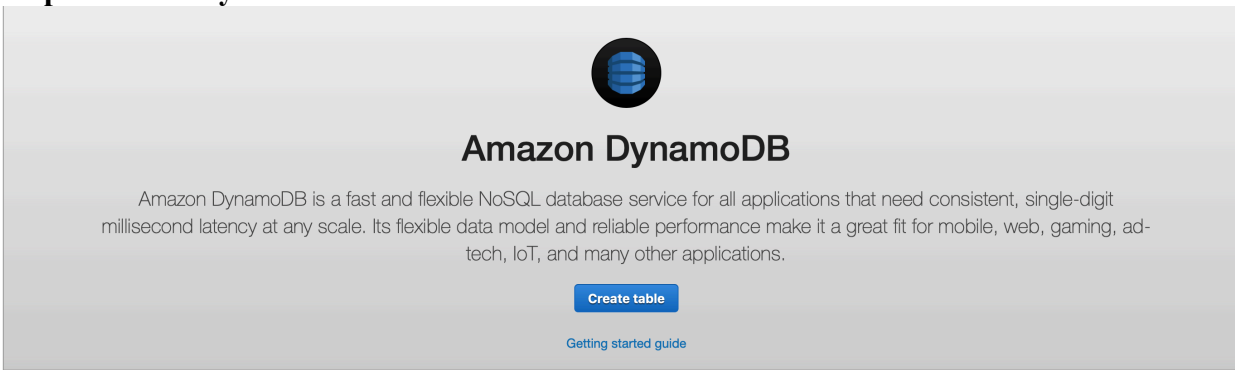
Step 7: Create a folder and store the access key and secret key in that folder



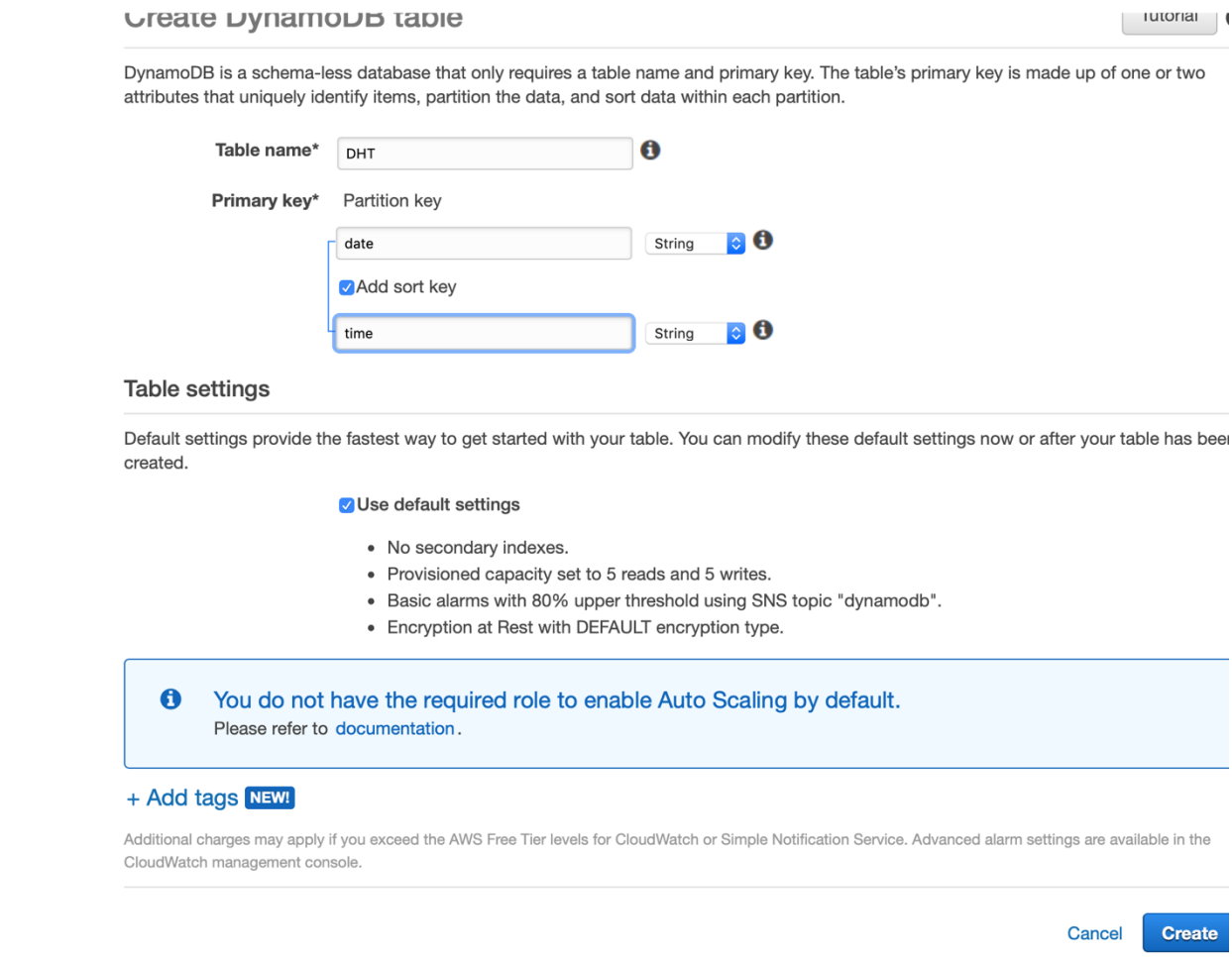
Step 8: Configure AWS CLI if you haven't installed pip install aws-cli or go the the documentation to download AWS CLI once downloaded we need to configure type the following command on mac or command Shell aws configure and paste your credentials



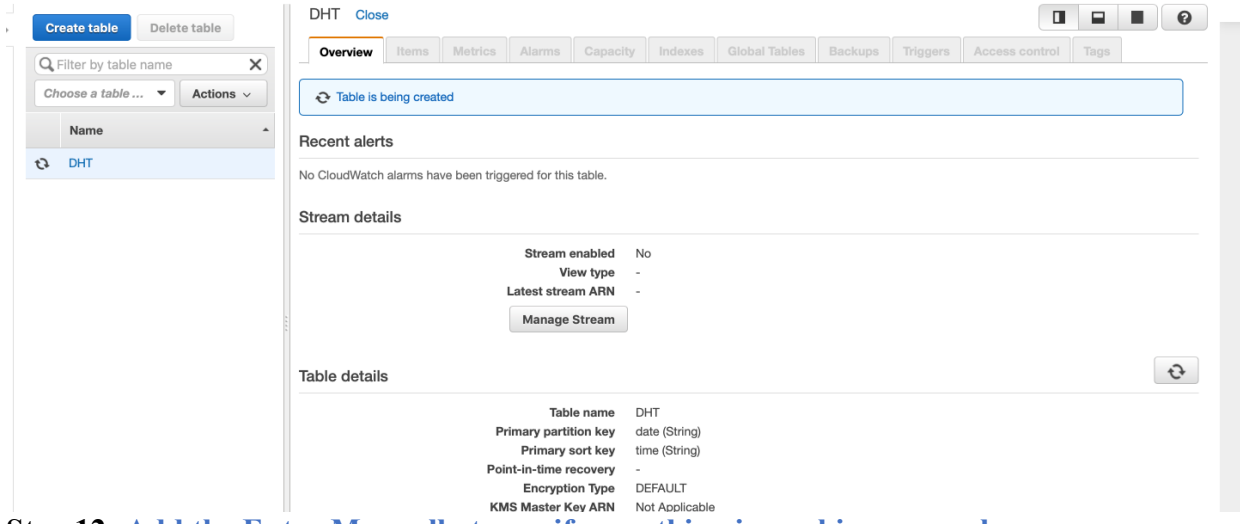
Step 9: Go to Dynamo DB



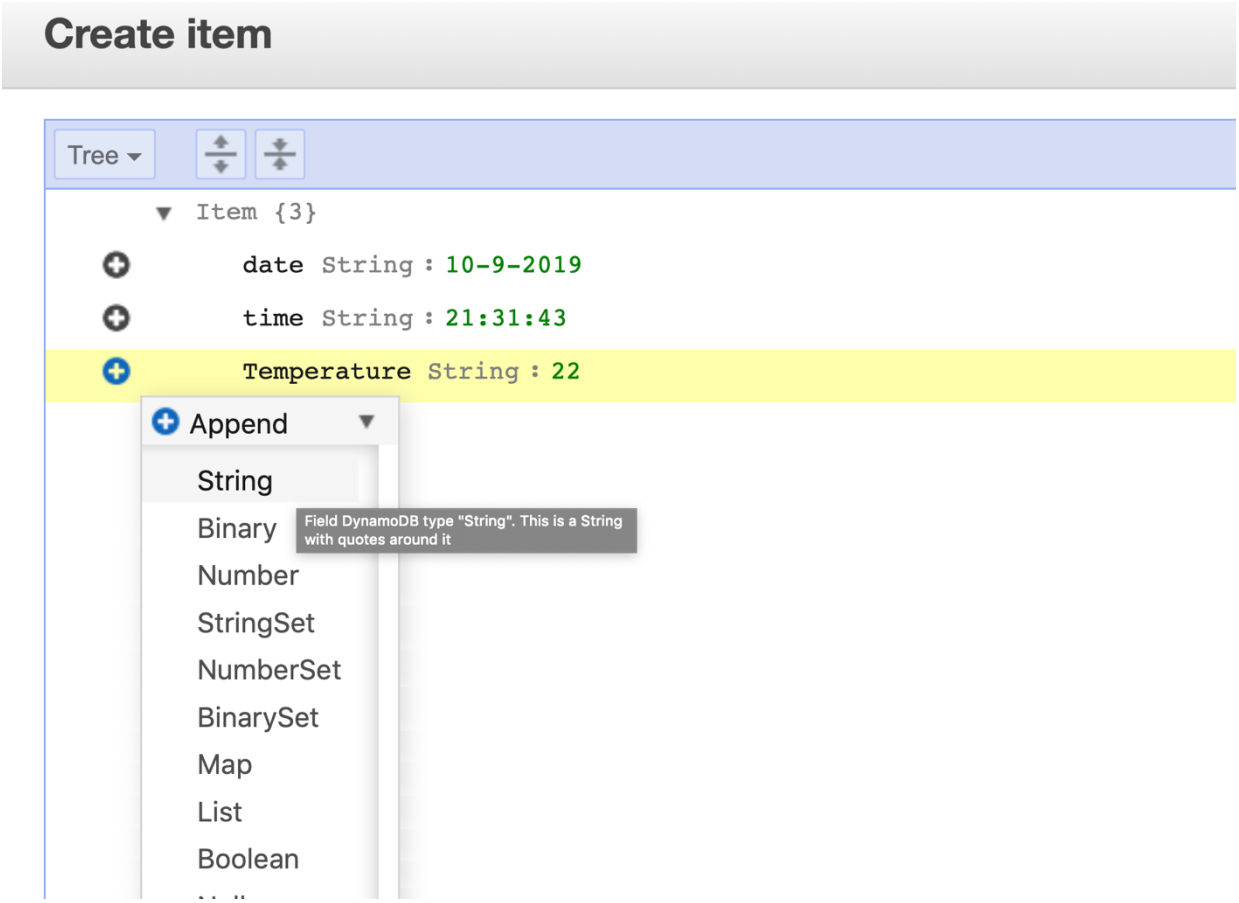
Step 10: Create a table with following name



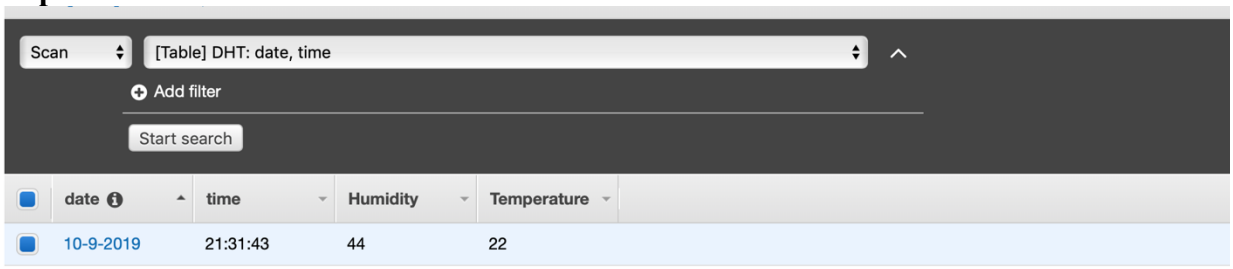
Step 11: Click on create button to create a table



Step 12: Add the Entry Manually to see if everything is working properly

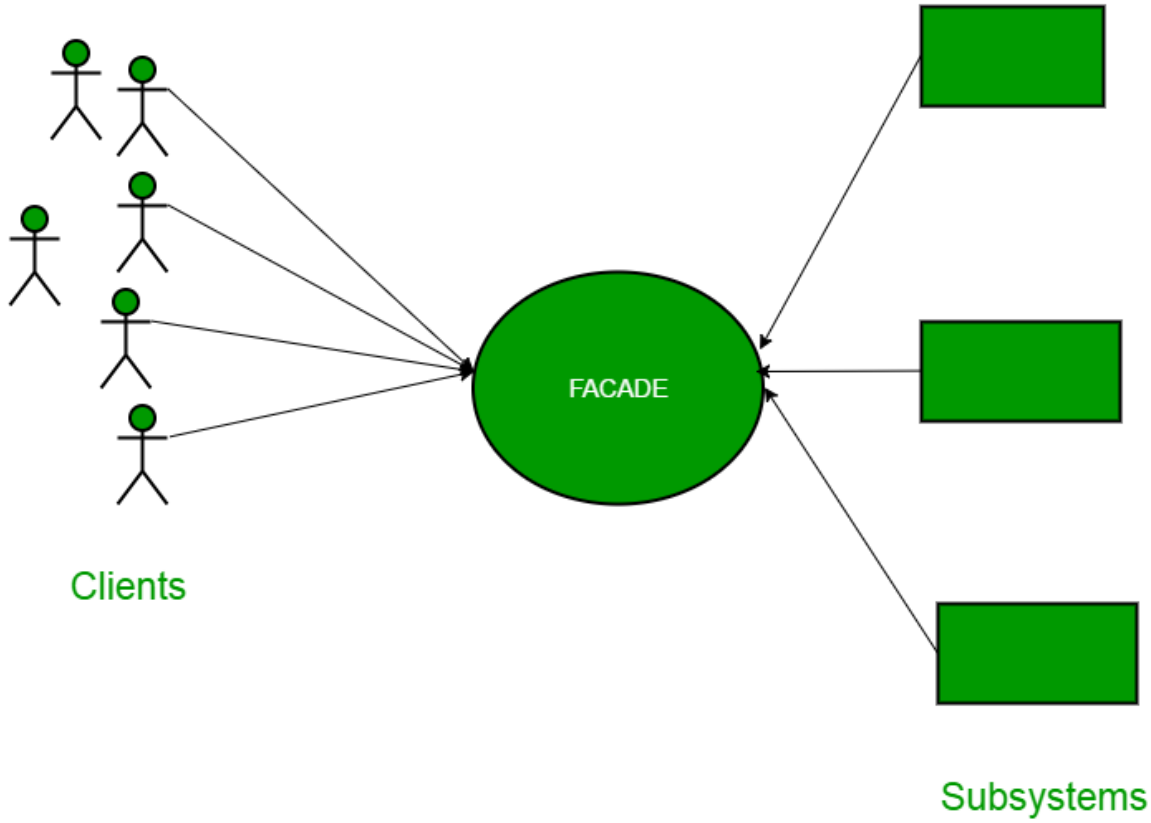


Step 13:



Congrats you have just added data on AWS Dynamo DB

Let’s us use Façade and Singleton Design Pattern in Python to Upload the Data on Dynamo DB

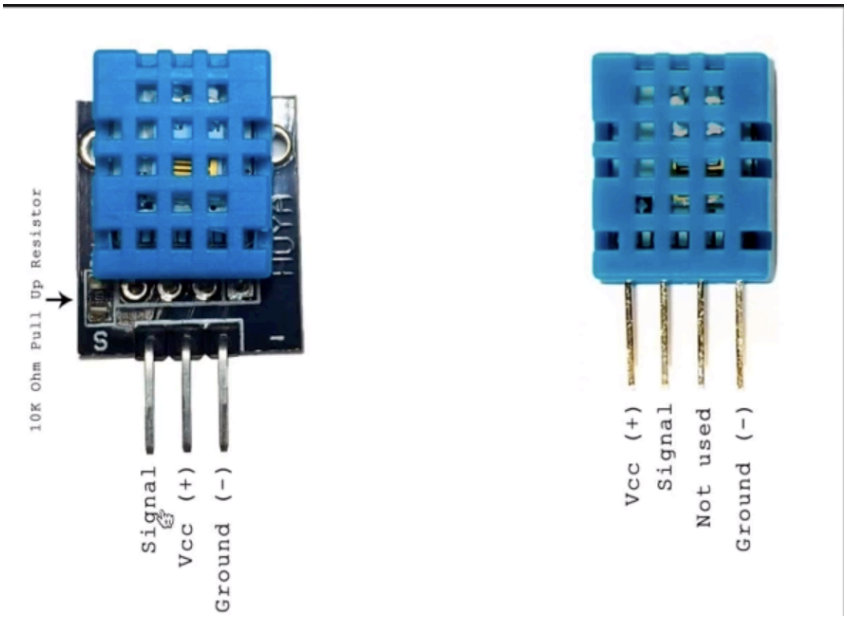


Code :

```
"""

We will use Fascade Design Pattern and Singleton Design Pattern to Upload the Sensor Data

"""

try:

    import os
    import sys
    import boto3
    import datetime
    import random

except Exception as e:

    print("Some Modules are Missings ")


class Metaclass(type):

    """ Implementing Singleton Design Pattern """

    _instance = {}

    def __call__(cls, *args, **kwargs):
```

```
        if cls not in cls._instance:
            cls._instance[cls] = super(Metaclass, cls).__call__(*args, **kwargs)
            return cls._instance[cls]


class Sensor(object):

    def __init__(self):
        """ Constructor """
        pass

    def get(self):

        """ Get the Sensor Data  """

        Temperature = random.randint(0,70)
        Humidity  = random.randint(0,70)
        d = datetime.datetime.now()
        _Date = "{}-{}-{}".format(d.month, d.day, d.year)
        _Time = "{}:{}:{}".format(d.hour, d.minute, d.second)
        return Temperature, Humidity, _Date, _Time


class DynamoDb(object):

    __slots__ = ["Table_Name", "db", "table", "client"]

    def __init__(self, Table_Name='DHT'):
        self.Table_Name=Table_Name
        self.db = boto3.resource('dynamodb')
        self.table = self.db.Table(Table_Name)
        self.client = boto3.client('dynamodb')

    def put(self, date='' , time ='',Temperature='', Humidity=''):
        self.table.put_item(
            Item={
                'date':date,
                'time':time,
                'Temperature':Temperature,
                'Humidity' :Humidity
            }
        )


class Facade(metaclass=Metaclass):

    def __init__(self):
        self._sensor = Sensor()
        self._database = DynamoDb()

    def upload(self):
        Temperature, Humidity, _Date, _Time = self._sensor.get()
        self._database.put(_Date, _Time, str(Temperature), str(Humidity))
        data = {
            "Date":_Date,
            "Time":_Time,
            "Temperature":Temperature,
            "_humidity":Humidity
        }
        print(data)

def main():
    facade = Facade()
    facade.upload()


if __name__ == "__main__":
    main()
```

Connecting Actual Sensor and Logging Value to Dynamo DB



Shows the Pinout of DHT-11 Sensor

Connect the Data Pin to GPIO 23 on raspberry pi and VCC and GND to respective pins. Once hooked up configure the AWS CLI on Raspberry pi and run the Following Python Code which will upload the sensor data to DynamoDB

```
"""

We will use Fascade Design Pattern and Singleton Design Pattern to Upload the Sensor Data

"""

try:

    import os
    import sys
    import boto3
    import datetime
    import random
    import Adafruit_DHT
except Exception as e:
```

```
    print("Some Modules are Missings ")


class Metaclass(type):

    """ Implementing Singleton Design Pattern """

    _instance = {}

    def __call__(cls, *args, **kwargs):
        if cls not in cls._instance:
            cls._instance[cls] = super(Metaclass, cls).__call__(*args, **kwargs)
            return cls._instance[cls]


class Sensor(object):

    def __init__(self):
        """ Constructor """
        pass

    def get(self):

        """ Get the Sensor Data  """
        pin = 23
        sensor = Adafruit_DHT.DHT11
        Humidity, Temperature = Adafruit_DHT.read_retry(sensor, pin)

        # Temperature = random.randint(0,70)
        # Humidity  = random.randint(0,70)

        d = datetime.datetime.now()
        _Date = "{}-{}-{}".format(d.month, d.day, d.year)
        _Time = "{}:{}:{}".format(d.hour, d.minute, d.second)

        return Temperature, Humidity, _Date, _Time


class DynamoDb(object):

    __slots__ = ["Table_Name", "db", "table", "client"]

    def __init__(self, Table_Name='DHT'):
        self.Table_Name=Table_Name
        self.db = boto3.resource('dynamodb')
        self.table = self.db.Table(Table_Name)
        self.client = boto3.client('dynamodb')

    def put(self, date='' , time ='',Temperature='', Humidity=''):
        self.table.put_item(
            Item={
                'date':date,
                'time':time,
                'Temperature':Temperature,
                'Humidity' :Humidity
            }
        )


class Facade(metaclass=Metaclass):

    def __init__(self):
        self._sensor = Sensor()
        self._database = DynamoDb()

    def upload(self):
        Temperature, Humidity, _Date, _Time = self._sensor.get()
        self._database.put(_Date, _Time, str(Temperature), str(Humidity))
        data = {
```

```
            "Date":_Date,
            "Time":_Time,
            "Temperature":Temperature,
            "_humidity":Humidity
        }
        print(data)

def main():
    facade = Facade()
    facade.upload()


if __name__ == "__main__":
    main()
```