Objective

In this exercise we will explore -

1. DynamoDB table modeling

2. Python code leveraging AWS library Boto3

Background

About a decade back energy utility company "North Power Corporation" (NPC) had invested in upgrading the power grid. The following were the prime objectives -

1. All substations must have smart meters and transformers

2. These devices need to transmit instantaneous parameters (IP) every 15 mins

3. Meters will transmit consumption data (CD) every 60 mins

4. Business systems (BS) in head office needs to have a database to persist the transmitted data and have an online application for engineers to monitor the substation.

The implementation was a success and the technology stack included Python and MySQL as the database (among others). The application and servers is being managed by NPC engineering team.

Now, NPC wants to take advantage of the cloud and is evaluating if this usecase is a good fit for migration. In addition, NPC intends to adopt a NoSQL because the data volume has grown and will continue to grow. A proof of concept (POC) needs to be done to determine the next steps.

POC scope

After several internal discussions the engineering team has built consensus on the following -

1. Cloud provider will be AWS

2. Evaluate DynamoDB as the backend datastore

3. Alleviate the need of servers

Meanwhile, another team has already completed the POC for capturing the IP and CD via the Application Loadbalancer with a lambda target group. This POC needs to take that effort forward by having lambda function(s) communicating to DynamoDB.

Understanding of meter and transformer operations is not relevant.

Entry Criteria

The "event" parameter of the lambda function will carry different data payload depending on IP and CD. This payload should be used for creating the test cases as well for unit testing the lambda(s). The structure of these payloads is given below -

IP payload structure & description

IP is transmitted by every device such as transformers, control and protective devices in the substation. Each device in a substation has a unique ID that will be available as part of the payload.

For this POC you can assume the following structure coming in from a transformer -

```
{
    "subid" : "s560102-u1.33x2",
    "devid" : "tns58E-xf220k",
    "devtype" : "xform",
    "pri_vol" : 33,
    "pri_vol_unit" : "kva",
    "pri_curr" :  200,
    "pri_curr-unit" : "amp",
    "sec1_vol" : 440,
    "sec1_vol_unit" : "v",
    "sec1_curr" :  25,
    "sec1_curr-unit" : "amp",
    "sec2_vol" : 220,
    "sec2_vol_unit" : "v",
    "sec2_curr" :  15,
    "sec2_curr-unit" : "amp",
    "temp" :  125,
    "tamper" : 0
}
```

CD payload structure & description

CD is transmitted by every meter in the substation. Each meter in a substation has a unique ID that will be available as part of the payload. For this POC you can assume the following structure coming in from a meter -

```
{
    "subid" : "s560102-u1.33x2",
    "devid" : "imet220sm5632a",
    "devtype" : "meter",
    "vol" : 218,
    "vol_unit" : "v",
    "curr" :  15,
    "curr-unit" : "amp",
    "units" : 15687
    "temp" :  30,
    "tamper" : 0

}
```

The BS needs to support the following query criteria -

1. Fetch the IP & CD data based on substation (subid) and optionally filter by device (devid)
2. Fetch the CD data based for a given meter (devid)

Exit criteria

The following are the goals of the POC -

1. Data model(s) for DynamoDB
2. Documenting the applicable best practices of NoSQL modeling
3. Identify possible areas for "transactional writes" if any
4. Unit test case(s) for the lambda function(s)
5. List all the AWS services used
6. Working lambda function(s) in Python

Based on the above findings NPC management team will decide if the cloud is a good for them.