Readme file for Google Cloud Billing Catalog app

Tables created

The program creates following different tables to store information for the IBM Cloud, Azure, and Digital ocean services:

IBM Cloud

For IBM cloud, following four tables were created:

1- Resource table

ID	Name	Display Name

2- Plan information table

Resource ID	Plan ID	Name	Display Name

3- Price metric table

Plan ID	Metric ID	Display Name	Unit Name	Unit	Unit quantity

4- Price table

Metric ID	Quantity tier	Price	Updated on

Digital Ocean

Following table was created for storing digital ocean droplet price info:

Slug	Memory	Vcpus	Disk	Transfer	Price monthly	Price Hourly	Updated On

Azure

Following information is stored in the azure table:

Category	Name	Effective date	Rate	Unit	Sub category	Updated on

Setup needed for Azure and Digital Ocean

A few manual steps need to be taken to make Azure pricing catalog work. These commands have to be performed on the Azure console and they yield few tokens and ids which need to be provided to the serverless app before deployment. Detail about those steps can be found on this link. Digital Ocean also requires an Auth token for pricing API to work. It can be easily generated by going into account settings of Digital Ocean account.

Additional Plugin for serverless

Although the plugin is already installed in the provided folder and only following command is required to run the app (provided all the required parameters are given in the custom section):

```
sls deploy
```

If the command throws error about a missing python plugin, following command will solve that issue:

```
sls plugin install -n serverless-python-requirements
```

This plugin reads the requirements.txt file and downloads the required libraries for handlers to run.

Write throughput issue for DynamoDB

Currently, due to testing purposes on the free tier, the write capacity units for DynamoDB in serverless.yml file are set at 10 per second. To solve the issue, a sleep (0.1) command is used before inserting a value in the dynamodb table. This will save overcharging the AWS but will also slow down the execution. If you have no issue in allowing greater write throughput, you can simply delete the sleep(0.1) command in all three handlers and then increase WriteCapacityUnits in serverless.yml from 10 to 100 in all three different lambdas.