Introduction

The Lambda has it's logic documented so it should pose no trouble to the seasoned python / aws dev. Still it's helpful to have some pointers so the next guy knows where to look for stuff.

App file

The app file shouldn't contain interaction logic. All the logic for Brightpearl and DB interaction should live in it's objects: Brightpearl and DBInteraction. Same with any AWS simplifications. the main method should only call and interact with whatever these objects return. Refer to the object and the comments inside so you can understand what's going on under the hood.

- For Brightpearl auth, check here.
- For Postgres API psycopg2, check here.

FAQs

How can I add a new schema/provider:

- Go to AWS Secrets manager
- Select "prod_brightpearl_integration"
- In the "Secret Value" field press "Retrieve Secret Value"
- Click "Edit"
- Add the following secrets:
    - brightpearl_[login_name]_login
    - brightpearl_[login_name]_api_key
    - brightpearl_[login_name]_reference
        - **NOTE:**
        - Maintaining this naming conventions is paramount. "brightpearl_loginname" drives all the decisions related to connections inside the lambda function.
- Create a schema in the database maintaining the naming convention:
- CREATE SCHEMA brightpearl_outofstep;
- Create the required tables inside this schema. Use the scripts provided inside the "sql" directory.
- Update template.yaml file to trigger passing the required schema info:
- Rebuild and deploy the function.
- (**OPTIONAL**) - you can create a test in the lambda GUI in order to test the deployment.
    - **NOTE:**
    - Keep in mind that for large customers you might need to increase the limits in the lambda so it doesn't timeout.