CGC DataOps

1. CGC Data Lake Client library

Getting Started

The CGC Data Lake Client library is a Python library that allows you to push data to the CGC data lake. To start, you need to request a tenant key from the CGC Data Lake team. The tenant key is used to authenticate the client library with the CGC data lake and to ensure that the data is pushed to the correct location in the data lake.

Prerequisites

The CGC Data Lake Client library is a Python library that allows you to push data to the CGC data lake. The library requires Python 3.8 or later.

Installation

The library is available on the AMD Artifactory. To install the library, set the PIP_INDEX_URL and PIP_TRUSTED_HOST environment variables and run the following command:

> To use the dev environment set environment variable `DATAPLATFORM_ENV=dev` or `DATAPLATFORM_ENV=prod` for production environment.`

Windows

```
C:\> set PIP_INDEX_URL=https://atlartifactory.amd.com:8443/artifactory/api/pypi/SW-CGCDATAOPS-DEV-VIRTUAL/simple
C:\> set PIP_TRUSTED_HOST=atlartifactory.amd.com
C:\> pip install cgcdataplatform
```

```
PS C:\> $env:PIP_INDEX_URL="https://atlartifactory.amd.com:8443/artifactory/api/pypi/SW-CGCDATAOPS-DEV-VIRTUAL/simple"
PS C:\> $env:PIP_TRUSTED_HOST="atlartifactory.amd.com"
PS C:\> pip install cgcdataplatform
```

On Linux/Mac

```
$ export PIP_INDEX_URL=https://atlartifactory.amd.com:8443/artifactory/api/pypi/SW-CGCDATAOPS-DEV-VIRTUAL/simple
$ export PIP_TRUSTED_HOST=atlartifactory.amd.com
$ pip install cgcdataplatform
```

Usage

Interacting with the CGC Data Platform SDK

For each method, you can pass the tenant key as an argument or set the `DATAPLATFORM_TENANT_KEY` environment variable with the tenant key. If you have the environment variable set and also pass the tenant key as an argument, the key in the argument takes precedence.

Upload a file use the send_file method.

This method uploads a file to the data lake. The method returns a dictionary with a message and the DataLakeObjectLocation object that represents the file uploaded to the data lake. The following is an example of how to use the send_file method to upload a file to the data lake.

> It is recommended that each file uploaded to the data lake has a unique name. Two files with the same name cannot be uploaded to the data lake on the same day. Files will be automatically compressed into bz2 format upon upload to the data lake. If you don't want to compress your file, set the compression parameter to None.

```
from cgcdataplatform import send_file
# tenant_key is the key provided by the CGC Data platform team
r = send_file( "path/to/file", tenant_key)
# or you can set the DATAPLATFORM_TENANT_KEY environment variable with the tenant key
r = send_file( "path/to/file")
# Don't compress your file
r = send_file( "path/to/file", compression=None)
```

Output:

```
{   "message": "File uploaded successfully",
    "datalake_location": {
        DataLakeObjectLocation (
            url='https://dvueapexetl.dfs.core.windows.net/datalake-ingress/path/to/file',
            account='dvueapexetl',
            container='datalake-ingress',
            blob_name='path/to/file',
        )
    }
}
```

Upload a directory use the send_directory method.

This method uploads all files in the top level directory to the datalake. This method uses the same parameters as the send_file method. This method returns a list of dictionaries, where each dictionary contains a message and a DataLakeObjectLocation object corresponding to the file uploaded.

> If the directory contains more than 9 files, the method will automatically convert your directory into a compressed tar.gz files. The tarfile will be named using your directory's name. Therefore, it is recommended to use a unique directory name as two files with same name cannot be uploaded in the same day.

```
# Has all the same parameters as the send_file method
from cgcdataplatform import send_directory
r = send_directory("path/to/directory", tenant_key)
# You can set the DATAPLATFORM_TENANT_KEY environment variable with the tenant key
r = send_directory("path/to/directory")
```

Output:

```
[
    {
        "message": "File uploaded successfully",
        "datalake_location": {
            DataLakeObjectLocation (
                url='https://dvueapexetl.dfs.core.windows.net/datalake-ingress/path/to/file1',
                account='dvueapexetl',
                container='datalake-ingress',
                blob_name='path/to/file1',
            )
        }
    },
    {
        "message": "File uploaded successfully",
        "datalake_location": {
            DataLakeObjectLocation (
                url='https://dvueapexetl.dfs.core.windows.net/datalake-ingress/path/to/file2',
                account='dvueapexetl',
                container='datalake-ingress',
                blob_name='path/to/file2',
            )
        }
    },

]
```

Get a file using the get_file_stream method.

This method returns a file stream that can be used to read the file. The send file method returns a dictionary with DataLakeObjectLocation in “datalake_location”. Pass that location to the get_file_stream method to get the file stream. The following is an example of how to use the get_file_stream method to get a file stream and write the file to disk.

```
from cgcdataplatform import send_file, get_file_stream
r = send_file( "path/to/file")

stream = get_file_stream(r["datalake_location"], tenant_key)
# or
stream = get_file_stream(r["datalake_location"])

with open("filename.txt", "wb") as f:
    f.write(stream.read())
```

List files in a directory using the list_files method.

This method returns a list of DataLakeObjectLocation objects that represent the files uploaded to the data lake by a tenant. You can pass a partition name to list files in a specific partition. The following is an example of how to use the list_files method to list files uploaded by a tenant. To

get the file stream, you can pass the DataLakeObjectLocation object to the get_file_stream method.

```
from cgcdataplatform import list_files
r = list_files(tenant_key=tenant_key)
# or you can set the DATAPLATFORM_TENANT_KEY environment variable with the tenant key
r = list_files()
# or you could specify the partition the file was uploded to
r = list_files(partition="partition_name")
```

Output:

```
[
    DataLakeObjectLocation (
        url='https://dvueapexetl.dfs.core.windows.net/datalake-ingress/path/to/file',
        account='dvueapexetl',
        container='datalake-ingress',
        blob_name='path/to/file',
    ),
]
```

Interacting with the CGC Data Platform CLI

To upload using the CLI, use the following command:

```
$ cgcdataplatform upload tenant_key path/to/file
#or
$ cgcdataplatform upload path/to/file -v
```

Output:

```
Uploaded file to data lake: {'message': 'File uploaded successfully', 'datalake_location':
DataLakeObjectLocation(url='https://dvueapexetl.dfs.core.windows.net/datalake-
ingress/path/to/file', account='dvueapexetl', container='datalake-ingress',
blob_name='datalake/path/to/blob')}
```

To list files uploaded by a tenant, use the following command:

```
$ cgcdataplatform list
```

Output:

```
datalake/path/to/file1
datalake/path/to/file2
datalake/path/to/file3
```

ARP functionality

Although conceptually separate, the CGC dataplatform also supports limited APEX-related functionality to interact with the APEX Results Pipeline (ARP).

Client applications can be built on top of the `apex` module, while the CLI also supports some ARP-related commands:

```
$ cgcdataplatform arp --help
```

Client SDK

- cgcdataplatform
- apex
- Authentication