gdrive2

Release 0.7.0

Preetham Paul

Apr 20, 2021

CONTENTS

CHAPTER

ONE

GITHUB

The code is available here

CHAPTER

TWO

WHAT IS GDRIVE2?

gdrive2 helps users to easily access files from Google Drive using paths, instead of File ids. Google API requires the users to know the File ID to access it, but this package which is built using the pyDrive package allows users to use a *file path* to access the file or folder.

In addition to this, the package also provides commands that can be called from a terminal, like `cd`, `ls`, `pull` or `push`, to quickly view, modify, download or upload files from or to Google drive using a python code or just the command line. pyDrive package is used to build this.

CHAPTER

THREE

GETTING STARTED

3.1 Setup

Use the following command to install the package. (The package is still under testing)

```
pip install gdrive2
```

3.2 Requirements

A client secrets file is required for authentication. The instructions to create a client secrets file are provided in pydrive Quickstart guide, but are shown here again:

To use Google Drive API, an application has to be created using the Google Cloud Console. To obtain a client secrets file, create an application as shown below.

1. Open the console and create a new project.
2. After assigning name and organisation to the project, go to **APIs & Services –> Dashboard**
3. Go to **OAuth consent screen** and configure a consent screen. Assign an application name and other details.
4. Go to **Credentials –> Create Credentials –> OAuth client ID**
5. For Application type, use **Web Application** and assign a name to it.
6. **For Authorized java script origins, use** `http://localhost`

   `http://localhost:8080`

   For Authorized redirect URIs, use: `http://localhost:8080/`

   `http://localhost:8090/`
7. Click "Create"

Once credentials are created, they can be downloaded as a **.json** file with the name **client_secrets<some_long_id>.json**. This file can be renamed to some simple name and save on the local computer.

Note: If a client secrets file is used to authenticate a google account, the registered application or project is given permission by the google user to access data. Anyone with this .json file will be able to able to access those accounts with the same permissions set previously, however only if after authenticating or if the authentication credentials are saved previously.

3.3 Quickstart

gdrive2 can be used as a python package right away, but try the following steps to be able to use gdrive2's functions directly from terminal. First, import the gdrive2 in a python console.:

```
$ python
>> import gdrive2
>> gdrive2.ROOT_PATH
```

ROOT_PATH is the local system path to the gdrive2 package. Add this path to the **PATH** environment variable. Now, *gd* can be used as a command in command prompt or bash shell.

3.4 Quickdemo 1 : Basics

Lets see how gdrive2 can be used from command line.

Open terminal and set some folder where you intend to download or from where you intend to upload files.

```
gd init
```

When asked for a username, this is not same as the google username. More about this explained in the *username* documentation. Enter some nickname (for example, *mygdrive2*) you would like to give to your google account, so that you can use this for quick authentication into your account in future. This should take you to a Oauth Consent screen, where you'll be asked to enter your Google username and password. You are good, if you see this a html page showing this:

```
The authentication is successful.
```

Once authentication is done, from the current working folder, you can try several gdrive2 commands. Just like *git*, gdrive2 also creates a hidden folder **.gd** which contains information about the fileIDs, driveIDs etc. More about this explained in the *parents* documentation.

Warning: After authentication of a new account, the credentials are stored in **ROOT_PATH/api_data** folder. The contents of this folder must be handled with discretion.

After, initialization and authentication, you can use all the gdrive2 commands from this directory.

```
gd status
```

```
---------------
Parent dicts :
---------------
// origin // <DEFAULT>
username : mygdrive2
path     : ''
id       : 'root'
drive    : 'My Drive'
driveId  : 'root'
client_sec : 'client_secrets'

No files/folders staged.
```

This command shows the contents of **.gd/.gdinfo.json** file created during initialization. The structure of this and the meaning of each term are explained in the *parents* documentation. For example, the folders in my drive are in this hierarchy:

```
My Drive
|
|___fruits
|       |___hard
|       |      |__apple.jpg
|       |      |__guava.png
|       |
|       |___soft
|              |__grapes.jpg
|
|___flowers
      |___yellow
      |      |__sunflower.tiff
      |
      |___red
           |__roses.jpg
           |__poppy.png
```

To list my files in the path stored in the **path** variable of the dictionary showed above:

```
gd ls
```

```
: D[2351.5kB] 1 : fruits
: D[6571.2kB] 2 : flowers
```

My Drive is the name of the google drive's root folder. **''** (empty string) denotes the path to it. This shows that there are two folders in my root folder - fruits and flowers.

```
gd ls fruits/hard
```

```
: f[1000.5kB] 1 : apple.jpg
: F[321.0kB] 2 : guava.jpg
```

This is how the paths can be simply used to get the file contents. Now, I'll change my path from '' to *'fruits/hard'*:

```
$ gd cd fruits/hard
origin cwd changed to 'fruits/hard'

$ gd ls
: f[1000.5kB] 1 : apple.jpg
: F[321.0kB] 2 : guava.jpg

$ gd status
---------------
Parent dicts :
---------------
// origin // <DEFAULT>
username : mygdrive2
path     : 'fruits/hard'
id       : '34eWf..iT23'
drive    : 'My Drive'
driveId  : 'root'
client_sec : 'client_secrets'
```

(continues on next page)

(continued from previous page)

```
No files/folders staged.
```

If I want to download the folder:

```
gd pull
```

Or, to just download the *apple.jpg*:

```
gd pull origin apple.jpg
```

If I want to upload *berry.png* to **fruits/hard**:

```
$ gd add <local_path_to_'berry.png'>
$ gd cd fruits/hard
$ gd push
```

3.5 Quickdemo 2 : Multiple Parent functionality

.gd/.gdinfo.json is a dictionary with each key defined as a parent (just like a remote in git). The first key of this dictionary is 'default_parent' whose value is the name of a default parent. Multiple parents can be set with multiple usernames(i.e. google accounts), paths, shared drives or even different client secrets files and one of these can be given the status of 'default_parent'. This makes frequent uploading and downloading as easy as git pull and push functions.

To add a new parent:

```
git init -add
```

Once a parent is added (say, origin2), we can assign different parameters (like username, path etc.) and push files simultaneously

```
git push origin origin2
```

3.6 Quickdemo 3 : Use of these commands in python script:

All these commands can be used in a python script as shown below. The only difference from the terminal commands is that apart from the main function (init, status, etc), the optional arguements must be passed as strings in a list. If there is no arguements, an empty list must be passed in the function.

```
>> import gdrive2 as gd
>>
>> gd.init([])
>> gd.status([])
>> gd.add(['berry.jpg', 'mango.jpg'])
```

3.7 Quickdemo 4 : Query search of filenames in drive

A query can be used to search for files in the parent path or a user-specified path using gdrive2.find() function.

The example command here searches for all png files that start with g.

```
$ gd find "g* and *.png*"

file_id : file_name
--------------------------

34efd...rtW : guava.png
```

For more information, see gdrive2.find() function and the Query for gdrive2.find() section below it.

CHAPTER

FOUR

DATA STRUCTURING

4.1 Parents

For convenient pushing and pulling files from various folders and google drive accounts with multiple api clients, a **parent** structure is used. When a working directory is **initialized**, multiple **parents** can be used to store paths and ids for multiple google drives. Each **parent** is a list of other data like username, parent_path, parent_id, drive_name, drive_id and client_name stored as strings. All the parents are stored as dictionary items as shown below with the *parent_name* being the name or key of the parent.

```
parent_name : [
      username,
      parent_path,
      parent_id,
      drive_name,
      drive_id,
      client_name
]
```

init function is used to initialize a directory which creates a **.gd** folder in it with the following contents.

```
|__.gd
    |__.gdinfo.json
    |__.gdstage
```

Each **.gdinfo.json** is a dictionary of **parents**, defined by user for that directory. For example, the user can initialize a directory and define **parents** as shown below, which are saved in the **.gd/.gdinfo.json**.

```
$ cat .gd/.gdinfo.json

{default_parent : 'origin',
'origin' : [
      'preetham_acc',
      'fruits',
      '1bW2H.....uY4on',
      'Shared drive 1',
      'nZTvt.....4kt9',
      'client_secrets'
],
'test' : [
      'paul_acc',
      'flowers/yellow',
      'WEpIZu.....HE3L7B',
      'My Drive',
```

(continues on next page)

(continued from previous page)

```
      'root',
      'client_secrets'
]}
```

The **default_parent** has a string as its value, which is one of the parent_names.

4.2 username

The **usernames** used in parents are different from the google usernames (like in <google_username>@gmail.com). Each **username** corresponds to a single <google_username>, but each <google_username> can have multiple **usernames**. A **username** is basically a nickname the user gives to their google account for easy pushing and pulling files. Once, a **username** is defined, the **username** can be used for other parents in different directories to use the same google account.

4.3 parent_path and parent_id

The **parent_path** is the path to a folder in drive in the google account registered with the nickname **username**. For example, assume a drive as shown below. The file hierarchy has several tiers as shown below.

```
My Drive                                  - - - - - - - - tier 1
|
|____fruits                               - - - - - - - - tier 2
|       |____hard                         - - - - - - - - tier 3
|       |      |__apple.jpg               - - - - - - - - tier 4
|       |      |__guava.png               - - - - - - - - tier 4
|       |
|       |____soft                         - - - - - - - - tier 3
|              |__grapes.jpg              - - - - - - - - tier 4
|
|____flowers                              - - - - - - - - tier 2
      |____yellow                         - - - - - - - - tier 3
      |       |__sunflower.tiff           - - - - - - - - tier 4
      |
      |____red                            - - - - - - - - tier 3
           |__roses.jpg                   - - - - - - - - tier 4
           |__poppy.png                   - - - - - - - - tier 4
```

The path to the **sunflower.tiff** folder would be:

```
flowers/yellow/sunflower.tiff
```

Note that the root folder here is **My Drive**, which will not be included in the path. The same path representation is followed for the folders in shared drives.

Each file or folder in google drive has an **ID**. See this to know more about file metadata in google drive. A **parent_id** is the ID of the folder at the **parent_path**.

4.4 drive_name and drive_id

This is the name of the drive in which the **parent_path** is located. Each google account can have multiple shared drives along with the main **"My Drive"**. **drive_id** is the ID of this drive.

4.5 client_name

A client secrets (app credentials) file can be created on Google Cloud Console as shown in Requirements. Once credentials are created, they can be downloaded as a **.json** file with the name **client_secrets<some_long_id>.json**. This file can be renamed as **<client_name>**.json and can be used to access parent_path. If gdrive2 is being used for the time, it asks the user to show the file location of a client secrets file and creates a default file - **client_secrets.json**. Later, more such files can be added with different **client_names** as <client_name1>.json, <someother_user_specified_name>.json etc. Each parent can be assigned a **client_name** different from the default name - **client_secrets**.

CHAPTER

FIVE

FUNCTIONS

- genindex
- search

CHAPTER

SIX

UPDATE/UNINSTALL GDRIVE2

For updating gdrive2:

```
pip install gdrive2 -U
```

For uninstalling gdrive2:

```
pip uninstall gdrive2
```

To remove the authentication data saved, use this before uninstalling gdrive2:

```
gd default
```

CHAPTER SEVEN

CONDITIONS FOR USING THE PACKAGE

1. The file path to the grapes.jpg in the example shown above must be passed as shown below:

```
fruits/soft/grapes.jpg
```

Note that the root folder (which is **My Drive** in this case), must not be passed in the path. The information of the root folder is included in the **drive_id** of the parent.

2. As much as possible, try to avoid having more than one folder with the same name in the same folder. Some functions like gdrive2.gd.find() might work, but others may not. Most functions throw an error when they find more than one file or folder with same name in the same folder.