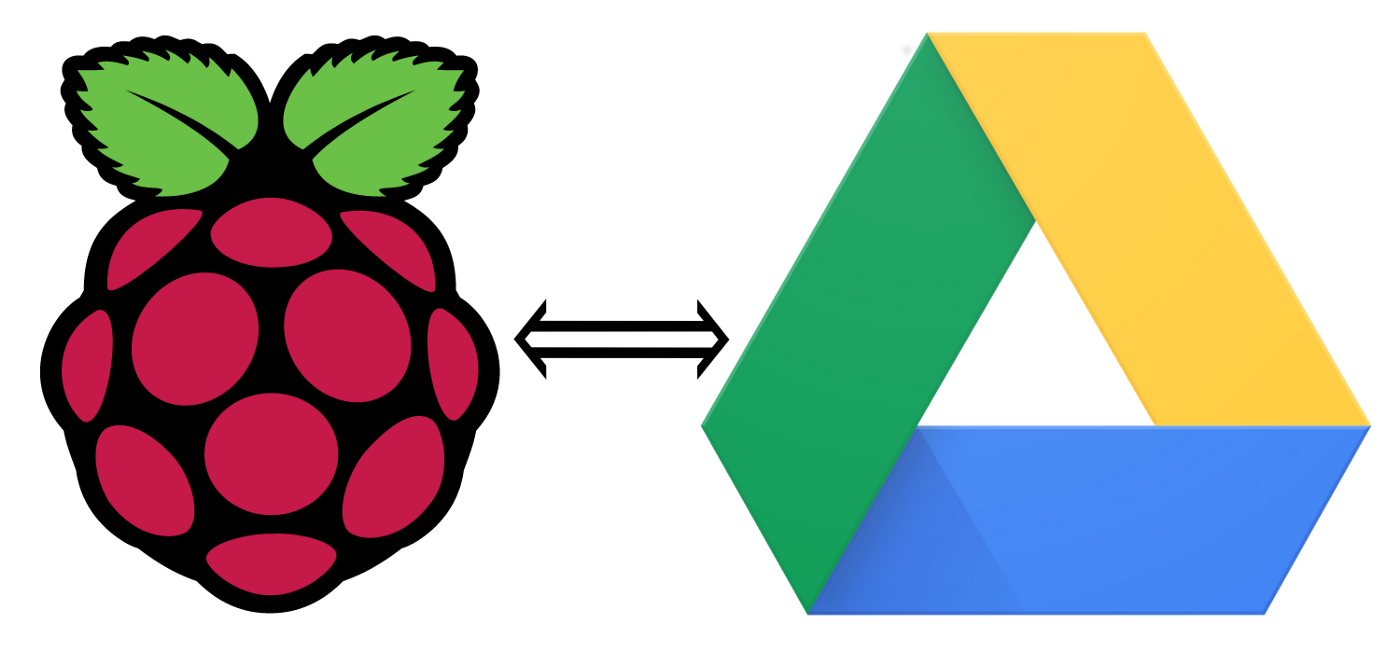
https://medium.com/@artur.klauser/mounting-google-drive-on-raspberry-pi-f5002c7095c2

**Mounting Google Drive on Raspberry Pi**

[Artur Klauser](https://medium.com/@artur.klauser?source=post_page-----f5002c7095c2--------------------------------)

[Artur Klauser](https://medium.com/@artur.klauser?source=post_page-----f5002c7095c2--------------------------------)

[May 13, 2019·10 min read](https://medium.com/@artur.klauser/mounting-google-drive-on-raspberry-pi-f5002c7095c2?source=post_page-----f5002c7095c2--------------------------------)



The [Raspberry Pi](https://www.raspberrypi.org/) is a great little device to run a variety of services in your home network. It saves files on a user-provided SD card or USB stick. But what if you want to store files off-device to safeguard the work that you’ve put in to set up whatever services you’re running on your Pi? What if you would like to share files between multiple Pis? Or what if you have files like your photos living in the cloud already and you’d like to easily access them from your Pi? This article shows how you can connect your Google Drive to your Raspberry Pi for all the above use cases and more.

Which Software to Use?

The first thing we have to decide is which software to use. When you look around for software to connect Google Drive to a Linux environment you run into projects with a wide range of different features and requirements. Some are better maintained than others. The one I’ve chosen here is [rclone](https://rclone.org/), which is actively maintained and provides a lot of functionality. In particular, with the help of the [FUSE](https://github.com/libfuse/libfuse) userspace filesystem layer, it allows you to mount Google Drive as part of your Linux file system, which greatly simplifies using the cloud storage from your Pi. These instructions will probably also work on most other Debian Linux based distributions with little or no change.

Rclone is much more versatile than the limited usage described in this article. It doesn’t only support Google Drive but a whole host of other cloud storage providers as well, such as Amazon Drive, Dropbox, or Microsoft OneDrive to name just a few of the larger ones. Besides allowing you to mount one of those cloud storage services into your file system, it also provides a set of CLI commands to directly manipulate files on those services. In this article, however, we won’t use them and concentrate on using rclone mount.

The installation instructions below assume that you are logged in to your Raspberry Pi and execute all instructions there, unless otherwise noted.

Installing Rclone on Raspbian

The easiest way to install rclone is to use the pre-compiled Debian package:

sudo apt-get install rclone

As of this writing, the current release of Raspbian is based on Debian v.9 (Stretch), and the version of rclone we get via apt-get is 1.35. However, rclone has progressed to version 1.47 in the meantime. To get a more up-to-date version, instead of using the apt package that comes with the default Raspbian package repositories, we’re going to install an up-to-date rclone package from its [download](https://rclone.org/downloads/) page:

cd /tmp  
wget <https://downloads.rclone.org/v1.47.0/rclone-v1.47.0-linux-arm.deb>  
sudo apt install ./rclone-v1.47.0-linux-arm.deb  
rm rclone-v1.47.0-linux-arm.deb

Configuring Rclone

Since the remote files we’re going to access are private to a particular user, rclone keeps configuration files separate per user. All secret access tokens necessary to interact with the cloud storage are stored in a user’s config file such that different users can’t access each other’s cloud storage. By default the config file is located in $HOME/.config/rclone/rclone.conf and should have permissions of 0600 (read-write only for file owner) to make sure your access tokens remain private.

For Google Drive there is another consideration. If you go with the default configuration, you’re using the *client\_id* that corresponds to the rclone application. Google imposes per-client rate limits on interactions with Google Drive. This means that by using the rclone default *client\_id* you’re competing against all other users of rclone, globally, for operation bandwidth to Google Drive. Even though the rclone authors try to mitigate this by asking Google for higher quotas, using the default is not an ideal situation. For this reason we’ll set up our own *client\_id* with Google and avoid the stampede.

Getting a Google Drive Client ID

### Making your own client\_id

When you use rclone with Google drive in its default configuration you are using rclone's client\_id. This is shared between all the rclone users. There is a global rate limit on the number of queries per second that each client\_id can do set by Google. rclone already has a high quota and I will continue to make sure it is high enough by contacting Google.

It is strongly recommended to use your own client ID as the default rclone ID is heavily used. If you have multiple services running, it is recommended to use an API key for each service. The default Google quota is 10 transactions per second so it is recommended to stay under that number as if you use more than that, it will cause rclone to rate limit and make things slower.

Here is how to create your own Google Drive client ID for rclone:

1. Log into the [Google API Console](https://console.developers.google.com/) with your Google account. It doesn't matter what Google account you use. (It need not be the same account as the Google Drive you want to access)
2. Select a project or create a new project.
3. Under "ENABLE APIS AND SERVICES" search for "Drive", and enable the "Google Drive API".
4. Click "Credentials" in the left-side panel (not "Create credentials", which opens the wizard), then "Create credentials"
5. If you already configured an "Oauth Consent Screen", then skip to the next step; if not, click on "CONFIGURE CONSENT SCREEN" button (near the top right corner of the right panel), then select "External" and click on "CREATE"; on the next screen, enter an "Application name" ("rclone" is OK) then click on "Save" (all other data is optional). Click again on "Credentials" on the left panel to go back to the "Credentials" screen.

(PS: if you are a GSuite user, you could also select "Internal" instead of "External" above, but this has not been tested/documented so far).

1. Click on the "+ CREATE CREDENTIALS" button at the top of the screen, then select "OAuth client ID".
2. Choose an application type of "Desktop app" if you using a Google account or "Other" if you using a GSuite account and click "Create". (the default name is fine)
3. It will show you a client ID and client secret. Make a note of these.
4. Go to "Oauth consent screen" and press "Publish App"
5. Provide the noted client ID and client secret to rclone.

Be aware that, due to the "enhanced security" recently introduced by Google, you are theoretically expected to "submit your app for verification" and then wait a few weeks(!) for their response; in practice, you can go right ahead and use the client ID and client secret with rclone, the only issue will be a very scary confirmation screen shown when you connect via your browser for rclone to be able to get its token-id (but as this only happens during the remote configuration, it's not such a big deal).

(Thanks to @balazer on github for these instructions.)

Sometimes, creation of an OAuth consent in Google API Console fails due to an error message “The request failed because changes to one of the field of the resource is not supported”. As a convenient workaround, the necessary Google Drive API key can be created on the [Python Quickstart](https://developers.google.com/drive/api/v3/quickstart/python) page. Just push the Enable the Drive API button to receive the Client ID and Secret. Note that it will automatically create a new project in the API Console.

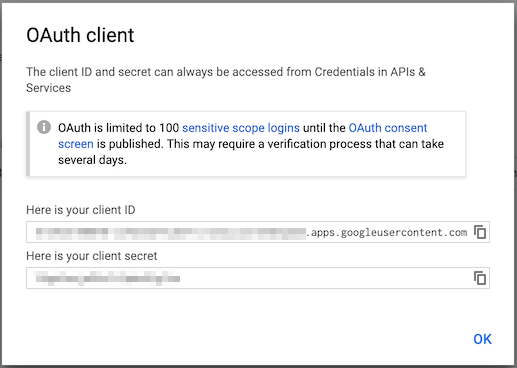
In a browser go to the [Google Developer Console](https://console.developers.google.com/) and log in with your Google account. You don’t have to do this on your Raspberry Pi. Any computer is fine.

* On the top go to **Select a project → New Project**
* — — Fill in a Project Name: ***my-rclone-gdrive-access***
* — — Hit **Create**
* Just below the top select **+ ENABLE APIS AND SERVICES**
* — — Search for **Drive**
* — — Select **Google Drive API**
* — — Hit **Enable**
* On the left side-panel select **Credentials**
* — — In the main panel you’ll get a warning about

Remember to configure the OAuth consent screen with information about your application.

* — — Next to the warning select **CONFIGURE CONSENT SCREEN**
* — — Fill in the Application Name: ***my-rclone***
* — — Hit **Add Scope**
* — — — — Pick **../auth/drive** (which can “see, edit, create, and delete all of your Google Drive files”) and hit **Add**
* — — Hit **Save**
* Go back to **Credentials** on the left side-panel and then **Credentials** on the top menu inside the main panel
* — — Select **Create credentials → OAuth client ID**
* — — Select Application type: **Other** and set some sensible name: ***my rclone client***
* — — Hit **Create**
* You’ll get a confirmation page containing the *client ID* and *client secret* which looks like this.





Copy the *client ID* and *client secret* since we’ll need it for the rclone configuration.

Setting up the Rclone Configuration

Back on the Raspberry Pi log in as the user who should be able to access the Google Drive files and run:

rclone config

Select the following answers:

No remotes found - make a new one  
n) New remote  
s) Set configuration password  
q) Quit config  
n/s/q> **n**  
name> **gdrive**Type of storage to configure.  
Enter a string value. Press Enter for the default ("").  
Choose a number from below, or type in your own value  
*... snip ...*  
12 / Google Drive  
 \ "drive"  
*... snip ...*  
Storage> **12**  
See help for drive backend at <https://rclone.org/drive/>Google Application Client Id  
Setting your own is recommended.  
See <https://rclone.org/drive/#making-your-own-client-id> for how to create your own.  
If you leave this blank, it will use an internal key which is low performance.  
Enter a string value. Press Enter for the default ("").  
client\_id> **425159802070-tup27t8cv4h2z3cjkqfmft4n8gju76lf.apps.googleusercontent.com**

This is the *client ID* that you saved in the previous step.

Google Application Client Secret  
Setting your own is recommended.  
Enter a string value. Press Enter for the default ("").  
client\_secret> **MspFTowlsy0SUSRm-1RvcTyx**

This is the *client secret* that you saved in the previous step.

Scope that rclone should use when requesting access from drive.  
Enter a string value. Press Enter for the default ("").  
Choose a number from below, or type in your own value  
1 / Full access all files, excluding Application Data Folder.  
 \ "drive"  
*... snip ...*  
scope> **1**ID of the root folder  
Leave blank normally.  
Fill in to access "Computers" folders. (see docs).  
Enter a string value. Press Enter for the default ("").  
root\_folder\_id>Service Account Credentials JSON file path  
Leave blank normally.  
Needed only if you want use SA instead of interactive login.  
Enter a string value. Press Enter for the default ("").  
service\_account\_file>Edit advanced config? (y/n)  
y) Yes  
n) No  
y/n> **n**Remote config  
Use auto config?  
 \* Say Y if not sure  
 \* Say N if you are working on a remote or headless machine  
y) Yes  
n) No  
y/n> **n**If your browser doesn't open automatically go to the following link: <https://accounts.google.com/o/oauth2/auth?access_type=offline&client_id=425159802070-tup27t8cv4h2z3cjkqfmft4n8gju76lf.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&response_type=code&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive&state=18109698827ea077947354ca9dab0b80>  
Log in and authorize rclone for access

Copy the URL that gets printed on your screen to a new browser window. You can do this on any computer, it doesn’t have to be your Pi. In the page flow that follows select your user account (if required) and confirm that you want to give your application access to your Google Drive. Copy the verification code from the confirmation page and fill it in below.

Enter verification code> **4/PzWg7ZOEvXIoa9tI7OMjerPq\_X5UpsES5ayNvtG8Gf9TK30FERTVrCM**Configure this as a team drive?  
y) Yes  
n) No  
y/n> **n**--------------------  
[gdrive]  
type = drive  
client\_id = 425159802070-tup27t8cv4h2z3cjkqfmft4n8gju76lf.apps.googleusercontent.com  
client\_secret = MspFTowlsy0SUSRm-1RvcTyx  
scope = drive  
token = {"access\_token":"xa14.Gv7qcPS5TvrumTpNplGxg5Ndct1gd6yica0d80iFjbKXJ3KpgneZLvn-Tg-YgLPXcEesHubbBcyyeEjGGb9mP5GhwLnEDxaPhPS3fj6ddTcmMUUOdjOhYW-rb1xY","token\_type":"Bearer","refresh\_token":"1/8jxfWbRseQmqrb5aebwK9zfwZRH0IjZhv-9lmtqsVC8",  
"expiry":"2020-01-23T16:21:03.502362868-07:00"}  
--------------------

This last section shown above is a copy of the information that will be put into the $HOME/.config/rclone/rclone.conf config file. If everything looks correct we’re going to accept it and exit the configuration menu.

y) Yes this is OK  
e) Edit this remote  
d) Delete this remote  
y/e/d> **y**Current remotes:  
Name Type  
==== ====  
gdrive drivee) Edit existing remote  
n) New remote  
d) Delete remote  
r) Rename remote  
c) Copy remote  
s) Set configuration password  
q) Quit config  
e/n/d/r/c/s/q> **q**

You can now test that you have set up the configuration correctly with:

rclone ls --max-depth 1 gdrive:

This rclone command should show you all regular files in the top-level directory of your Google Drive. The next step is to test mounting Google Drive into your file system:

cd ~  
mkdir -p mnt/gdrive  
rclone mount gdrive: $HOME/mnt/gdrive

In a second terminal on your Raspberry Pi run:

ls -l ~/mnt/gdrive

This should again list all files in the top-level directory of your Google Drive. Since your cloud storage is now part of the regular file system, any program on you Pi can access it just like any other file. After this test stop the above rclone command in the first terminal with Ctrl-C.

Automatically Mounting Google Drive

The setup above is all that is necessary to use the rclone mount command and mount Google Drive by hand into your file system whenever you need it. However, that is a bit cumbersome. It would be better if Google drive were mounted automatically whenever the corresponding user logs in. This can be achieved with [systemd](https://github.com/systemd/systemd), a daemon that starts and stops services when needed. Systemd operates in one of two modes, a *— system* mode that handles machine-wide services such as bringing up networking, and a *— user* mode that deals with per-user services such as starting the desktop environment and in our case we want it to mount the rclone FUSE file system for Google Drive. A good place to learn more is the archlinux wiki for [per-user systemd](https://wiki.archlinux.org/index.php/Systemd/User).

Configuring Systemd

To tell systemd what to do it depends on configuration files in some standard locations. One of these locations where systemd looks for *user* config files is $HOME/.config/systemd/user. To have the rclone FUSE file system mounted automatically at login we’ll create a ~/.config/systemd/user/rclone@.service file with the following contents:

[Unit]  
Description=rclone: Remote FUSE filesystem for cloud storage config %i  
Documentation=man:rclone(1)[Service]  
Type=notify  
ExecStartPre=/bin/mkdir -p %h/mnt/%i  
ExecStart= \  
 /usr/bin/rclone mount \  
 --fast-list \  
 --vfs-cache-mode writes \  
 --vfs-cache-max-size 100M \  
 %i: %h/mnt/%i[Install]  
WantedBy=default.target

*Note: If you’re using rclone version < 1.46, remove the--vfs-cache-max-size line, as that flag is not yet supported in earlier versions.*

Here the special @ syntax in the config filename defines a service template which allows to write a single config file that can be used for multiple instantiations. Even though we wouldn’t really need a service template for our simple use case, we use it so the configuration already supports multiple mounts, e.g. if you also want to use it for other cloud storage providers as well. When using the systemd service name, which derives from the file name, you just append an *instance name* to @ and systemd will replace all occurrences of *%i* in the config file with that *instance name*. In our case the instance name represents the rclone *remote* name which we have configured in the rclone.conf file above, i.e. *gdrive* in our case.

On the rclone mount command line above we use a particular set of mount options that are supported for Google Drive. A full list of options can be found in the [rclone Google Drive documentation](https://rclone.org/drive/" \l "standard-options). More details on general mount options can be found in the [rclone mount documentation](https://rclone.org/commands/rclone_mount/).

Once we’ve got the config file in place we need to let systemd know about the new configuration for our *gdrive* rclone remote:

systemctl --user enable rclone@gdrive

This should install a symbolic link rclone@gdrive.service -> $HOME/.config/systemd/user/rclone@.service in ~/.config/systemd/user/default.target.wants/.

Now that systemd is aware of the new service, the next step is to turn it on:

systemctl --user start rclone@gdrive

Once turned on, when you list the mount directory

ls -l ~/mnt/gdrive

you should see the contents of your Google Drive directory again.

At this point what we have achieved works great for interactive user sessions. Every time you log in to your Raspberry Pi as the user for which you have set up Google Drive, it is mounted automatically into the file system which gives you access to the contents of your cloud storage.

Starting User-Systemd at Boot Time

For some use cases, mounting at user login time is still not quite enough. For example on my system I run an [RStudio Server](https://www.rstudio.com/), which is a web IDE for the R programming language. The RStudio Server web interface first presents a login page to authenticate the user. Once it has checked the user’s credentials it starts an rsession for them. However, starting an rsession with the UID of the authenticated user doesn’t constitute a *login* or *session* for that user. Thus no systemd user session is started and the actions that would mount the cloud storage are not performed, so the RStudio IDE doesn’t have access to the cloud storage directory of the user.

There is a solution to this problem though. We can instruct systemd to [start a user session at boot time](https://wiki.archlinux.org/index.php/Systemd/User#Automatic_start-up_of_systemd_user_instances) instead of login time with the following command:

loginctl enable-linger $USER

Now, whenever our Raspberry Pi reboots, a systemd user session is started immediately for that user, mounting the configured Google Drive and enabling use cases like RStudio Server mentioned above.

If you’ve followed the setup to here, congratulations! You’ve got your Raspberry Pi connected to your Google Drive and can easily access files from your cloud storage from any program running on your Pi just by accessing files under $HOME/mnt/gdrive.

*2019–05–13 —*[*Eddies in Entropy*](https://nexus.eddiesinentropy.net/)