Uploading files to Google Drive directly from the Terminal (using Curl)

An alternative method of pushing data from a computer to the cloud.

Daniel Ellis Sep 14, 2020 · 4 min read ★

In many cases, it may be difficult to send data from a new machine to another. Examples of this include HPC facilities which are hidden behind a login portal but do not allow ssh tunnelling or simple headless machines which only have a few core programs installed. In this article, we look at using cURL (a command-line program for transferring data) to push a zipped file (containing log files) onto our google drive account for further analysis.

Photo by Element5 Digital on Unsplash

Installation

Most machines will come with cURL installed (try typing `which curl`). If this is not the case we can install it with

```
sudo apt install curl # Linux Debian/Ubuntu
```

or

```
brew install curl     # Mac
```

Now we have it installed, we can look at creating the credentials needed to send files.

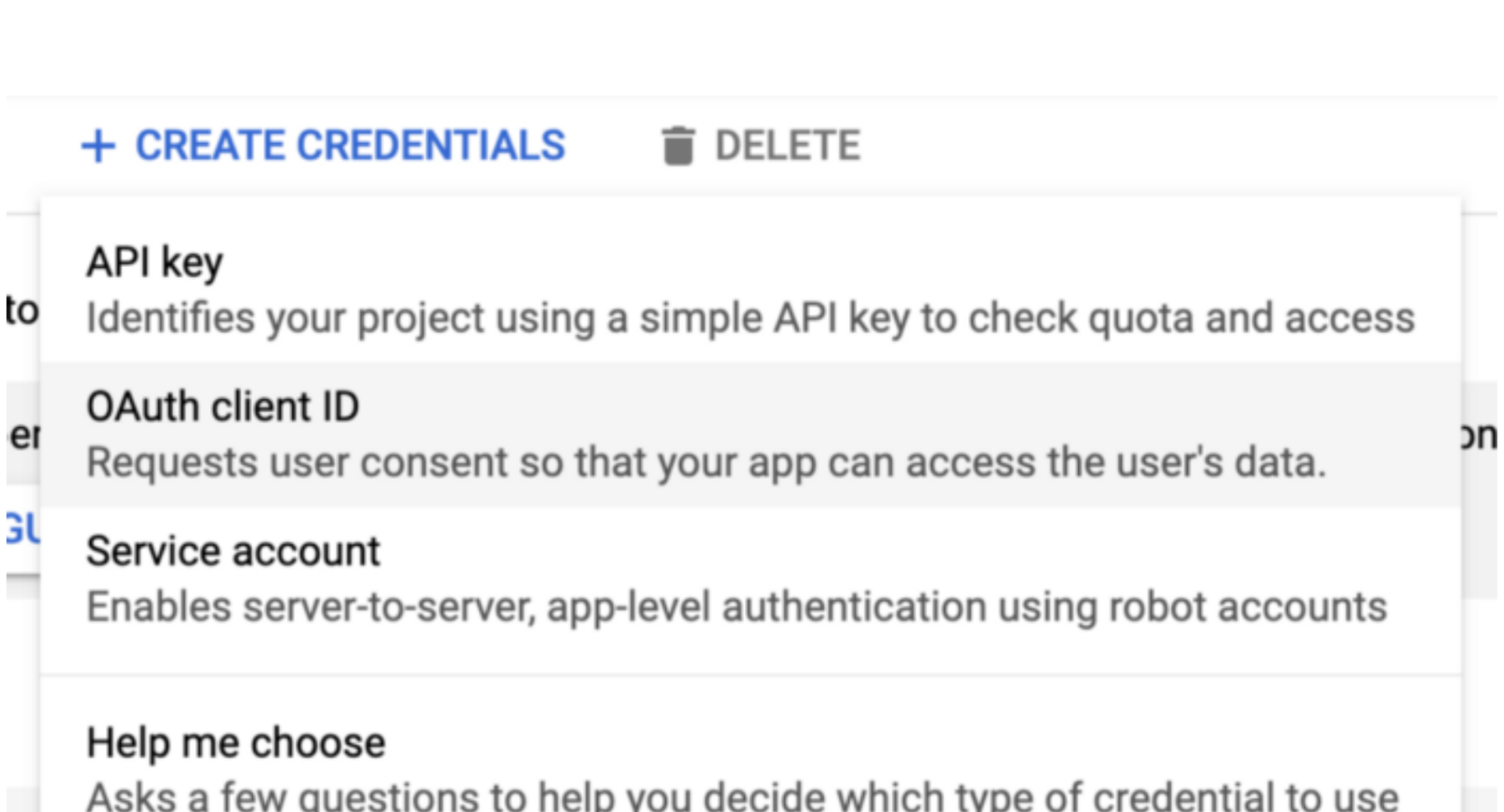
· · ·

Create your project credentials

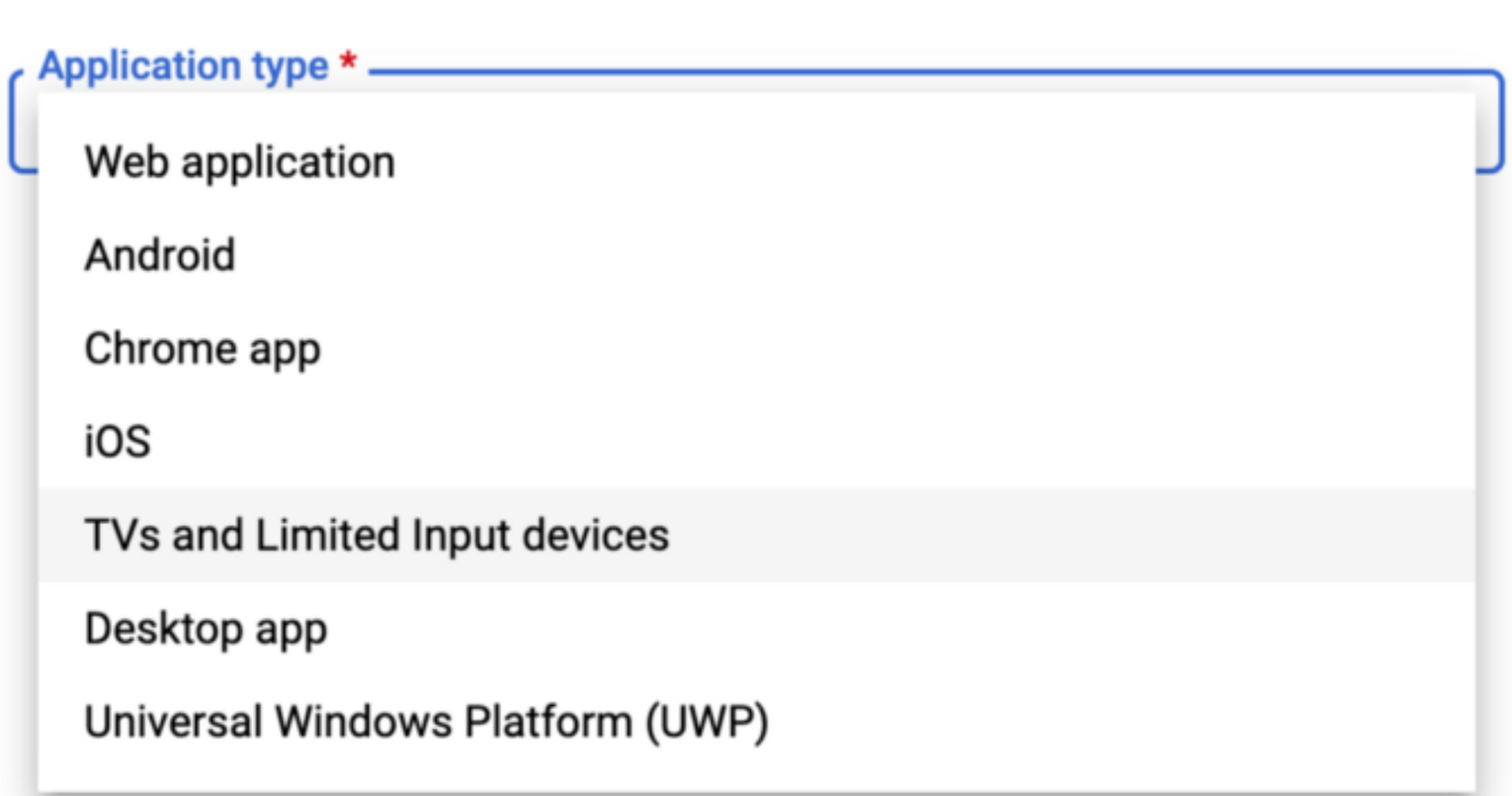
As we are allowing access to our google drive, we want to be able to manage this. This is done by creating a project with user-defined permissions to act as the proxy between our users (in this case us on a different machine) and our account. We start by going to the following page (link below) and creating a new project.

Google Cloud Platform
Google Cloud Platform lets you build, deploy, and scale applications, websites, and services on the same infrastructure...
console.developers.google.com

After this has been done we select the Credentials tab (on the left) and "create credentials" from the top.



When asked for the app type, we select TV and other.



Finally, this generates a `client id` and a `client seacret` .

This is your username and password so copy it somewhere secure.

· · ·

Now we need to verify the device

To do this we `ssh` into the machine we wish to upload from and run the following command:

```
curl -d "client_id=
<client_id>&scope=https://www.googleapis.com/auth/drive.file"
https://oauth2.googleapis.com/device/code
```

Doing so we get a response in the following format

```
{"device_code": "<long string>",
"user_code": "xxx-xxx-xxx",
"expires_in": 1800,
"interval": 5,
"verification_url": "https://www.google.com/device"}
```

Here we need to visit the URL (https://www.google.com/device) and provide the user code to complete our verification. We now continue to select our google account and grant the relevant permissions. When doing this make sure to note down the **device code** for the next step.

· · ·

Get Bearer code

When we start uploading, this is the code we shall need to use to identify our account. We get it by using the following:

```
curl -d client_id=<client id> -d client_secret=<client secret> -d
device_code=<device code> -d
grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Adevice_code
https://accounts.google.com/o/oauth2/token
```

The client id and secret are saved from the *first step*, and the device code in the *previous section*. The output should be in the format:

```
{
"access_token": ".....",
"expires_in": 3599,
"refresh_token": "....",
"scope": "https://www.googleapis.com/auth/drive.file",
"token_type": "Bearer"
}
```

Write down the **access_token** as it will be needed in the upload stage.

· · ·

Upload files

The command we use to upload files is given below

```
curl -X POST -L \
    -H "Authorization: Bearer <enter access token here>" \
    -F "metadata={name
:'<our.zip>'};type=application/json;charset=UTF-8" \
    -F "file=@<our.zip>;type=application/zip" \
    "https://www.googleapis.com/upload/drive/v3/files?
uploadType=multipart"
```

Here you may need to enable the app API before being allowed to upload data. The link to do this is given in the error message if this is the case.

Here *multipart* files are expected to only be a couple of MB in size. However if you are looking at moving larger files *resumable* may be better suited (see https://developers.google.com/drive/api/v3/manage-uploads)

· · ·

Wrap it all up in a script

Now we know our commands work we can create an executable script to do all the work for us. Here we can provide a group of files, it zips them up and then sends them to google drive.

We start by creating a new file with `nano curlgoogle;` and enter the following code — remember to add your own personal auth token! Python 2.7 has been chosen as this is still the default python version on older systems, however the script below should also run for python 3.

It should require no new dependencies provided curl already exists on the system.

```
#!/usr/bin/python

'''
A quick python script to automate curl->googledrive interfacing
This should require nothing more than the system python version and
curl. Written for python2.7 (with 3 in mind).

Dan Ellis 2020
'''

import os,sys,json

if sys.version[0]=='3':
  raw_input = lambda(x): input(x)

##############################
#Owner information goes here!#
##############################
name = 'curldata'
client_id= '<enter your client id>'
client_secret='<enter your client secret>'

##############################


cmd1 = json.loads(os.popen('curl -d
"client_id=%s&scope=https://www.googleapis.com/auth/drive.file"
https://oauth2.googleapis.com/device/code'%client_id).read())

str(raw_input('\n Enter %(user_code)s\n\n at %(verification_url)s
\n\n Then hit Enter to continue.'%cmd1))

str(raw_input('(twice)'))

cmd2 = json.loads(os.popen(('curl -d client_id=%s -d client_secret=%s
-d device_code=%s -d
grant_type=urn~~3Aietf~~3Aparams~~3Aoauth~~3Agrant-
type~~3Adevice_code https://accounts.google.com/o/oauth2/token'%
(client_id,client_secret,cmd1['device_code'])).replace('~~','%')).rea
d())
print(cmd2)

# zip files
cmd3 = os.popen('zip -r %s.zip %s'%(name,'
'.join(sys.argv[1:]))).read
print(cmd3)

cmd4 = os.popen('''
curl -X POST -L \
    -H "Authorization: Bearer %s" \
    -F "metadata={name :\'%s\'};type=application/json;charset=UTF-8"
\
    -F "file=@%s.zip;type=application/zip" \
    "https://www.googleapis.com/upload/drive/v3/files?
uploadType=multipart"
    '''%(cmd2["access_token"],name,name)).read()

print(cmd4)
print('end')
```

We then make it executable `chmod a+x curlgoogle` allowing us to use it in an executable manner:

```
./curlgoogle file1 file2.txt file3.jpg etc...
```

· · ·

Conclusions

And there we have it, an easy way to send multiple log files from a headless machine to a google drive repository, which can be accessed by multiple people for analysis.

If you are in need of more information, stack overflow answers by Tanaike and HAKS (amongst others) were particularly helpful in creating this post.