

How to source functions from non-public GitLab repositories?

5th May 2022 by Victoria Agudetse

Sourcing functions in R

In an R script or session, we can import code written in other files that we want to use (e.g. a function) using `source()`.

Sometimes, when we are working with GitLab, it is more practical to **source the files from the online repository**, rather than from our own local copy.

Sourcing functions from a public GitLab repository is very simple! We just need the URL to the 'raw' file:

```
> source('https://earth.bsc.es/gitlab/external/cstools/-/raw/master/R/s2dv_cube.R')
```

However, when we try to source from an **internal** or **private** repository, things get complicated...



Sourcing from a non-public repository

For example, let's try to source the function `get_regrid_params.R` from the CSOperational repository:

```
> source("https://earth.bsc.es/gitlab/external/cstools/-/raw/master/R/s2dv_cube.R")  
# no problem!
```

```
> source("https://earth.bsc.es/gitlab/es/csoperational/-/raw/master/R/  
get_regrid_params.R")  
Error in source("https://earth.bsc.es/gitlab/es/csoperational/-/raw/master/R/  
get_regrid_params.R") :
```

```
https://earth.bsc.es/gitlab/es/csoperational/-/raw/master/R/get_regrid_params.R:1:1:  
unexpected '<'
```

```
1: <  
   ^
```

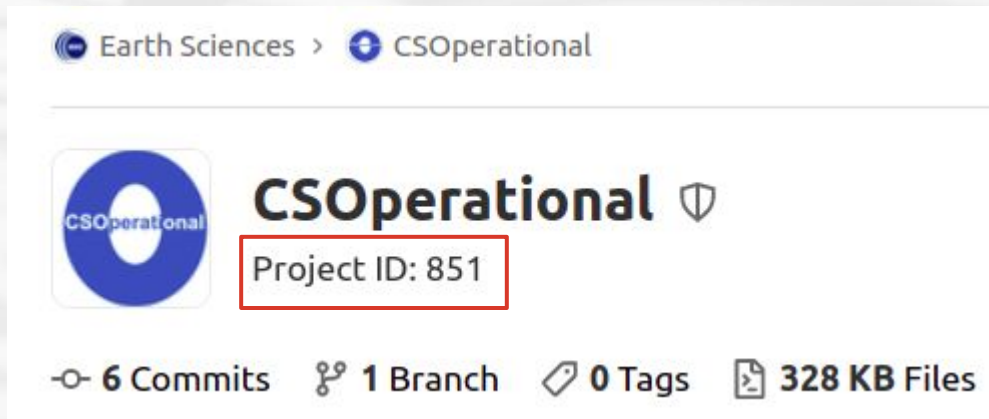
```
# What happened!?
```

The GitLab API: Repository Project ID

GitLab is not returning the raw version of the file! Non-public files can only be downloaded or sourced using the **GitLab API**. The way to access a file through the API is a little different...

First of all, to access any repository through the API, we need its ID number.

We can open our browser and find the repository **Project ID** right underneath its name on GitLab:



The screenshot shows a GitLab repository page for the user 'CSOperational'. At the top, the breadcrumb 'Earth Sciences > CSOperational' is visible. Below this is the repository header, which includes the user's profile picture (a blue circle with 'CSOperational' text), the name 'CSOperational' with a shield icon, and a red-bordered box containing the text 'Project ID: 851'. At the bottom of the header, there are statistics: '6 Commits', '1 Branch', '0 Tags', and '328 KB Files'.

The GitLab API: File (“blob”) SHA

According to the [GitLab API documentation](#), we can access a certain file through its unique SHA (Simple Hashing Algorithm). An SHA is an alphanumeric string that identifies the file.

Raw blob content

Get the raw file contents for a blob by blob SHA. This endpoint can be accessed without authentication if the repository is publicly accessible.

```
GET /projects/:id/repository/blobs/:sha/raw
```



Supported attributes:

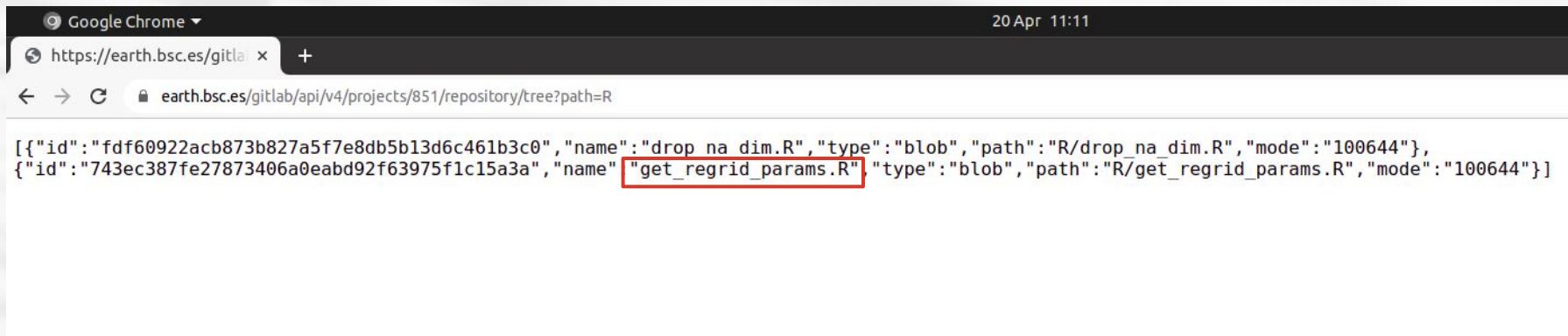
Attribute	Type	Required	Description
<code>id</code>	integer or string	yes	The ID or URL-encoded path of the project owned by the authenticated user.
<code>sha</code>	string	yes	The blob SHA.

How to get your file SHA (I)

We can see the top level files and directories in the repository tree on our browser by typing:

`https://earth.bsc.es/gitlab/api/v4/projects/<project_id>/repository/tree`

If we add “`?path=<path_to_directory>`” at the end of above URL, we will get the contents of our target directory. In this example, the file we want is under a directory named “R”:



```
Google Chrome 20 Apr 11:11
https://earth.bsc.es/gitlab/api/v4/projects/851/repository/tree?path=R
[{"id":"fdf60922acb873b827a5f7e8db5b13d6c461b3c0","name":"drop_na_dim.R","type":"blob","path":"R/drop_na_dim.R","mode":"100644"},
{"id":"743ec387fe27873406a0eabd92f63975f1c15a3a","name":"get_regrid_params.R","type":"blob","path":"R/get_regrid_params.R","mode":"100644"}]
```

How to get your file SHA (II)

Files will have the **"type":"blob"** key-value pair. The value of the "id" key is our file SHA. In this case:

```
"id":"743ec387fe27873406a0eabd92f63975f1c15a3a"
```

Alternative: If you have a local copy of the repository, you can skip this process, open your terminal and simply `cd` to the folder where your file is and type the command `git ls-files -s`:

```
/esarchive/scratch/vagudets/repos/csoperational/R> git ls-files -s  
100644 fdf60922acb873b827a5f7e8db5b13d6c461b3c0 0    drop_na_dim.R  
100644 743ec387fe27873406a0eabd92f63975f1c15a3a 0    get_regrid_params.R
```

The GitLab API: Personal Access Tokens

The last thing you need to do is generate a [Personal Access Token](#) (PAT). You can do this from your GitLab profile following the steps [in this tutorial](#). Choose either 'api' or 'read_api' (more secure if you only need read access).

Once you have your PAT, you can build your URL by replacing the relevant bits. In summary:

```
https://earth.bsc.es/gitlab/api/v4/projects/<project_id>/repository/blobs/<file_id>/raw?ref=<branch>&private_token=<api_token>
```

[<project_id>](#): Repository "Project ID" number

[<file_id>](#): File SHA

[<branch>](#): The name of the branch you want, e.g. 'master'

[<api_token>](#): Personal Access Token for the GitLab API