Python

Jul 12, 2021

Contents

- A utility to clone and pull Gitlab groups, subgroups, projects based on path selection

CHAPTER 1

Purpose

Gitlabber clones or pulls all projects under a subset of groups / subgroups by building a tree from the Gitlab API and allowing you to specify which subset of the tree you want to clone using glob patterns and/or regex expressions.

CHAPTER 2

Installation

- You can install gitlabber from PyPi :

```
pip install gitlabber
```

- You'll need to create an access token from Gitlab with API scopes *read_repository* and *read_api*

CHAPTER 3

Usage

- **Arguments can be provided via the CLI arguments directly or via environment variables::**

Argument	Flag	Environment Variable
token	-t	*GITLAB_TOKEN*
url	-u	*GITLAB_URL*
method	-m	*GITLABBER_CLONE_METHOD*
naming	-n	*GITLABBER_FOLDER_NAMING*
include	-i	*GITLABBER_INCLUDE*
exclude	-x	*GITLABBER_EXCLUDE*

- To view the tree run the command with your includes/excludes and the *-p* flag it will print your tree like so

```
root [http://gitlab.my.com]
├── group1 [/group1]
│   └── subgroup1 [/group1/subgroup1]
│       └── project1 [/group1/subgroup1/project1]
└── group2 [/group2]
    ├── subgroup1 [/group2/subgroup1]
    │   └── project2 [/group2/subgroup1/project2]
    ├── subgroup2 [/group2/subgroup2]
    └── subgroup3 [/group2/subgroup3]
```

- To see how to use glob patterns and regex to filter tree nodes see globre project page .
- Cloning vs Pulling: when running Gitlabber consecutively with same parameters it will scan the local tree structure, if the project directory exists and is a valid git repository (has .git folder in it) gitlabber will perform a git pull in the directory, otherwise the project directory will be created and the gitlab project will be cloned into it.
- Cloning submodules: use the *-r* flag to recurse git submodules, uses the *–recursive* for cloning and utilizes GitPython's smart update method for upading cloned repositories
- Printed Usage:

```
usage: gitlabber [-h] [-t token] [-u url] [--verbose] [-p]
                 [--print-format {json,yaml,tree}] [-m {ssh,https}] [-i csv]
                 [-x csv] [--version]
                 [dest]

Gitlabber - clones or pulls entire groups/projects tree from gitlab

positional arguments:
dest                  destination path for the cloned tree (created if doesn't exist)

optional arguments:
-h, --help            show this help message and exit
-t token, --token token
                        gitlab personal access token https://docs.gitlab.com/ee/user/
↪profile/personal_access_tokens.html
-u url, --url url     base gitlab url (e.g.: 'http://gitlab.mycompany.com')
--verbose             print more verbose output
-p, --print           print the tree without cloning
--print-format {json,yaml,tree}
                        print format (default: 'tree')
-n {name,path}, --naming {name,path}
                        the folder naming strategy for projects (default: "name")
-m {ssh,http}, --method {ssh,http}
                        the git transport method to use for cloning (default: "ssh")
-a {include,exclude,only}, --archived {include,exclude,only}
                        include archived projects and groups in the results (default:
↪"include")
-i csv, --include csv
                        comma delimited list of glob patterns of paths to projects or␣
↪groups to clone/pull
-x csv, --exclude csv
                        comma delimited list of glob patterns of paths to projects or␣
↪groups to exclude from clone/pull
-r, --recursive       clone/pull git submodules recursively
--version             print the version

examples:

    clone an entire gitlab tree using a base url and a token:
    gitlabber -t <personal access token> -u <gitlab url> .

    # the following examples assume you provided token/url in environment variables␣
↪so these arguments are omitted
    only print the gitlab tree:
    gitlabber -p .

    clone only projects under subgroup 'MySubGroup' to location '~/GitlabRoot':
    gitlabber -i '/MyGroup/MySubGroup**' ~/GitlabRoot

    clone only projects under group 'MyGroup' excluding any projects under subgroup
↪'MySubGroup':
    gitlabber -i '/MyGroup**' -x '/MyGroup/MySubGroup**' .

    clone an entire gitlab tree except projects under groups named 'ArchiveGroup':
    gitlabber -x '/ArchiveGroup**' .

    clone projects that start with a case insensitive 'w' using a regular expression:
    gitlabber -i '/{[w].*}' .
```

CHAPTER 4

Debugging

- You can use the *–verbose* flag to get Gitlabber debug messages printed
- For more verbose gitlab messages you can get GitPython module to print more debug messages by setting the environment variable:

```
export GIT_PYTHON_TRACE='full'
```

CHAPTER 5

Toubleshooting

- *GitlabHttpError: 503*: make sure you provide the base url to your gitalb installation (e.g., *https://gitlab.my.com* and not *https://gitlab.my.com/some/nested/path*)

CHAPTER 6

Known Limitations

- Project Renaming: Gitlabber doesn't maintain local state and will not rename local projects when they are renamed on the server and it will clone them again under their new name.
- Folder Naming Strategy: consecutively running gitlabber with different values for the *-n* parameter will produce undesirable results, keep the same value as previous runs or simply don't change it from the default (project name)
- When using gitlab.com observe rate limits when cloning large number of projects and the ones for on-premise installations