Gitutils

Jun 01, 2022

Contents

Gitutils is a tool to facilitate the server-side operations when developing software that uses git repositories.

Gitutils functionalities:

- **addldap** Add a ldap group user to a group.
- **clonegroup** Clones all existing projects within a group.
- **creategroups** Create a new group (or multiple).
- **createprojects** Create a new project (or multiple) inside the specified group.
- **find** General search inside all the groups/projects.
- **fork** Creates a fork from the repository. Doing a fork is strongly recommended to freely experiment your changes and/or development in a safe working space without affecting the original project.
- **login** Fetches the gitlab token (saved in ~/.gitutils_token).
- **merge** Creates a request to merge the defined fork to the original repository.
- **setrole** Sets the role for a specific user on a specific group or project (or multiple)

It is developed using python on an open-source project (github repository), distributed using anaconda via the Paul Scherrer Institute channel (anaconda channel) and documented here using readthedocs. Gitutils authenticates on the git server using the OAuth2 protocol. If the token is non-existent or not valid, gitutils will request username and password and store the token in a file located on the user's home directory called `.gitutils_token`. The user will not be requested for username nor password until the saved token is not valid anymore.

Contents:

CHAPTER 1

Installation

`gitutils` is compatible with Python 3.5+.

Use `conda` to install the latest stable version of `gitutils`:

```
$ conda install -c paulscherrerinstitute gitutils
```

The current development version is available on github.

CHAPTER 2

Getting started with Gitutils

It is assumed that the gitutils' user is familiar with basic git commands (as add, commit, push and pull). If not, we refer to in atlassian tutorial to learn the basics.

Before any command, a token will be fetched to validate the access to the gitlab server. If the user wants to create the token without any specific command, one can use:

```
$ gitutils login
```

A file (~/.gitutils_token) will be created on the home directory and it will store the token.

2.1 Fork Walk-through

1. Define a project to fork and issue the command. Once a repository is forked, it also creates a local clone and an upstream link to the reference repository. Arguments: **-p**, **-n**, **-c**. Examples:
   - To fork and clone into a repository, use the following command:

     ```
     $ gitutils fork <group_name>/<repository_name>
     ```

   - To fork and not clone, add the directive **-n** at the end of the previous command, as in:

     ```
     $ gitutils fork -n <group_name>/<repository_name>
     ```

   - To delete existing fork and create a clean fork of a repository, use the following command:

     ```
     $ gitutils fork -c <group_name>/<repository_name>
     ```

2. Implement the changes/development necessary on the forked repository.
3. Add all changes, commit and push the changes to your forked repository using git command line, as in:

```
$ git add .
$ git commit -m <commit_message>
$ git push
```

Note: When a successful fork happens, it already creates the upstream link. This is done automatically. Therefore, to synchronize your fork with the current state of the original repository and deal with possible merge conflicts, do the following:

```
$ git fetch upstream
$ git merge upstream/master
```

2.2 Merge Walk-through

1. Once all the necessary changes/development have been committed and pushed to a forked repository.
2. **Navigate to the home folder of your forked repository (where the `/.git` folder is). Issue the command to merge. Argume**

    - To create a merge request for a repository, use the following command while on a git repository folder:

    ```
    $ gitutils merge -t <title> -d <description>
    ```

    - To create a merge request for a repository by using the argument **-p** to indicate the project:

    ```
    $ gitutils merge -p <group_name>/<repository_name> -t <title> -d
    ↪<description>
    ```

    - To create a merge request indicating the full-path to the repository and without giving a description:

    ```
    $ gitutils merge -p https://git.psi.ch/<group_name>/<repository_name> -t
    ↪<title>
    ```

Note: Gitutils will assume the command is being executed on the git repository folder. Alternatively, one can use the directive **-p** to indicate directly which project should be merged. If title or description are not provided by the user, default values are going to be used.

2.3 Find

1. The find command will do a general search for all projects and groups.

    - To search for term:

    ```
    $ gitutils find <term>
    ```

    Note: This task can take some minutes depending the number of groups and projects.

    - The output will display the group and the enumerated matching cases according to this example:

```
Gitutils searching for term " S10CB04-CVME-DBAMT1 "...
Group:  archiver_config
    1 )   S10CB04-CVME-DBAMT1  :

    Weblink: https://git.psi.ch/archiver_config/sf_archapp/blob/master/S_
↪DI_BAM_S10CB04-DBAMT1.config#L6

            #  BAM vme ioc cpu/memory usage
            #
            S10CB04-CVME-DBAMT1:MEM_USED                    Monitor 1 60
            S10CB04-CVME-DBAMT1:MEM_FREE                    Monitor 1 60
            S10CB04-CVME-DBAMT1:IOC_CPU_LOAD                Monitor 1 60
            S10CB04-CVME-DBAMT1:UPTIME                      Monitor 1 60
            S10CB04-CVME-DBAMT1:STATUS                      Monitor 1 60
            #
            S10CB04-CVME-DBAMT2:MEM_USED                    Monitor 1 60
```

2.4 Clonegroup

1. **The clonegroup command clones all the existing projects from a specified group.**
    - To clone all projects of group_name:

    ```
    $ gitutils clonegroup <group_name>
    ```

Note: This will clone each repo into its specific folder, depending on the amount of projects this command might take a while. Additionally, a 2 seconds sleep time had to be added in between clones in order not to be blocked by Gitlab API.

2.5 Fork & Merge walk-through

1. Fork and clone a repository:

    ```
    $ gitutils fork <group_name>/<repository_name>
    ```

2. Change the current working directory to your local project `cd <repository_name>`.
3. Do the changes and/or development necessary.
4. Stage your changes to commit by adding them:

    ```
    $ git add .
    ```

5. Commit your changes with a descriptive commit_message:

    ```
    $ git commit -m <commit_message>
    ```

6. Push changes to the forked repository:

    ```
    $ git push
    ```

7. Once you're ready to create the merge request, fetch and merge changes from original repository:

```
$ git fetch upstream
```

Note: Fetch the branches and their respective commits from the upstream repository:

```
$ git merge upstream/master
```

Note: This brings your fork's 'master' branch into sync with the upstream repository without losing your changes. You might have to deal with existing conflicts between your changes and the original repo changes. Decide if you want to keep only your branch's changes, keep only the other branch's changes, or make a brand new change, which may incorporate changes from both branches. If this is the case, go back to step 4 after solving the merge conflicts (add, commit and push the resolved merge conflicts files).

8. Finally, create a merge request:

```
$ gitutils merge -p <group_name>/<repository_name> -t <title> -d <description>
```

Note: if you are located on the repository folder, simply:

```
$ gitutils merge  -t <title> -d <description>
```

CHAPTER 3

Usage

3.1 Gitutils

```
```

usage: gitutils.py [-h] [-e ENDPOINT] {addldap,clonegroup,creategroups,createprojects,find,fork,login,merge,setrole} ...

GITUTILS is a tool to facilitate the server-side operations when developing software that uses git repositories.

optional arguments:

-h, --help show this help message and exit

-e ENDPOINT, --endpoint ENDPOINT Endpoint of the git server. Default: https://git.psi.ch

command: valid commands

{addldap,clonegroup,creategroups,createprojects,find,fork,login,merge,setrole}

commands

addldap Add a ldap group user to a group. clonegroup Clones all existing projects within a group. creategroups Create a new group (or multiple). createprojects Create a new project (or multiple) inside the specified group. find General search inside all the groups/projects. fork Creates a fork from the repository. login Fetches the gitlab token (saved in ~/.gitutils_token). merge Creates a request to merge the defined fork to the original repository. setrole Sets the role for a specific user on a specific group or project (or multiple)

Note: To see the gitutils usage help, you can use:

```
$ gitutils -h
```

3.2 Addldap

```
usage: gitutils.py addldap [-h] group ldapgroup [role]

positional arguments:
group       Group that the LDAP group will be added to.
ldapgroup   LDAP group common name.
role        The role defines the permissions. Options: guest, reporter, dev,␣
↪maintainer, owner
```

optional arguments:

-h, --help	show this help message and exit

Note: To see the addldap usage help, you can use:

```
$ gitutils addldap -h
```

3.3 Clonegroup

```
usage: gitutils.py clonegroup [-h] group [url] [pattern [pattern ...]]

positional arguments:
group       Group name
url         Url to clone the projects: http_url (https://git...) or url (git@git...)
pattern     Initial name pattern of the repositories that will be cloned (3 letters␣
↪minimum).

optional arguments:
-h, --help  show this help message and exit
```

Note: To see the clonegroup usage help, you can use:

```
$ gitutils clonegroup -h
```

3.4 Creategroups

```
usage: gitutils.py creategroups [-h] name [name ...]

positional arguments:
name        Group name or multiple (if multiple groups should be created).

optional arguments:
-h, --help  show this help message and exit
```

Note: To see the creategroups usage help, you can use:

```
$ gitutils creategroups -h
```

3.5 Createprojects

```
usage: gitutils.py createprojects [-h] group name [name ...]

positional arguments:
group       Group name
name        Name of the new project (or multiple separated with spaces).

optional arguments:
-h, --help  show this help message and exit
```

Note: To see the createprojects usage help, you can use:

```
$ gitutils createprojects -h
```

3.6 Find

```
usage: gitutils find [-h] term

positional arguments:
term        Term to search.

optional arguments:
-h, --help  show this help message and exit
```

Note: To see the find usage help, you can use:

```
$ gitutils find -h
```

3.7 Fork

```

```

usage: gitutils.py fork [-h] [-n] [-c] [project]

positional arguments:

project (REQUIRED) Indicates the project to be forked. It must be indicated as follow:

- <group_name>/<project_name>.

optional arguments:

-h, --help show this help message and exit

-n, --no_clone	Indicates that the forked project will not be cloned after forking. A fork will be created on the server-side and no clone nor upstream will be generated on the local git repository.
-c, --clean	Flag to delete personal fork of the project.

Note: To see the fork usage help, you can use:

```
$ gitutils fork -h
```

3.8 Merge

```
usage: gitutils merge [-h] [-t TITLE] [-p PROJECT] [-d DESCRIPTION] project

optional arguments:
  -h, --help            show this help message and exit
  -t TITLE, --title TITLE
                         The title of the merge request that is going to be created.
  -p PROJECT, --project PROJECT
                        Indicates the project to be forked. It can be of four␣
↪different formats:
                        - "" : (DEFAULT) The user doesn't provide this argument, the␣
↪project's group and name
                               will be fetched from the /.git/config folder within the␣
↪path where the
                               gitutils is being called.
                        - <group_name>/<project_name> : The user provides a␣
↪combination of group_name and
                               project_name divided by "/".
  -d DESCRIPTION, --description DESCRIPTION
                        The description of the merge request that is going to be␣
↪created.
```

Note: To see the merge usage help, you can use:

```
$ gitutils merge -h
```

3.9 setrole

```
usage: gitutils.py setrole [-h] [-p] role username group [group ...]

positional arguments:
role           The role defines the permissions. Options: guest, reporter, dev,␣
↪maintainer, owner
username       Username that will be given the role.
group          Group in which the user will be given such role.

optional arguments:
```

(continues on next page)

(continued from previous page)

```
-h, --help     show this help message and exit
-p, --project  If indicated, the setrole gives the access on a project level (and not␣
↪on the default group level).
```

Note: To see the setrole usage help, you can use:

```
$ gitutils setrole -h
```

CHAPTER 4

Development

4.1 Tests

To run the unit tests:

```
$ python -m unittest gitutils/tests/gitutils_test.py
```

To run the functional tests:

```
$ python -m unittest gitutils/tests/gitutils_cmds.py
```

To test new functionalities:

$ python -m gitutils.gitutils <new_cmd>

4.2 Installation for distribution

The package has to be installed as root on `gfalcd.psi.ch`:

```
# Sourcing the python
$ source /opt/gfa/python
# Installing the gitutils package
$ conda install -c paulscherrerinstitute gitutils
```

As this will change the global Python distribution, make sure that only the gitutils package gets updated.

4.3 Building the conda package automatically

After every new release, github actions will automatically build the new package and upload to the anaconda channel being ready to install, as explained above.

4.4 Building the conda package manually

First, login into `gfa-lc6-64`, source the right anaconda environment by executing the command:

```
$ source /opt/gfa/python
```

After that, clone into the repository or pull the latest changes (if you've already cloned it before). The package can be build using:

```
$ conda build conda-recipe
```

Note: Remember to increase the package version before the build (inside *setup.py* and *conda-recipe/meta.yaml*)

After building, the package should be uploaded to anaconda.org via the command displayed at the end of the build process, as in:

```
$ anaconda -t <PERSONAL_CONDA_TOKEN> upload /afs/psi.ch/user/<PATH_TO_USER>/conda-bld/
↪linux-64/<PACKAGE_NAME>
```

Note: If you need to build for different python versions, use the command (where X.X is the specific needed version of python):

```
$ conda build conda-recipe --python=X.X
```

4.5 Contribute and create a merge request

Alternatively, you can fork gitutils:

```
$ gitutils fork controls_highlevel_applications/gitutils
```

Develop new feature(s) on your personal fork, and submit a merge request:

```
# From the home directory of your fork
$ gitutils merge -t <new_feature_merge_title> -d <description_of_new_feature>
```

If the merge request is approved, it will be integrated and the conda package will be updated.

CHAPTER 5

FAQ

Gitutils can't find a group/project... Gitutils have access only to the groups/projects that your token allows it to access. If your permissions are not valid to access certain groups, even though they might exist you will not be able to fork or merge.

I cannot fork the requested repository... It is likely that you need to specify the group and project name in the following format:

```
$ gitutils fork <group_name>/<project_name>
```

It refuses to merge saying that it is not a fork... Make sure when you issue a merge using gitutils that you point to the forked repository. Gitutils will get the original repository (in which the merge request is going to be created) from the properties of the forked repository.

Use

```
$ gitutils merge <personal_fork>/<project_name>
```

Instead of:

```
$ gitutils merge <original_project>/<project_name>
```

Why it doesn't asks for authentication? At the first run of `gitutils` on your local machine, `gitutils` creates a file on your home folder `~/.gitutils_token` and stores your personal token there. It will be continuously used until it is not valid anymore.

If you have any other question, please let us know.

CHAPTER 6

Changelog

All notable changes to gitutils projecct will be documented in this file.

[1.1.5] 2022-06.01

6.1 Added

- Merge cmd new parameters source_branch and destination_branch, allows to work not only with master branches but with a user's specified branch. (default: master)

[1.1.4] 2021-08-17

6.2 Added

- Clonegroup's new parameter to define which url should be used for cloning the projects. http_url or url

6.3 Removed

- Find tests from the behavior tests because of the time that it consumes.

[1.1.3] 2021-05-18

6.4 Added

- Clonegroup command accepts pattern to filter by name in the repositories of the specified group.

6.5 Fixed

- Shows help when no command is used.

[1.1.2] 2021-04-29

6.6 Changed

- Merge allows fork from projects with different repository names.
- Improvements on the fork verbose mode.

6.7 Added

- Find verbose mode.
- Merge verbose mode.
- Behavior test find command.

6.8 Fixed

- Unnecessary confirmation to delete personal fork.
- bugfix (403 forbidden) when retrieving branches inside private projects using gitutils find command.

6.9 Removed

- Cleanup of unused functions and unit tests.

[1.1.1] 2021-04-23

6.10 Added

- Initial implementation of verbosity mode using -v to facilitate debug sessions.

6.11 Changed

- Fork command ignores -c command if a personal fork project is not found (behavior of v1.0.X)

[1.1.0] - 2021-04-22

6.12 Added

- Functional tests Check gitutils/tests/gitutils_cmds.py for details.

6.13 Changed

- Complete gitutils restructure. Each command is now divided into its own python file for easy maintenance and implementation of new functionalities.
- Fork command requires the parameter in the format group_name/project name. Previously used: `https://git.psi.ch/<group_name>/<project_name>` and `<project_name>` are deprecated.
- Fork command only forks into personal space and the removal using the clause -c is not treated on gitutils but directly with git message if not possible.

[1.0.21] - 2020-11-27

6.14 Added

- addldap feature: Add a ldap group user to a group (or multiple).
- setrole feature: Sets the role for a specific user on a specific group or project (or multiple).

6.15 Changed

- optional flag -p when using the find command to specify search on file names only (and not include the content of files on the search).

6.16 Fixed

- Bugfix: verifying if projects were existing and not accessible.
- Bugfix: search for groups/projects was not consistent because of pagination (returning different lists every time).

[1.0.20] - 2020-11-13

6.17 Changed

- Alphabetic order of the commands and the functions in the code.
- Readme instructions

6.18 Added

- creategroups and createprojects commands allowing to do bulk operations with groups and projects.

6.19 Removed

- Search and grep commands.

[1.0.19] - 2020-11-13

6.20 Added

- Find function allows to do a general search for terms within groups and projects.

6.21 Changed

- Search and grep commands are now deprecated (the new find command replaces both).

6.21.1 [1.0.18] - 2020-07-29

6.22 Added

- Publish conda package automatically directly using github actions after a new release.
- Python lint verification (flake8) using github actions.

6.23 Changed

- Improved readme with badges and minor improvements in python format files.

6.23.1 [1.0.17] - 2020-05-20

6.24 Changed

- bugfix when fetching an empty project.

6.24.1 [1.0.15] - 2020-04-09

6.25 Added

- Gitutils search allows users to search for a specific filenames inside the projects of a group.
- Gitutils grep allows users to search for specific filenames and terms inside a specific project.

6.26 Changed

- Improved readme with the instructions for the new commands.

6.26.1 [1.0.14] - 2020-04-03

6.27 Added

- Gitutils clonegroup function allows users to clone into all projects of a existing group.

6.28 Changed

- Improved readme with new command and new help messages.

6.28.1 [1.0.12] - 2020-01-06

6.29 Added

- Gitutils login function allow users to retrieve the token without any related gitutils function.

6.30 Changed

- Increased sleep time after deletion of project because the server wasn't processing it in time.

6.30.1 [1.0.10] - 2019-12-20

6.31 Changed

- Gitutils now uses SSH to perform git commands. HTTP has issues due to security/access.

6.31.1 [1.0.2] - 2019-12-06

6.32 Added

- New parameter on the fork command. -g indicates the group/namespace that the fork will be created. Permissions to do operations in different groups are needed.

6.33 Changed

- Username and password are now appropriately url encoded by using urllib.parse.quote.
- Python-Gitlab method returns only 20 items per search. Fixed by additions parameter all=True in all retrieval of projects or groups.

6.33.1 [1.0.1] - 2019-09-13

6.34 Added

- First release of the gitutils library.
- Gitutils implements fork and merge (server-side) functions using oauth2 authentication.
- Usage of Python-Gitlab library instead of gitlab api.
- Gitutils recovers from an invalid token (fetched from .gitutils_token) by requesting username and password again.
- Gitutils offers a readthedocs documentation.
- gitutils argument '-e' to indicate a different repository endpoint.
- fork argument 'project' is a positional required argument.
- fork argument '-c' to clean existing forks or local folders.
- fork argument '-n' to not clone into forked repository.
- Allow merge argumentless possibility when executing from within the repository's folder.
- Merge allows project indication without the usage of the '-p' flag. Project can also be a positional argument.
- When forking a project that is exists in multiple groups, a list of the groups is displayed.
- Unit tests.
- Oauth2 token saved on user's home directory file '.gitutils_token'.
- Merge allows possibility to define project, title and description. If merge command is executed inside the forked repository's folder, gitutils detects it and does not need the '-p' argument to indicate the project.

Note: The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

CHAPTER 7

Contact

For questions, suggestions or contributions contact leonardo.hax@psi.ch