badges-gitlab

Felipe P. Silva

Oct 16, 2021

CONTENTS

This project was created to generate badges for Gitlab in CI jobs, mainly for private repositories where other common methods are not available (direct API Calls, shields.io, etc…).

By default, Gitlab supports only two types of badges: pipeline and test coverage.

These badges are better detailed at: Gitlab Project Badges.

CHAPTER ONE

INSTALLATION

You can install this package from pypi using pip.

```
$ pip install badges-gitlab
```

CHAPTER

TWO

GENERAL USAGE

```
usage: badges-gitlab [-h] [-p PATH] [-t TOKEN] [--junit-xml FILE_PATH] [-s LABEL MESSAGE␣
↪COLOR]
[-lb URLS [URLS ...]] [-V]

Generate Gitlab Badges using JSON files and API requests. Program version v0.0.0.

optional arguments:
  -h, --help            show this help message and exit
  -p TEXT, --path TEXT  path where json and badges files will be generated/located␣
↪(default: ./public/badges/)
  -t TEXT, --token TEXT specify the private-token in command line (default: ${PRIVATE_
↪TOKEN})
  --junit-xml TEXT      specifies the path of a JUnit XML file for parsing the test␣
↪results
  -s LABEL MESSAGE COLOR, --static-badges LABEL MESSAGE COLOR
                        specify static badges in command line using lists
 -lb URLS [URLS ...], --link-badges URLS [URLS ...]
                        specify shields.io urls to download badges
  -V, --version         returns the package version
```

CHAPTER THREE

AUTHOR

Felipe Pinheiro Silva

3.1 Contributors

Benjamin Maréchal (irmo322)

CHAPTER FOUR

FURTHER DOCUMENTATION

Slowly moving documentation to ReadTheDocs.

4.1 Usage

4.1.1 Common Usage

Install this package from pip and run it in your project folder.

```
$ pip install badges-gitlab
$ badges-gitlab
```

This package was intended to be used in CI jobs, but if you want to test locally, you must point a folder with the json files in the format used by shields.io endpoint, otherwise it won't work because most of the badges uses the Gitlab API Token and CI Environment Variables.

4.1.2 Continuous Integration Job

Below it is an example of a job running at the end of the pipeline in the default branch (main) and using cache for job optimization. Make sure to adequate your pipeline.

To ensure all possible badges are generated, include the personal access token as an environment variable direct into the .gitlab-ci.yml or in the CI/CD Variables configuration.

```
badges:
  image: python:3.9
  stage: badges
  variables:
    PIP_CACHE_DIR: "$CI_PROJECT_DIR/.cache/pip"
    PRIVATE_TOKEN: $ACCESS_TOKEN
  cache:
    key: badges
    paths:
      - .cache/pip
      - venv/
  before_script:
      - python -V
      - pip install virtualenv
      - virtualenv venv
```

(continues on next page)

(continued from previous page)

```
    - source venv/bin/activate
  script:
    - pip install badges-gitlab
    - badges-gitlab -V
    - badges-gitlab
  artifacts:
    when: always
    paths:
      - public/badges/*.svg
    expire_in: 3 months
  rules:
    - if: $CI_COMMIT_BRANCH == $CI_DEFAULT_BRANCH
      when: always
      allow_failure: true
```

As the badges are generated only during this job, and if you want to make sure there are available for some time. So, adjust the expiration of the artifacts accordingly.

Schedule Pipelines

Some badges have dynamic data and are generated only during this job, and the data can be outdated. If you don't want to use third party APIs all the time to generate the badges (sometimes these APIs fail), you could use the pipeline schedule function in conjunction with the rules option to run once a day as example.

```
  rules:
    - if: '$CI_PIPELINE_SOURCE == "schedule"'
```

Non compatible Python jobs

It is possible to use artifacts from previous jobs with information to generate new badges.

The json files must be "shields.io" compliant and located in the folder specified in the badges job (default: public/badges). Below is the accepted format.

```
{
  "schemaVersion": 1,
  "label": "hello",
  "message": "sweet world",
  "color": "orange"
}
```

Dockerfile Example

Alternatively, you can optimize even further the job building a docker image for this job and uploading to Gitlab Container Registry.

```
FROM python:3.9-alpine

MAINTAINER foo@bar.com

RUN pip install badges-gitlab
```

4.1.3 Using the Badges

This package seeks to use the post jobs availability of the artifacts through links, which are described in Gitlab Documentation.

Gitlab Project Badges

You can insert using the project badges Project Badges.

Examples of a link for the project license in project badges section:

```
https://gitlab.com/%{project_path}/-/jobs/artifacts/%{default_branch}/raw/public/badges/
↪license_name.svg?job=badges
```

Readme

Other option is to use in the Readme file, through links. In Gitlab you can leverage the relative links feature.

Example of a link in a markdown Readme.

```
![License](../-/jobs/artifacts/main/raw/public/badges/license_name.svg?job=badges)
```

4.2 Configuration

It is now possible to configure this tool using pyproject.toml. Currently the parameters path, junit_xml, static_badges and link_badges are supported. Example of pyproject.toml section:

```
[tool.badges_gitlab]
    path = "public/badges"
    junit_xml = "tests/report.xml"
    # List of Lists Format [["label", "message", "color"]]
    static_badges = [
        ["conventional commits", "1.0.0", "yellow"]
    ]
    # List of Links
    link_badges = [
        'https://img.shields.io/pypi/wheel/badges-gitlab'
    ]
```

Priority is:

- Command line parameters
- Toml file configuration

4.3 Frequently Asked Questions

Is this project for me?

Although it is possible to generate badges with other API's such as shields.io, usually this process is not available in private repositories.

So if you are hosting a public project, this package is not specifically meant for you as you can workaround with other easier implementations.

One good project to be consulted is from @asdoi, available on https://gitlab.com/asdoi/gitlab-badges and https://gitlab.com/asdoi/git_badges.

But, if you are hosting a private project and don't want to expose your project (Gitlab pages) or don't want to risk exposing your credentials (API Requests), maybe this project is for you.

Another reason would be to avoid overloading servers (e.g. shields.io) with unnecessary requests for (re)creating badges.

How does it work?

Some design choices were made to create this package.

1. The badges' generation were converted into two stages:
   - The first stage uses the Gitlab API (if the private-token turns out to be valid) to generate the json for some badges.
   - The second stage gets all the JSON files from the target folder and creates badges using anybadge.
2. These two stages have a purpose, if any other CI Pipeline job generates json files with their own data, you can also use these files to create badges.
3. The default directory is /public/badges:
   - This folder may be used later for Gitlab pages, although this can be modified through parameters.

4.4 Contributing

Merge requests are welcome. For major changes, please open an issue first to discuss what you would like to change.

Please make sure to update tests as appropriate.

Below are instructions to set up the environment and testing.

4.4.1 Installing Environment

This package uses the Pipenv Virtual Environment for managing the dependencies. They are all listed in the Pipfile.

Current supported version is Python >= 3.8, and this virtual environment is configured for Python 3.8.

Install pipenv if you don't have it yet.

```
$ pip install -U pipenv
```

Clone the Repository and download the dependencies.

```
$ git clone https://gitlab.com/felipe_public/badges-gitlab.git
$ cd badges-gitlab
$ pipenv install --dev
```

4.4.2 Testing

This project uses some tools for static code analysis and the python embedded unittest for Unit Testing.

To run locally the static tests, a script was developed.

```
$ pipenv run statictest
```

To run unittests locally you can use a scripted short version.

```
$ pipenv run unit
```

Dependencies Requirements

This package depends on the following dependencies:

- Python Gitlab API
- Anybadge
- Iso8601
- xmltodict
- toml

4.5 Modules Documentation

4.5.1 CLI

Main Package File, parses CLI arguments and calls functions

`badges_gitlab.cli.`**`main`**`()` → None
    Main Function for calling arg parser and executing functions

`badges_gitlab.cli.`**`parse_args`**`(`*args*`)`
    Create arguments and parse them returning already parsed arguments

4.5.2 Badges API

This VI uses the Gitlab API Functions to create json files for badges

`badges_gitlab.badges_api.`**`create_api_badges`**(*directory_path: Any*, *private_token: str*) → None
    Authenticates to API and call the json creation functions

`badges_gitlab.badges_api.`**`general_data`**(*project_ref: Any*, *directory_path: Any*) → None
    Retrieves General Data

`badges_gitlab.badges_api.`**`issues`**(*project_ref: Any*, *directory_path: Any*) → None
    Retrieves issues count data

`badges_gitlab.badges_api.`**`releases_commits`**(*project_ref: Any*, *directory_path: Any*) → None
    Retrieves Releases, Tags and Commits related data

`badges_gitlab.badges_api.`**`validate_path`**(*directory_path: Any*) → None
    Validates destination path, if not found, creates it

4.5.3 Badges JSON

This modules has functions to manipulate and generate standardized json files

`badges_gitlab.badges_json.`**`json_badge`**(*directory_path*, *filename: str*, *json_string: dict*) → None
    Write to JSON file to disk to the specified directory

`badges_gitlab.badges_json.`**`print_json`**(*label: str*, *message: str*, *color: str*) → dict
    Returns a JSON (Dict) in the format used by shields.io to create badges

`badges_gitlab.badges_json.`**`validate_json_path`**(*directory_path: Any*) → None
    Validate Path

4.5.4 Badges Static

This module handles generation of static badges read from pyproject.toml or from command line parameters

`badges_gitlab.badges_static.`**`convert_list_json_badge`**(*badges_list: list*) → list
    Converts the list of badges list to json format to be printed in the json file

`badges_gitlab.badges_static.`**`download_badges`**(*directory: str*, *badges_urls: list*)
    Get the badges from websites and save it locally. Now this was written specifically for shields.io but it must be studied to use other websites.

`badges_gitlab.badges_static.`**`extract_svg_title`**(*xml_svg*) → str
    Get the raw SVG (XML), convert to dict and retrieve the title

`badges_gitlab.badges_static.`**`print_static_badges`**(*directory: str*, *badges_list: list*)
    Call functions to perform actions in order to write a file with json badge information

`badges_gitlab.badges_static.`**`to_snake_case`**(*value*) → str
    Convert the label from a badge to snake case

4.5.5 Badges SVG

Creates standardized badges files using anybadge package

`badges_gitlab.badges_svg.`**`print_badges`**`(`*directory_path: Any*`)` → None
: Iterates within the directory finding json files and then use anybadge pkg to generate them

`badges_gitlab.badges_svg.`**`replace_space`**`(`*string: str*`)` → str
: Replaces any spaces to make it easier to use later as url in badges linking

`badges_gitlab.badges_svg.`**`validate_json_path`**`(`*directory_path: Any*`)` → bool
: Validates path to check if there is any json files there or the if the directory is valid

4.5.6 Badges Test

Generate Tests Badges related by parsing JUnit XML files

`badges_gitlab.badges_test.`**`create_badges_test`**`(`*json_directory*, *file_path: str*`)` → str
: This function parses a JUnit XML file to extract general information about the unit tests.

`badges_gitlab.badges_test.`**`create_test_json_badges`**`(`*json_directory*, *test_results: list*`)` → str
: This function returns parses a list with the test summary to json format. The list order must be: total tests, total failures, total errors, total_skipped, total_time

`badges_gitlab.badges_test.`**`tests_statistics`**`(`*stats_tests_dict: dict*, *testsuite*`)` → dict

This function returns the Test Statistics Dictionary with added values from the testsuite.

Args: stats_tests_dict (dict): dictionary with listed tests testsuite ([junitparser.junitparser.TestSuite]): a testsuite xml node needed for filling the stats tests dicitionary

Returns: dict: returns the stats_tests_dict with the new values.

4.5.7 Read pyproject.toml

Read pyproject.toml and import the parameters to be used in the function

`badges_gitlab.read_pyproject.`**`load_pyproject`**`(`*file_path: str*`)` → dict
: Load the tool.badges_gitlab section from the toml file. Most of the cases it is pyproject.toml because it is hard-coded into main function

`badges_gitlab.read_pyproject.`**`pyproject_config`**`(`*file_path: str*`)` → dict
: Check if the file exists then return the dictionary if it exists with the configuration for the badges_gitlab tool

`badges_gitlab.read_pyproject.`**`pyproject_exists`**`(`*file_path: str*`)` → bool
: Verify if the file exists, used internally.

4.6 Changelog

All notable changes to this project will be documented in this file.

See python-semantic-release for commit guidelines and this project adheres to Semantic Versioning.

4.6.1 v0.8.3 (2021-10-16)

Fix

- Change definition of color for test result badges (#29) (`0845f4c`)
- Minor typo in badges test complete svg (#30) (`82e567b`)
- Includes back the xmltodict (#28) (`57aa4bb`)

4.6.2 v0.8.2 (2021-10-11)

Fix

- Bug with parsing xml without testsuites (#26) (`f4b621c`)

4.6.3 v0.8.1 (2021-09-12)

Fix

- Bug with parsing xml with one testsuite only (#25) (`6be7060`)
- Adds incrementing func for stats tests dict (#25) (`41417d1`)

Documentation

- Update docs to include asdoi projects (#24). (`7ffe89a`)

4.6.4 v0.8.0 (2021-05-26)

Feature

- Include documentation using sphinx (`6eb6b3f`)

4.6.5 v0.7.0 (2021-05-23)

Feature

- Adds support for downloading shields.io badges (#13) (`1a2a606`)
- **deps:** Adds requests as dependencu (#13) (`eb410ed`)

Documentation

- Update documentation to feature #13 (`52d6879`)

4.6.6 v0.6.0 (2021-05-20)

Feature

- Creates the option for printing static badges from lists (#14) (`1a3491b`)

Documentation

- Readme updated with static badges feature (#14) (`3876ba3`)

4.6.7 v0.5.1 (2021-05-19)

Fix

- Naming of the junit_xml key in the pyproject.toml (#21) (`004c255`)

Documentation

- Update setup.py (`2513d97`)

4.6.8 v0.5.0 (2021-05-19)

Feature

- Includes support for configuring in pyproject.toml (#20) (`132f160`)
- **deps:** Includes toml as dependency (#20) (`a6101dc`)

Documentation

- Update readme with information about pyproject.toml (#20) (`c8d1785`)

4.6.9 v0.4.0 (2021-05-17)

Feature

- Includes the option to generate badges for tests results (#2) (`2ca6d26`)

4.6.10 v0.3.0 (2021-05-15)

Feature

- Includes command line option for printing version (#11) (`54606f5`)

Fix

- Changes the badges' labels to lower case (#6) (`1b2e1a2`)

Documentation

- Project documentation is updated in readme (`2a5df92`)

4.6.11 v0.2.0 (2021-05-13)

Feature

- Implementes semantic versioning in the package (#15) (`7e9169a`)
- **deps:** Includes dependencies (`268af3b`)

PYTHON MODULE INDEX

INDEX