quickCI Documentation

Release 0.4.0

Roberto Preste

Dec 19, 2019

CONTENTS:

Have a quick look at the status of CI projects from the command line.

- Free software: MIT license
- Documentation: https://quickci.readthedocs.io
- GitHub repo: https://github.com/robertopreste/quickci

CHAPTER

ONE

FEATURES

quickCI allows to have a quick overview of the status of build jobs on several CI services, for a specific branch of the repository being built. Currently, quickCI supports checking build status for the following CI services:

- Travis CI
- CircleCI
- AppVeyor
- Buddy
- Drone

More services to come!

CHAPTER

TWO

USAGE

2.1 Configuration

1. Create a config file (it will be located in `~/.config/quickci/tokens.json`):

```
$ quickci config create
```

2. Replace placeholders with your own authentication tokens:

```
$ quickci config update <service> <token>
```

Available services are:

- Travis CI: `travis`
- CircleCI: `circle`
- AppVeyor: `appveyor`
- Buddy: `buddy`
- Drone: `drone`

3. Check that everything is correct:

```
$ quickci config show
```

2.2 Check build status

Check the build status of your projects:

```
$ quickci status
```

The build status of your Travis CI, CircleCI, AppVeyor, Buddy and Drone projects will be returned (`master` branch). If you want to monitor one specific branch of your repositories (suppose you have many repos with a dedicated `dev` branch for development), you can easily add the `--branch <branch_name>` option:

```
$ quickci status --branch dev
```

If the `--branch` option is not provided, the build status of the `master` branch will be retrieved by default.

If you want to check one specific repository, you can provide the `--repo <reponame>` option:

```
$ quickci status --repo my_repo
```

It is obviously possible to combine the `--repo` and `--branch` options to check a given branch of a specific repository.

It is also possible to check a specific service using subcommands of `quickci status`:

```
$ quickci status travis
$ quickci status circle
$ quickci status appveyor
$ quickci status buddy
$ quickci status drone
```

These subcommands also accept the `--branch` and `--repo` options. If the token for a specific service is not listed in `~/.config/quickci/tokens.json`, it is possible to provide it using the `--token <service_token>` option:

```
$ quickci status travis --token <TravisCI token>
```

Please refer to the Usage section of the documentation for further information.

CHAPTER

THREE

INSTALLATION

quickCI can be installed using pip (**Python>=3.6 only**):

```
$ pip install quickci
```

Please refer to the Installation section of the documentation for further information.

CHAPTER

FOUR

CREDITS

This package was created with Cookiecutter and the cc-pypackage project template.

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 quickCI

Have a quick look at the status of CI projects from the command line.

- Free software: MIT license
- Documentation: https://quickci.readthedocs.io
- GitHub repo: https://github.com/robertopreste/quickci

5.1.1 Features

quickCI allows to have a quick overview of the status of build jobs on several CI services, for a specific branch of the repository being built. Currently, quickCI supports checking build status for the following CI services:

- Travis CI
- CircleCI
- AppVeyor
- Buddy
- Drone

More services to come!

5.1.2 Usage

Configuration

1. Create a config file (it will be located in `~/.config/quickci/tokens.json`):

```
$ quickci config create
```

2. Replace placeholders with your own authentication tokens:

```
$ quickci config update <service> <token>
```

Available services are:

- Travis CI: `travis`
- CircleCI: `circle`
- AppVeyor: `appveyor`
- Buddy: `buddy`
- Drone: `drone`

3. Check that everything is correct:

```
$ quickci config show
```

Check build status

Check the build status of your projects:

```
$ quickci status
```

The build status of your Travis CI, CircleCI, AppVeyor, Buddy and Drone projects will be returned (`master` branch). If you want to monitor one specific branch of your repositories (suppose you have many repos with a dedicated `dev` branch for development), you can easily add the `--branch <branch_name>` option:

```
$ quickci status --branch dev
```

If the `--branch` option is not provided, the build status of the `master` branch will be retrieved by default.

If you want to check one specific repository, you can provide the `--repo <reponame>` option:

```
$ quickci status --repo my_repo
```

It is obviously possible to combine the `--repo` and `--branch` options to check a given branch of a specific repository.

It is also possible to check a specific service using subcommands of `quickci status`:

```
$ quickci status travis
$ quickci status circle
$ quickci status appveyor
$ quickci status buddy
$ quickci status drone
```

These subcommands also accept the `--branch` and `--repo` options. If the token for a specific service is not listed in `~/.config/quickci/tokens.json`, it is possible to provide it using the `--token <service_token>` option:

```
$ quickci status travis --token <TravisCI token>
```

Please refer to the Usage section of the documentation for further information.

5.1.3 Installation

quickCI can be installed using pip (**Python>=3.6 only**):

```
$ pip install quickci
```

Please refer to the Installation section of the documentation for further information.

5.1.4 Credits

This package was created with Cookiecutter and the cc-pypackage project template.

5.2 Installation

5.2.1 Stable release

To install quickCI, run this command in your terminal (**Python>=3.6 only**):

```
$ pip install quickci
```

This is the preferred method to install quickCI, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

5.2.2 From sources

The sources for quickCI can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/robertopreste/quickci
```

Or download the tarball:

```
$ curl  -OL https://github.com/robertopreste/quickci/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

5.3 Usage

5.3.1 Command Line Interface

`quickci` offers two main CLI commands:

- `status` shows the current status of your projects on one or more CI services;
- `config` creates or updates the configuration file needed.

quickci status

This command retrieves the status of your projects on one or more CI services (`master` branch by default).

If issued as `quickci status`, will retrieve these information for all the CI services for which an authentication token is available in the config file:

```
$ quickci status

CircleCI (master branch)
    project1 -> success
    project2 -> failed
Travis CI (master branch)
    project1 -> passed
    project2 -> passed
AppVeyor (master branch)
    project1 -> passed
Buddy (master branch)
    project2 -> enqueued
Drone CI (master branch)
    project1 -> success
```

If you want to monitor one specific branch of your repositories (suppose you have many repos with a dedicated `dev` branch for development), you can easily add the `--branch <branch_name>` option:

```
$ quickci status --branch dev
```

If the `--branch` option is not provided, the build status of the `master` branch will be retrieved by default.

If you want to check one specific repository, you can provide the `--repo <reponame>` option:

```
$ quickci status --repo my_repo
```

It is obviously possible to combine the `--repo` and `--branch` options to check a given branch of a specific repository.

It is also possible to check a specific service using subcommands of `quickci status`:

```
$ quickci status travis
$ quickci status circle
$ quickci status appveyor
$ quickci status buddy
$ quickci status drone
```

These subcommands also accept the `--branch` and `--repo` options:

```
$ quickci status travis --branch master
$ quickci status circle --branch feature1 --repo my_repo
$ quickci status drone --branch new_feature --repo my_other_repo
```

If you have not set up a config file, you can still retrieve information from CI services providing their authentication token right into the command:

```
$ quickci status travis --token <TRAVIS_CI_TOKEN>
```

quickci config

This command allows to create a config file for `quickci`, or update it if a config file is already available.

The `create` command will create a brand new config file, located in `~/.config/quickci/tokens.json`:

```
$ quickci config create
```

If a config file is already present at that location, you will be prompted to confirm your desire to clear it and create a new one. New config files fill the authentication tokens with a temporary string, which you will need to update with proper tokens.

The `update` command allows to update one of the authentication tokens in the existing config file:

```
$ quickci config update <CIservice> <token>
```

The `show` command will show all the stored authentication tokens:

```
$ quickci config show
```

5.4 API

5.4.1 quickci status

Return the status of the given branch of each project in each CI.

```
quickci status [OPTIONS] COMMAND [ARGS]...
```

Options

-b, --branch <branch>
Branch to check

-r, --repo <repo>
Repo to check

appveyor

Return the status of the given branch of each project in AppVeyor.

```
quickci status appveyor [OPTIONS]
```

Options

-t, --token <token>
AppVeyor auth token

-b, --branch <branch>
Branch to check

-r, --repo <repo>
Repo to check

buddy

Return the status of the given branch of each project in Buddy.

```
quickci status buddy [OPTIONS]
```

Options

-t, --token <token>
Buddy auth token

-b, --branch <branch>
Branch to check

-r, --repo <repo>
Repo to check

circle

Return the status of the given branch of each project in CircleCI.

```
quickci status circle [OPTIONS]
```

Options

-t, --token <token>
CircleCI auth token

-b, --branch <branch>
Branch to check

-r, --repo <repo>
Repo to check

drone

Return the status of the given branch of each project in Drone CI.

```
quickci status drone [OPTIONS]
```

Options

-t, --token <token>
Drone CI auth token

-b, --branch <branch>
Branch to check

-r, --repo <repo>
Repo to check

travis

Return the status of the given branch of each project in Travis CI.

```
quickci status travis [OPTIONS]
```

Options

-t, --token <token>
Travis CI auth token

-b, --branch <branch>
Branch to check

-r, --repo <repo>
Repo to check

5.4.2 quickci config

Create or update the configuration file.

```
quickci config [OPTIONS] COMMAND [ARGS]...
```

create

Create a new configuration file or overwrite an existing one.

```
quickci config create [OPTIONS]
```

show

Display the content of the configuration file.

```
quickci config show [OPTIONS]
```

update

Update a specific service token in the configuration file.

```
quickci config update [OPTIONS] [travis|circle|appveyor|buddy] TOKEN
```

Arguments

SERVICE
Required argument

TOKEN
Required argument

5.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.5.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/robertopreste/quickci/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

quickCI could always use more documentation, whether as part of the official quickCI docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/robertopreste/quickci/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.5.2 Get Started!

Ready to contribute? Here's how to set up *quickci* for local development.

1. Fork the *quickci* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/quickci.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv quickci
$ cd quickci/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 quickci tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5 and 3.6. Check https://travis-ci.org/robertopreste/quickci/pull_requests and make sure that the tests pass for all supported Python versions.

5.5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_quickci
```

5.5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch  # possible: major / minor / patch
$ git push
$ git push --tags
```

5.6 Credits

5.6.1 Development Lead

- Roberto Preste <robertopreste@gmail.com>

5.6.2 Contributors

None yet. Why not be the first?

5.7 History

5.7.1 0.1.0 (2019-04-20)

- First release.

0.1.1 (2019-04-29)

- Update Config methods and attributes for better handling of tokens;
- Update CLI commands.

0.1.2 (2019-06-03)

- Minor code fix;
- Update requirements and documentation;
- Fix AppVeyor request class and add GitLab draft.

0.1.3 (2019-06-06)

- Add Buddy class.

0.1.4 (2019-06-07)

- Change fetching functions to asyncio.

5.7.2 0.2.0 (2019-07-02)

- Change `config` and `status` commands to group commands and add related subcommands;
- Change classes to use concurrent functions when possible;
- Clean code.

0.2.1 (2019-07-03)

- Fix imports and tox test config.

0.2.2 (2019-07-03)

- Fix setup.py installation process;
- Update documentation.

0.2.3 (2019-07-13)

- Add Drone CI class and CLI commands;
- Update tests;
- Update documentation.

5.7.3 0.3.0 (2019-07-28)

- Add `--branch` option to check for specific branch;
- Update documentation.

5.7.4 0.4.0 (2019-12-12)

- Add `--repo` option to check for a specific repository;
- Update documentation.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Symbols

Q

S

T