circleci.py Documentation

Release 2.0.0

Lev Lazinskiy

Feb 23, 2020

Contents

circleci.py is a python wrapper around the CircleCI API.

The source code for circleci.py can be found on GitHub.

CHAPTER 1

Quickstart

1.1 Quickstart

1.1.1 Installation

Note: circleci.py requires python3

You can install the latest version of circleci.py with:

```
pip install circleci
```

1.1.2 Basic Usage

Make a new API token in the CircleCI application.

Import the CircleCI API and start using methods:

```
from circleci.api import Api

circleci = Api("$YOUR_TOKEN")

# get info about your user
circleci.get_user_info()

# get list of all of your projects
circleci.get_projects()
```

CHAPTER 2

Tutorial

CHAPTER 3

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

3.1 API

Note: Unless otherwise noted all arguments are of the `str` type.

3.1.1 API Object

circleci.api

This module provides a class which abstracts the CircleCI REST API.

Changed in version 2.0.0: Removed legacy 1.0 endpoints. See CHANGELOG for more details.

class `circleci.api.`**`Api`**(*token*, *url='https://circleci.com/api/v1.1'*)

A python interface into the CircleCI API

Instantiate a new circleci.Api object.

Parameters

- **`url`** – The URL to the CircleCI instance. Defaults to https://circleci.com/api/v1.1. If you are running CircleCI server, the API is available at the same endpoint of your own installation url. i.e (https://circleci.yourcompany.com/api/v1.1).
- **`token`** – Your CircleCI API token.

`_download`(*url*, *destdir=None*, *filename=None*)

File download helper.

Parameters

- **url** – The URL to the artifact.
- **destdir** – The optional destination directory. Defaults to None (curent working directory).
- **filename** – Optional file name. Defaults to the name of the artifact file.

_request (*verb*, *endpoint*, *data=None*)
Request a url.

Parameters

- **endpoint** – The api endpoint we want to call.
- **verb** – POST, GET, or DELETE.
- **params** (*dict*) – Optional build parameters.

Raises **requests.exceptions.HTTPError** – When response code is not successful.

Returns A JSON object with the response from the API.

add_envvar (*username*, *project*, *name*, *value*, *vcs_type='github'*)
Adds an environment variable to a project

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **name** – Name of the environment variable.
- **value** – Value of the environment variable.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: POST: `/project/:vcs-type/:username/:project/envvar`

add_ssh_key (*username*, *project*, *ssh_key*, *vcs_type='github'*, *hostname=None*)
Create an ssh key

Used to access external systems that require SSH key-based authentication.

Note: The ssh_key must be unencrypted.

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **branch** – Defaults to master.
- **ssh_key** – Private RSA key.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.
- **hostname** – Optional hostname. If set, the key will only work for this hostname.

Endpoint: POST: `/project/:vcs-type/:username/:project/ssh-key`

add_ssh_user (*username*, *project*, *build_num*, *vcs_type='github'*)
Adds a user to the build's SSH permissions.

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **build_num** – Build number.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: POST: `/project/:vcs-type/:username/:project/:build_num/ssh-users`

cancel_build(*username*, *project*, *build_num*, *vcs_type='github'*)
Cancels the build.

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **build_num** – Build number.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: POST: `/project/:vcs-type/:username/:project/:build_num/cancel`

create_checkout_key(*username*, *project*, *key_type*, *vcs_type='github'*)
Create a new checkout keys for a project

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **key_type** – The type of key to create. Valid values are 'deploy-key' or 'github-user-key'
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Raises **InvalidKeyError** – When key_type is not a valid key type.

Endpoint: POST: `/project/:vcs-type/:username/:project/checkout-key`

delete_checkout_key(*username*, *project*, *fingerprint*, *vcs_type='github'*)
Delete a checkout key.

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **fingerprint** – The fingerprint of the checkout key.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: DELETE: `/project/:vcs-type/:username/:project/checkout-key/:fingerprint`

delete_envvar(*username*, *project*, *name*, *vcs_type='github'*)
Delete an environment variable

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`name`** – Name of the environment variable.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: DELETE `/project/:vcs-type/:username/:project/envvar/:name`

`download_artifact`(*url*, *destdir=None*, *filename=None*)
Download an artifact from a url

Parameters

- **`url`** – The URL to the artifact.
- **`destdir`** – The optional destination directory. Defaults to None (curent working directory).
- **`filename`** – Optional file name. Defaults to the name of the artifact file.

`follow_project`(*username*, *project*, *vcs_type='github'*)
Follow a new project on CircleCI.

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: POST: `/project/:vcs-type/:username/:project/follow`

`get_artifacts`(*username*, *project*, *build_num*, *vcs_type='github'*)
List the artifacts produced by a given build.

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`build_num`** – Build number.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET: `/project/:vcs-type/:username/:project/:build_num/artifacts`

`get_build_info`(*username*, *project*, *build_num*, *vcs_type='github'*)
Full details for a single build.

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`build_num`** – Build number.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET: `/project/:vcs-type/:username/:project/:build_num`

`get_checkout_key`(*username*, *project*, *fingerprint*, *vcs_type='github'*)
Get a checkout key.

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`fingerprint`** – The fingerprint of the checkout key.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET: `/project/:vcs-type/:username/:project/checkout-key/:fingerprint`

`get_envvar`(*username*, *project*, *name*, *vcs_type='github'*)
Gets the hidden value of an environment variable

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`name`** – Name of the environment variable.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET `/project/:vcs-type/:username/:project/envvar/:name`

`get_latest_artifact`(*username*, *project*, *branch=None*, *status_filter='completed'*, *vcs_type='github'*)
List the artifacts produced by the latest build on a given branch.

Note: This endpoint is a little bit flakey. If the "latest" build does not have any artifacts, rathern than returning an empty set, the API will 404.

Parameters

- **`username`** – org or user name
- **`project`** – case sensitive repo name
- **`branch`** – The branch you would like to look in for the latest build. Returns artifacts for latest build in entire project if omitted.
- **`filter`** – Restricts which builds are returned. defaults to 'completed' valid filters: "completed", "successful", "failed"
- **`vcs_type`** – defaults to github on circleci.com you can also pass in bitbucket

Raises ***`InvalidFilterError`*** – when filter is not a valid filter.

Endpoint: GET: `/project/:vcs-type/:username/:project/latest/artifacts`

`get_project_build_summary`(*username*, *project*, *limit=30*, *offset=0*, *status_filter=None*, *branch=None*, *vcs_type='github'*)
Build summary for each of the last 30 builds for a single git repo.

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **limit** (*int*) – The number of builds to return. Maximum 100, defaults to 30.
- **offset** (*int*) – The API returns builds starting from this offset, defaults to 0.
- **status_filter** – Restricts which builds are returned. Set to "completed", "successful", "running" or "failed". Defaults to no filter.
- **branch** – Narrow returned builds to a single branch.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Raises ***InvalidFilterError*** – when filter is not a valid filter.

Endpoint: GET: `/project/:vcs-type/:username/:project`

get_projects()

List of all the projects you're following on CircleCI.

Endpoint: GET: `/projects`

get_recent_builds(*limit=30*, *offset=0*)

Build summary for each of the last 30 recent builds, ordered by build_num.

Parameters

- **limit** (*int*) – The number of builds to return. Maximum 100, defaults to 30.
- **offset** (*int*) – The API returns builds starting from this offset, defaults to 0.

Endpoint: GET: `/recent-builds`

get_test_metadata(*username*, *project*, *build_num*, *vcs_type='github'*)

Provides test metadata for a build

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.
- **build_num** – Build number.
- **vcs_type** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET: `/project/:vcs-type/:username/:project/:build_num/tests`

get_user_info()

Provides information about the signed in user.

Endpoint: GET: `/me`

list_checkout_keys(*username*, *project*, *vcs_type='github'*)

List checkout keys for a project

Parameters

- **username** – Org or user name.
- **project** – Case sensitive repo name.

- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET: `project/:vcs-type/:username/:project/checkout-key`

`list_envvars`(*username*, *project*, *vcs_type='github'*)
Provides list of environment variables for a project

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: GET: `/project/:vcs-type/:username/:project/envvar`

`retry_build`(*username*, *project*, *build_num*, *ssh=False*, *vcs_type='github'*)
Retries the build.

Parameters

- **`username`** – Org or user name.
- **`project`** – Case sensitive repo name.
- **`build_num`** – Build number.
- **`ssh`** (*`bool`*) – Retry a build with SSH enabled. Defaults to False.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: POST: `/project/:vcs-type/:username/:project/:build_num/retry`

`trigger_build`(*username*, *project*, *branch='master'*, *revision=None*, *tag=None*, *parallel=None*, *params=None*, *vcs_type='github'*)
Triggers a new build.

Note:

- `tag` and `revision` are mutually exclusive.
- `parallel` is ignored for builds running on CircleCI 2.0

Parameters

- **`username`** – Organization or user name.
- **`project`** – Case sensitive repo name.
- **`branch`** – The branch to build. Defaults to master.
- **`revision`** – The specific git revision to build. Default is null and the head of the branch is used. Can not be used with the tag parameter.
- **`tag`** – The git tag to build. Default is null. Cannot be used with the tag parameter.
- **`parallel`** (*`int`*) – The number of containers to use to run the build. Default is null and the project default is used.
- **`params`** (*`dict`*) – Optional build parameters.
- **`vcs_type`** – Defaults to github. On circleci.com you can also pass in `bitbucket`.

Endpoint: POST: `project/:vcs-type/:username/:project/tree/:branch`

3.1.2 Errors

circleci.error

This module provides some classes that subclass Exception to provide some useful exceptions for the API wrapper.

class `circleci.error.`**`CircleCIException`**(*argument*)

Base class for CircleCI exceptions

Parameters **`argument`** – The argument that was passed into the function.

class `circleci.error.`**`BadVerbError`**(*argument*)

Exception raises for bad HTTP verb

Parameters **`argument`** – The argument that was passed into the function.

`message = "verb must be one of 'GET', 'POST', or 'DELETE'"`

class `circleci.error.`**`BadKeyError`**(*argument*)

Exception raises for bad Key Type

Parameters **`argument`** – The argument that was passed into the function.

`message = "key must be one of 'deploy-key' or 'github-user-key'"`

class `circleci.error.`**`InvalidFilterError`**(*argument*, *filter_type*)

Exception raises for an invalid filter

Parameters

- **`argument`** – The argument that was passed into the function.
- **`filter_type`** – Filter for status or artifacts.

`filter_message = "status_filter must be one of 'completed''successful', 'failed', or '`

`artifacts_message = "must be one of 'completed', 'successful', or 'failed'"`

CHAPTER 4

Utilities

Refernce for various utility modules in circleci.py

4.1 Utilities

4.1.1 version module

circleci.version

This module provides some helper functions to set version in various places.

New in version 1.2.0.

`circleci.version.`**`VERSION = '2.0.0'`**
Current version of circleci.py.

`circleci.version.`**`get_short_version`**`()`
Format "short" version in the form of X.Y

Returns short version

CHAPTER 5

Developer Documentation

5.1 Developer Documentation

5.1.1 Installing Development Environment

Your life will be a lot better if you use a virtualenv when working with python.

1. Fork and Clone this repo
2. Install python-pip and virtualenv if you do not already have it.
3. Create a new virtualenv with `virtualenv -p python3 env`.
4. Actiavte the new virtualenv with `source env/bin/activate`.
5. Run `make dev`
6. Hack away!

5.1.2 Running Tests

Tests can be found in the `tests` directory.

You can run tests with `make tests`.

If you want to run a specific test file you can do so with:

```
python -m unittest tests/circle/test_$MODULE.py
```

This project has two main types of tests.

- Unit tests. These are tests of specific functions using mocked API data.
- Integration tests. These are tests that actually hit the CircleCI API. Unfortunately, due to the way that permissions work most of the currently written tests will only work properly for the `levlaz` user and token.

Code Coverage

This project attempts to have 100% code coverage. when you run `make test` code coverage is automatically ran. You can view the code coverage report locally by opening up the index.html file in the `htmlcov` directory that gets created when you run `make test`.

5.1.3 Documentation

This project uses sphinx for documentation. You can generate the latest docs locally by running `make docs`. You can then view them by opening up the `index.html` file in the `docs/build/html` directory.

5.1.4 Linting and Style

This project follows the PEP 8 style guidelines. You can install `pylint` in order to ensure that all of your code is compliant with this standard.

CHAPTER 6

Changelog

6.1 Changelog

Here you can see the full list of changes between each circleci.py release.

6.1.1 Version 2.0.0

Warning: This release deprecates all 1.0 endpoints. If you need to continue to use some of the 1.0 endpoints (i.e. you are running CircleCI server and still have some 1.0 builders) then you should pin yourself to the 1.x release.

Released on $DATE

- Split off the SDK into it's own repo. It can now be found here. This decision was made to reduce the complexity of this library, and decouple the development of the mature API wrapper from the very early stage SDK.
- Remove 1.0 features. This API wrapper no longer works with any of the legacy 1.0 endpoints. The specific methods that have been removed are shown below:
  - The Experimental module which implemented a method to retry a build without cache.
  - The clear_cache() method, Caches in 2.0 are immutable.
  - The add_heroku_key() method. Heroku is no longer auto configured in CircleCI 2.0+; this means you need to use environment variables for this type of configuration.

6.1.2 Version 1.2.2

Note: What happened to 1.2.0 and 1.2.1?

Due to a bug in the SDK, these releases has been unpublished from pypi. The bug was an issue in the build_singleton logic that did not exclude the current build, therefore putting us into an infinite loop. Ironically, I did not test this on

CircleCI so this is why the bug was not found in testing. Since pypi does not allow re-publishing versions (for good reason) I had to unpublish 1.2.0 and 1.2.1

In the future, I may start to use the -dev publishing model to try to verify this behavior ahead of time.

Released on March 10, 2019

- Add SDK module which allows folks to do intersting or complicated things using the API.
- Add demo modules which shows some sample usage of the API and SDK.
- Add version module which provides some helpers to get the correct version of circleci.py in various places.

6.1.3 Version 1.1.3

Released on November 19, 2018

- Fix a bug where triggering a build with parameters was not working. Thanks to @hush-hush for the contribution.
- Update CI configuration for CircleCI 2.1. Thanks to @felicianotech for the contribution.
- Added smoke tests to test out all supported versions of Python3.

6.1.4 Version 1.1.2

Released on August 29, 2018

- Minor patch release, which unpins the request library. Thanks to @r1b for the contribution.

6.1.5 Version 1.1.1

Released on October 29, 2017

- 100% Code Coverage
- Add Sphinx and upload docs to readthedocs.org
- Create CD pipeline to pypi when using git tags
- **Implement additional API endpoints**
  - Add Heroku Key #18
  - Test Metadata #17
  - Update trigger build to handle optional build parameters #12
  - Get artifacts of latest build #4
  - Update retry build to include retry with SSH #5
  - Add ability to download artifacts #3
  - **Work with Env Vars**
    - List #13
    - Add #14
    - Get #15
    - Delete #16

6.1.6 Version 1.1.0

Released on October 25, 2017

- **Basic project tooling put into place.**
    - Continuous Integration with CircleCI
    - Packaging and uploading to PyPI
    - Code Coverage
    - Testing with unittest
- Basic Documentation in place
- Add support for using the API with a token with both circleci.com and CircleCI Server
- Add support for most basic API endpoints that deal with projects and builds
- Add Experimental API for using API methods that are undocumented

CHAPTER 7

Attribution

- circleci.py relies on the wonderful requests library for all HTTP requests. This library is licensed under the Apache License, Version 2.0.
- A majority of the API doc strings are adapted from the official CircleCI API documentation which is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Python Module Index

Index