Maintain Documentation

Release 0.1.0

Kyle Fuller

Sep 10, 2018

Contents

A unified interface to maintaining projects of any language.

CHAPTER 1

Installation

```
$ pip install maintain
```

CHAPTER 2

Release Automation

Maintain offers a command to automate bumping version numbers and releasing your project.

```
$ maintain release patch
```

This release command will figured out the type of package, whether it be a Python package, Ruby Gem, NPM package etc and then perform the steps necessary to release it. It will bump the version number, create a release commit, tag the release and push it to the respective package manager.

Maintain allows you to specify the *major*, *minor* or *patch* to automatically bump the respective version number, or you can explicitly specify a version.

You may also specify *semver* as the version to automatically determine the next semantic version. This is not supported on all releasers.

2.1 Pull-request

You can configure Maintain to create a pull request and perform the release in two steps. This is useful if you use code-review for the release process:

```
$ maintain release 1.2.0-beta.1 --pull-request
```

Once the pull request is merged, you can perform the actual release:

```
$ maintain release --no-bump
```

Note: This step could be done during continuous integration.

2.2 Releasers

Maintain supports the following releasers:

2.2.1 Repo

Commands to operate on a collection of repositories at a time.

print

Subcommand to print all matched repositories, this is useful to test which repositories will be used for other operations.

check

Checks all repositories for unstaged, unsynced or untracked changes.

run

Allows running the provided command on a collection of repositories.

Options

-s/–silent - Don’t print subcommand output –exit - Exit after first failure (non zero exit)

cp

Copies a file into each repository.

2.2.2 Hooks Releaser

By creating a `.maintain.yml` file in the root of your repository, you can add hooks to various stages of the release process.

```
release:
  hooks:
    bump:
      pre:
        - echo '$VERSION will be bumped'
      post:
        - echo '$VERSION was bumped'
    release:
      pre:
        - echo '$VERSION will be released'
      post:
        - echo '$VERSION was released'
```

Environment Variables

VERSION

VERSION environment variable includes the new version.

2.2.3 git Releaser

Configuration

```
release:
  git:
    commit_format: Release {version}
    tag_format: {version}
```

Detect

Detects Git repositories.

Bump

Commits all of the pending changes for the release.

Release

Tags the commit and pushes the changes.

2.2.4 GitHub Releaser

Configuration

```
release:
  github:
    artefacts:
      - main.js
      - main.min.js
```

Detect

Detects Git repository with an origin of a GitHub repository.

Bump

There is no bump steps for the GitHub releaser.

Release

Creates a GitHub release. You can provide artefacts to be uploaded to the release in the configuration.

GitHub Releaser will detect `CHANGELOG.md` files and pull out the changelog for the current release to be uploaded to the release.

2.2.5 `VERSION` file Releaser

Detect

Detects the `VERSION` file in the root of your library.

Bump

Updates the contents of the `VERSION` file.

Release

There is no release steps for the `VERSION` file releaser.

2.2.6 Changelog Releaser

Changelog releaser allows you to update a semantic changelog.

Configuration

Changelog Releaser allows you to specify the types of changelog sections and their underlying semantic meaning.

```
release:
  changelog:
    sections:
      'Breaking': major
      'Enhancements': minor
      'Bug Fixes': patch
```

Detect

Detects a `CHANGELOG.md` file in the root of your library.

Bump

Finds a “Master” release in your CHANGELOG and changes it to the current version with the current date.

```
## Master

### Bug Fixes

- Adds support for building Swift 2.2.1 from source, and installing 2.2.1
```

(continues on next page)

(continued from previous page)

```
  development snapshots.
- `swiftenv uninstall` will now uninstall Swift toolchains on OS X.
```

Release

There is no release steps for the changelog releaser.

2.2.7 Python Releaser

Detect

The Python releaser looks for a `setup.py` file in the root of your library.

Bump

Bumping will update the version qualifier directly in your `setup.py`.

Release

The release stage of the Python releaser depends on `twine` and `wheel`, which can be installed via pip:

```
$ pip install twine wheel
```

You will need to register your project with PyPI before you can release, which can be done via the following:

```
$ python setup.py register
```

2.2.8 RubyGems Releaser

Detect

Detects a `*.gemspec` file in the root of your library.

Bump

Bumping will update the version specified in your gemspec.

Release

The release stage of the RubyGems releaser will `gem build` and `gem push` your library.

2.2.9 Node Package Manager Releaser

Detect

The NPM releaser will detect a `package.json` file in the root of your library.

Bump

Bumping simply updates the version inside the `package.json` file.

Release

The release stage of the NPM releaser will `npm publish` your library, which depends on NPM being installed.

2.2.10 CocoaPods Releaser

Detect

The CocoaPods releaser is enabled when any files which have the extension `.podspec` or `.podspec.json` are found in the root of the library.

Bump

Bumping is supported for both Ruby podspecs and JSON podspecs. For Ruby podspecs, the version of the specification is bumped directly.

For JSON podspecs, both the version and the git source `tag` of the repository is updated.

Release

The release stage of the CocoaPods releaser depends on the `cocoapods` gem, which can be installed via RubyGems:

```
$ gem install cocoapods
```

You will also need to login to trunk before releasing:

```
$ pod trunk register
```

2.2.11 C/C++ Releaser

Detect

Detects a header file named `version.h` in either `include/<library>/` or `src/`.

Bump

The releaser will update the values of the following `#defines`.

- `VERSION_MAJOR`
- `VERSION_MINOR`
- `VERSION_PATCH`

For example:

```
#define VERSION_MAJOR 1
#define VERSION_MINOR 0
#define VERSION_PATCH 0
```

Release

There is no release steps for this releaser.

2.3 Custom Hooks

By creating a *.maintain.yml* file in the root of your repository, you can add hooks to various stages of the release process.

```
release:
  bump:
    pre:
    - echo 'Hook ran before the version is bumped'
    post:
    - echo 'Hook ran after the version is bumped'
  publish:
    pre:
    - echo 'Hook ran before the library is published'
    post:
    - echo 'Hook ran after the library is published'
```