Bork

Release 8.0.0

Ellen Marie Dash

Feb 26, 2024

CONTENTS

A build and release tool for Python projects, with ZipApp support.

```
usage: bork [-h] [--version] [--verbose] [--debug]
            {aliases,build,clean,download,release,run} ...
```

CHAPTER

ONE

NAMED ARGUMENTS

--version Print version information and exit.

Default: False

--verbose Enable verbose logging.

Default: False

--debug Enable VERY verbose logging. (Sometimes too noisy to be helpful.)

Default: False

CHAPTER

TWO

SUB-COMMANDS

2.1 aliases

Prints the aliases configured via pyproject.toml.

```
bork aliases [-h]
```

2.2 build

Build the project.

```
bork build [-h]
```

2.3 clean

Remove files generated by *bork build*.

```
bork clean [-h]
```

2.4 download

Download a release of the specified project.

```
bork download [-h] [--files FILES] [--directory DIRECTORY] PACKAGE [RELEASE]
```

2.4.1 Positional Arguments

PACKAGE The package to download. Format: *SOURCE:PACKAGE_NAME*, where *PACKAGE_NAME* is the name of the package to download, and *SOURCE* is one of *gh*, *github*, *pypi*, or *testpypi*.

RELEASE The release or tag to download.

Default: "latest"

2.4.2 Named Arguments

--files Comma-separated list of filenames to download.Supports wildcards (* = everything, ? = any single character).

Default: "*.pyz"

--directory Directory to save files in. Created if missing. (Default: *downloads*)

Default: "downloads"

2.5 release

Publish a built project.

```
bork release [-h] [--pypi-repository PYPI_REPOSITORY] [--test-pypi]
             [--dry-run]
```

2.5.1 Named Arguments

--pypi-repository Repository to use. Valid values are pypi, testpypi, or anything defined in '.pypirc'.

Default: "pypi"

--test-pypi Release to test.pypi.org instead of pypi.org. Equivalent to '–pypi-repository testpypi'.

Default: False

--dry-run Don't actually release, just show what a release would do.

Default: False

2.6 run

Run the specified alias.

```
bork run [-h] ALIAS
```

2.6.1 Positional Arguments

ALIAS

CHAPTER

THREE

CONFIGURATION

Bork is configured using pyproject.toml.

3.1 Build System/Backend

You always need to specify the `[build-system]` table. This should be configured according to the build backend you're using.

If you are not using Bork to create ZipApps, this should be the only configuration that Bork requires. The build system (aka "backend") you use will likely want at least `project.name` and `project.version` or equivalent. See "Recommended pyproject.toml Configuration" below.

Python Packaging User Guide: Writing your pyproject.toml explains this well if you're using Hatchling, Setuptools, Flit, or PDM. The information for Hatchling and Setuptool are also included below.

For Setuptools, this looks like:

```
[build-system]
requires = ["setuptools >= 61", "wheel"]
build-backend = "setuptools.build_meta"
```

If you are working on an existing Setuptools project with `setup.py` or `setup.cfg`, creating a `pyproject.toml` file with those contents should be enough to start using Bork and other PEP 517-compliant frontends.

For Hatchling it looks like:

```
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"
```

3.2 Recommended pyproject.toml Configuration

In addition to the `[build-system]` configuration, it is recommended to include at least `project.name` and `project.version` (or equivalent). Some systems (e.g. Poetry) require the values to be specified in a different way.

If you wish to have these values defined in pyproject.toml directly, it will look like this:

```
[project]
name = "some-project-name"
version = "0.0.1"
```

For more advanced configurations, check Python Packaging User Guide: Writing your pyproject.toml or open a GitHub Discussion for Bork.

3.3 ZipApps

To have Bork build a ZipApp, you need the entire recommended configuration described above.

You also need to:

- enable ZipApp functionality
- specify the main function/entrypoint.

For Emanate, which has the ZipApp call the *emanate.cli.main()* function, this looks like:

```
[tool.bork.zipapp]
enabled = true
main = "emanate.cli:main"
```

3.4 Releasing to PyPi and GitHub

Configuring the release process for Bork happens in the `[tool.bork.release]` table in `pyproject.toml`.

As an example, Bork releases itself on PyPi and GitHub, and uses this configuration:

```
[tool.bork.release]
pypi = true
github = true
github_repository = "duckinator/bork"
strip_zipapp_version = true
```

Here's what each option does:

- `pypi`: If true, release to PyPi using the project name reported by the build system.
- `github`: If true, release to GitHub using the specified `github_repository`.
- `github_repository`: The name of the GitHub repository to publish releases to.
- `strip_zipapp_version`: If true, remove the version number from the ZipApp name.

Setting `strip_zipapp_version` to true is recommended, because it means the latest ZipApp is always available at the same URL.

For Bork, this URL is: https://github.com/duckinator/bork/releases/latest/download/bork.pyz

You can provide a template for GitHub Releases by providing a *GitHub Release Template*.

3.5 Aliases

Aliases can be configured in the `[tool.bork.aliases]` section of `pyproject.toml`.

Each alias can be either a string (one command) or a list of strings (multiple commands, ran one at a time, stopping if any of them fail).

An example configuration (if your code is in a directory named `example-project`) might look something like this:

```
[tool.bork.aliases]
lint = [
    "pylint example-project tests",
    "mypy example-project",
]
test = "pytest --verbose"
```

CHAPTER

FOUR

GITHUB RELEASE TEMPLATE

You can create a template for GitHub releases, named `.github/BORK_RELEASE_TEMPLATE.md`.

You can reference a few variables using the syntax `{variable_name}`.

The avialable variables, and what they contain, are:

- `project_name`: The project name, as specified by the build backend.
- `owner`: The part before the `/` in `github_repository` in the *Configuration*.
- `repo`: The part after the `/` in `github_repository` in the *Configuration*.
- `tag`: The tag created by the release. This is `v` followed by the version number.
- `changelog`: A generated changelog based on merged Pull Requests since the last release.

An example template might look like:

```
{project_name} {tag} is now available!

PyPI package: https://pypi.org/project/bork/{version}/
Docker image: https://hub.docker.com/r/duckinator/bork/

The ZipApp is available below, as bork.pyz.

You can help {project_name} by supporting me on [Patreon](https://www.patreon.com/
↪duckinator)!

---

Changes:

{changelog}
```

CHAPTER

FIVE

BORK.API

bork.api.**aliases**()

Returns a list of the aliases defined in pyproject.toml.

bork.api.**build**()

Build the project.

bork.api.**clean**()

Removes artifacts generated by *bork.api.build()*.

(Specifically: *./build*, *./dist*, and any **.egg-info* files.)

bork.api.**download**(*package*, *release_tag*, *file_pattern*, *directory*)

Saves files from the designated package, to the specified directory.

The format for *package* is <*SOURCE*>*:*<*NAME*>, where:

- <*SOURCE*> is one of: *gh*/*github*, *pypi*, or *testpypi*.
- <*NAME*> is the identifier for the package on the specified source.

E.g., Bork would be *gh:duckinator/bork* or *pypi:bork*.

NOTE: *pypi-test* is a deprecated alias of *testpypi*, and will be removed in a future version.

Arguments:

package:
Package to download files for, in the format <*SOURCE*>*:*<*NAME*>.

release_tag:
The version or tag to download.

file_pattern:
Any files matching this pattern will be downloaded.

(Wildcards: * matches anything, *?* matches any single character.)

directory:
The directory where files are saved. This directory is created, if needed.

bork.api.**release**(*repository_name*, *dry_run*)

Uploads build artifacts to a PyPi instance or GitHub, as configured in pyproject.toml.

Arguments:

repository_name:
The name of the PyPi repository. (You probably want 'pypi'.)

dry_run:
If True, don't actually release, just show what a release would do.

bork.api.**run**(*alias*)

Run the alias specified by *alias*, as defined in pyproject.toml.

CHAPTER

SIX

BORK.ASSET_MANAGER

bork.asset_manager.**download_assets**(*asset_list*, *directory*, *name_key=None*, *url_key=None*)

CHAPTER

SEVEN

BORK.BUILDER

bork.builder.**dist**(*backend_settings=None*)

Build the sdist and wheel distributions.

Parameters

backend_settings – Passed as `config_settings` to `build.ProjectBuilder.get_requires_for_build(distribution, config_settings)` and ``build.ProjectBuilder.build(distribution, outdir, config_settings)

bork.builder.**version_from_bdist_file**()

bork.builder.**zipapp**()

Build a zipapp for the project.

dist() should be called before zipapp().

CHAPTER

EIGHT

BORK.CLI

Command-line interface for Bork.

Usage:

bork COMMAND [OPTIONS] [ARGS]

Options that exist for all commands include:

–verbose: enable verbose logging

–debug: enable even more verbose logging (sometimes too noisy to be helpful)

Commands:

bork.cli.**aliases**(*_args*)

bork aliases

Prints a list of aliases (configured via pyproject.toml).

bork.cli.**build**(*_args*)

bork build

Build the project.

bork.cli.**clean**(*_args*)

bork clean

Remove files created by *bork build*.

bork.cli.**download**(*args*)

bork download [–files FILES] [–directory DIRECTORY] PACKAGE RELEASE

Download a release of the specified project.

Arguments:

–files=FILES:
(default *.pyz) A comma-separated list of filenames to download. Supports wildcards (* = everything, ? = any single character).

–directory=DIRECTORY:
(default *downloads*) The directory to save files in. Created if missing.

PACKAGE:
The package to download. Of the format *SOURCE:PACKAGE_NAME*, where *PACKAGE_NAME* is the name of the package to download, and *SOURCE* is one of *gh*, *github*, *pypi*, or *testpypi*.

RELEASE:
The release or tag of the package that you want to download.

bork.cli.**main**(*cmd_args=None*)

Command-line entrypoint for bork.

cmd_args should be either *None* or equivalent to *sys.argv[1:]*.

bork.cli.**release**(*args*)

bork release [–pypi-repository=REPO | –test-pypi] [–dry-run]

Arguments:

–pypi-repository=REPO:
(default *pypi*) Repository to use. Valid values are pypi, testpypi, or anything defined in ".pypirc".

–test-pypi:
Equivalent to *–pypi-repository testpypi*

–dry-run:
Don't actually release, just show what a release would do.

bork.cli.**run**(*args*)

bork run NAME

Run the alias specified by NAME, as defined in pyproject.toml.

bork.cli.**zipapp_main**()

CHAPTER

NINE

BORK.FILESYSTEM

bork.filesystem.**find_files**(*globs*)

bork.filesystem.**load_pyproject**()

Loads the pyproject.toml data.

Will synthesize data if a legacy setuptools project is detected.

bork.filesystem.**try_delete**(*path*)

CHAPTER

TEN

BORK.GITHUB

class bork.github.**GithubConfig**(*token: str*, *repository: str*, *project_name: str*)

Bases: `object`

class bork.github.**GithubRelease**(*config:* GithubConfig, *tag: str*, *commitish: str | None = None*, *body: str | None = None*, *globs=None*, *dry_run=False*, *prerelease=None*, *strip_zipapp_version=False*)

Bases: `object`

prepare()

publish()

static release_template()

bork.github.**download**(*repo*, *release*, *file_pattern*, *directory*)

CHAPTER

ELEVEN

BORK.GITHUB_API

class bork.github_api.**GithubApi**(*owner*, *repo*, *project_name*, *token*)

Bases: object

Basic wrapper for the GitHub API.

Usage:

gh = GithubApi('duckinator', 'bork', '<token>') gh.create_release('TEST-RELEASE', assets={'dist/bork-4.0.5.pyz': 'bork.pyz'})

add_release_asset(*upload_url*, *local_file*, *name*)

changelog()

create_release(*tag_name*, *name=None*, *commitish=None*, *body=None*, *draft=True*, *prerelease=None*, *assets=None*)

tag_name is the name of the tag. *commitish* is a commit hash, branch, tag, etc. *body* is the body of the commit. *draft* indicates whether it should be a draft release or not. *prerelease* indicates whether it should be a prerelease or not. *assets* is a dict mapping local file paths to the uploaded asset name.

property last_release

publish(*release*)

static run(**command*)

CHAPTER

TWELVE

BORK.LOG

bork.log.**logger**(*context: FrameInfo | None = None*) → Logger

Provide a logger with a name appropriate for a given context.

The default context is the caller's.

bork.log.**trace**(*func: F*, *level: int = 10*) → F

Decorator to log function entry and exit.

CHAPTER

THIRTEEN

BORK.PYPI

class bork.pypi.**Downloader**(*repository_name*)

Bases: object

PYPIRC_DEFAULTS = {'pypi': {'_bork-download-endpoint': 'https://pypi.org/simple/'}, 'testpypi': {'_bork-download-endpoint': 'https://test.pypi.org/simple/'}}

PYPIRC_KEY = '_bork-download-endpoint'

download(*package*, *release*, *file_pattern*, *directory*)

bork.pypi.**download**(*repository_name*, *package*, *release*, *file_pattern*, *directory*)

bork.pypi.**upload**(*repository_name*, **globs*, *dry_run=False*)

CHAPTER

FOURTEEN

BORK.PYPI_API

class bork.pypi_api.**SimplePypiParser**(*, *convert_charrefs=True*)

Bases: HTMLParser

current_url = None

error(*message*)

files: Dict[str, str] = {}

handle_data(*data*)

handle_endtag(*tag*)

handle_starttag(*tag*, *attrs*)

in_anchor = False

bork.pypi_api.**get_download_info**(*base_url*, *package*, *release*, *file_pattern*)

CHAPTER

FIFTEEN

BORK.VERSION

PYTHON MODULE INDEX

b

INDEX

M

P

R

S

T

U

V

Z