
tuulbachs

Dave Smith

Aug 23, 2020

CONTENTS:

1	Key Features	3
2	Quickstart	5
2.1	tuulbash	5
2.2	tuulcli	5
2.3	tuuldevops	5
2.4	tuulgit	6
2.5	tuulver	6
2.6	tuulyaml	7
2.7	install	7
2.8	deploy	7
2.9	internal API	8
2.10	Indices and tables	8
	Python Module Index	9
	Index	11

Automate low-level, repetitive, yet important development tasks related to semantic versioning, distributed version control, configuration parsing, and build pipelines.

KEY FEATURES

- Increment [semantic version](#) parts for a versioned entity
- Manage the status of a local Git working tree
- Manage Git tags, local and remote, signed and unsigned
- Parse and update basic YAML configuration files

QUICKSTART

TBD

2.1 tuulbash

kickpy - A bash script intended to kick a Python script from an environment that doesn't have an established Python environment yet. For example:

```
./kickpy.sh example.py
```

2.2 tuulcli

Tuuls for command line interface (CLI).

Define colors and font styles for use in CLI output

```
class tuulcli.cli_color.CliColor
    Contain the list of colors and font styles
```

2.3 tuuldevops

Automation tuuls for common tasks around software development

Provide routines for outputting automated pipeline steps consistently

```
tuuldevops.pipeline_steps.major_step (title, description)
    Output title and description of a major step in the pipeline
```

Git tag the commit on the current branch with the current version of this software

```
tuuldevops.tag_current_version.tag_product_version (conf_filename)
    Git tag the commit on the current branch with the version of this software given in conf_filename
```

Update the version in the tuulbachs-formatted YAML version file

```
tuuldevops.update_version.update_product_version (conf_filename, new_ver)
    Write a new_ver as the new value for the 'version' key in the conf_filename
```

2.4 tuulgit

An opinionated set of Git tuuls.

Check the status of the local git working tree

`tuulgit.check_status.has_staged_uncommitted()`

Return a boolean indicating whether the repository has staged, but uncommitted changes

`tuulgit.check_status.has_unstaged_changes()`

Return a boolean indicating whether the working tree has unstaged changes

`tuulgit.check_status.has_untracked_ignored_files()`

Return a boolean indicating whether the working tree has untracked, ignored files

`tuulgit.check_status.is_clean_working_tree(check_if_working_tree=True)`

Return a boolean indicating whether the Git working tree is clean or not

`tuulgit.check_status.is_working_tree()`

Check if this is a git working tree at all

Raises *TuulError* – when the caller attempts to use this function outside of a git working tree

`tuulgit.check_status.repo_toplevel_path()`

Return a string containing the path of the repo's top-level directory

Git tag the commit on the current branch, *only if* the working tree is clean

`tuulgit.tag_commit.tag_current(tag)`

Git tag (annotated) the current commit from a clean working tree

Raises *TuulError* – when the caller attempts to tag an unclean working tree or to use a tag that already exists on the repo

`tuulgit.tag_commit.tag_current_signed(tag)`

Git tag (signed) the current commit from a clean working tree

Raises *TuulError* – when the caller attempts to tag an unclean working tree or to use a tag that already exists on the repo

`tuulgit.tag_commit.tag_delete_local(tag)`

Delete the named Git tag (local only). This function does *not* delete remote tags

Raises *TuulError* – if the tag delete fails

2.5 tuulver

Parsing tuuls for a tuulbachs-formatted version YAML input file.

Utility functions for managing a tuulbachs-formatted version YAML file

`tuulver.version.bump_build(filename)`

Bump the “build” portion of the version from the input YAML filename

`tuulver.version.bump_major(filename)`

Bump the major portion of the version from the input YAML filename

`tuulver.version.bump_minor(filename)`

Bump the minor portion of the version from the input YAML filename

`tuulver.version.bump_patch(filename)`

Bump the patch portion of the version from the input YAML filename

`tuulver.version.bump_pre(filename, prebase='pre')`
Bump the “pre” portion of the version from the input YAML filename

`tuulver.version.create_version_file(filename, product_name)`
Create an initial tuulbachs-formatted version YAML file

`tuulver.version.emit_product_name(filename)`
Return the product name value from the input YAML filename

`tuulver.version.emit_version(filename)`
Return the version value from the input YAML filename

2.6 tuulyaml

Low level tuuls for interacting with YAML files.

Parse an input YAML file

`tuulyaml.parse.parse_yaml(filename)`
Given input path filename, parse YAML file.

Update a simple top-level value in a YAML file

`tuulyaml.update_simple_value.update_value(inout_path, existing_key, new_value)`
Update existing_key to new_value in the existing inout_path YAML file.

2.7 install

Note that tuulbachs is not yet published at PyPi.

1. Set up and activate a Python [virtual environment](#) at the top level of this project
2. `python -m pip install -r requirements.txt`
3. `cd` to the local `auto` directory
4. `./install_local.sh`

2.8 deploy

How to deploy updates to tuulbachs itself.

Prerequisite: User must have already followed [install](#) guidance at least once in the target environment.

1. Decide which type of [semantic version](#) upgrade this is (major, minor, patch, etc.)
2. From `tuulver/version.py`, use the appropriate `bump_*` function to update the version string in `version.yaml`
3. Follow [install](#) guidance
4. Commit changes to Git
5. From `tuuldevops/tag_current_version.py`, use the `tag_product_version` function to properly tag this release
6. Push the Git update (including tags) to this repo’s remotes

7. In a temp dir, [download all required packages](#) without installing them, tar and zip these for deployment.
8. In the src dir, run `pyinstaller --add-data ../version.yaml:. --onefile tuul.py`
9. Publish the release (including offline packages tarball and `tuul` executable) on the repo's remote (Github, for instance)

2.9 internal API

This is code intended for use by tuulbachs itself, not external users.

Project exception class

exception `tuulbachs.exception.TuulError(msg=None)`
Class used for exceptions thrown by tuulbachs

2.10 Indices and tables

- [genindex](#)
- [modindex](#)

PYTHON MODULE INDEX

t

- `tuulbachs.exception`, 8
- `tuulcli.cli_color`, 5
- `tuuldevops.pipeline_steps`, 5
- `tuuldevops.tag_current_version`, 5
- `tuuldevops.update_version`, 5
- `tuulgit.check_status`, 6
- `tuulgit.tag_commit`, 6
- `tuulver.version`, 6
- `tuulyaml.parse`, 7
- `tuulyaml.update_simple_value`, 7

INDEX

B

`bump_build()` (in module `tuulver.version`), 6
`bump_major()` (in module `tuulver.version`), 6
`bump_minor()` (in module `tuulver.version`), 6
`bump_patch()` (in module `tuulver.version`), 6
`bump_pre()` (in module `tuulver.version`), 7

C

`CliColor` (class in `tuulcli.cli_color`), 5
`create_version_file()` (in module `tuulver.version`), 7

E

`emit_product_name()` (in module `tuulver.version`), 7
`emit_version()` (in module `tuulver.version`), 7

H

`has_staged_uncommitted()` (in module `tuulgit.check_status`), 6
`has_unstaged_changes()` (in module `tuulgit.check_status`), 6
`has_untracked_ignored_files()` (in module `tuulgit.check_status`), 6

I

`is_clean_working_tree()` (in module `tuulgit.check_status`), 6
`is_working_tree()` (in module `tuulgit.check_status`), 6

M

`major_step()` (in module `tuuldevops.pipeline_steps`), 5
module
 `tuulbachs.exception`, 8
 `tuulcli.cli_color`, 5
 `tuuldevops.pipeline_steps`, 5
 `tuuldevops.tag_current_version`, 5
 `tuuldevops.update_version`, 5
 `tuulgit.check_status`, 6

`tuulgit.tag_commit`, 6
 `tuulver.version`, 6
 `tuulyaml.parse`, 7
 `tuulyaml.update_simple_value`, 7

P

`parse_yaml()` (in module `tuulyaml.parse`), 7

R

`repo_toplevel_path()` (in module `tuulgit.check_status`), 6

T

`tag_current()` (in module `tuulgit.tag_commit`), 6
`tag_current_signed()` (in module `tuulgit.tag_commit`), 6
`tag_delete_local()` (in module `tuulgit.tag_commit`), 6
`tag_product_version()` (in module `tuuldevops.tag_current_version`), 5
`tuulbachs.exception`
 module, 8
`tuulcli.cli_color`
 module, 5
`tuuldevops.pipeline_steps`
 module, 5
`tuuldevops.tag_current_version`
 module, 5
`tuuldevops.update_version`
 module, 5
`TuulError`, 8
`tuulgit.check_status`
 module, 6
`tuulgit.tag_commit`
 module, 6
`tuulver.version`
 module, 6
`tuulyaml.parse`
 module, 7
`tuulyaml.update_simple_value`
 module, 7

U

`update_product_version()` (*in module tuuldevops.update_version*), [5](#)

`update_value()` (*in module tuulyaml.update_simple_value*), [7](#)