Stitches

May 10, 2019

Contents

Stitches is a task runner for GRASS GIS, an alternative to running BASH and Python scripts with Grass's `--exec` option.

CHAPTER 1

Features

- Session support: no need to start GRASS GIS before running any tasks.
- *Caching*: task state is tracked to skip tasks when possible to do so.
- Composability: tasks may be organised into pipelines and used as tasks.
- Pipelines may be called with custom variables and use Jinja2 in their definitions for more generic data processing.
- Custom tasks may be written as simple python functions.

1.1 Installation

Stitches works on Python 2.7 and Python 3.7 or later with GRASS GIS 7.4+. It is currently only *tested* on Linux (other platforms may follow).

1.1.1 Pip

```
$ pip install stitches-gis
```

1.1.2 Git

```
$ git@github.com:davebrent/stitches.git
$ cd stitches
$ python setup.py install
```

1.2 Quickstart

Once stitches is installed, the `stitches` command should become available in your `$PATH`.

Create a simple pipeline file

Save this file as `pipeline.toml` (or any name you like).

Then run the pipeline with stitches in verbose mode

```
$ stitches --verbose pipeline.toml
```

This should print the following to the console

```
[0]: Hello world
  Completed
```

Please see the examples folder for more advanced uses of pipelines.

1.3 Usage

```
Stitches.

Usage:
  stitches [--gisdbase=<path>] [--location=<name>] [--mapset=<name>]
           [[--skip=<task>]... [--force] | --only=<task>]
           [--log=<path>] [--verbose] [--nocolor]
           [--vars=<vars>] <pipeline>

Options:
  -h --help             Show this screen.
  -v --verbose          Show more output.
  --log=<path>          Task log output path.
  --nocolor             Disable colorized output.
  --gisdbase=<path>     Initial GRASS GIS database directory.
  --location=<name>     Initial GRASS location.
  --mapset=<name>       Initial GRASS Mapset.
  --skip=<task>         Comma-separated list of tasks to skip.
  --only=<task>         Run a single task.
  --force               Force all tasks to run.
  --vars=<vars>         Initial pipeline variables.
```

Run a pipeline with custom variables

```
$ stitches --vars="foo='hello' bar='world'" pipeline.toml
```

Skip the 2nd and 4th tasks in a pipeline

```
$ stitches --skip=1,3 pipeline.toml
```

1.4 Concepts

1.4.1 Pipeline

A pipeline is a Jinja2 template file, that renders a TOML file, containing a list of *Task* definitions, to be executed sequentially.

Although there is no hard restriction, it is expected that a pipeline be run multiple times (such as during development) so it is suggested that they be indempotent with respect to its inputs and outputs.

A pipeline may declare the GRASS GIS database, location and mapset that it should be run against, or these values may be passed in via the command line.

1.4.2 Task

A task may consist of one of the following:

- One of the provided *Built-in Tasks*.
- Another pipeline.
- An importable python callable, in the form of `importable.module:function`. The referenced function is called with the task definition's `params` field as keyword arguments.

1.4.3 Resource

Resources may consist of GRASS GIS maps or regular files, their references should follow the format `<type>/(<filepath> | <grassref>)`. Examples of valid references:

```
'file/foobar/baz.tif'                   # Relative path
'file//foobar/baz.tif'                  # Absolute path
'vector/map@gisdbase/location/mapset'   # Map in specific database
'vector/map@location/mapset'            # Map in a specific location
'vector/map@mapset'                     # Map in a specific mapset
'vector/map'                            # Map in this mapset
```

Its recommended to reference the resources used by a task to make the most of *Caching*.

1.4.4 Caching

The current state of resources used in a pipeline is tracked. If the following conditions are met the task will be skipped:

- The task is executed in the same *region* as its previous execution.
- The tasks `params` are unchanged.
- No input files have been modified.
- Tasks that created any input maps were also skipped.
- Its output resources already exist.

A task will not be skipped if it is not possible for stitches to track the creation of any mapset used by the task.

1.4.5 State

The state of the initial pipeline's execution is stored in a file called `stitches.state.json` in the pipeline's *initial* mapset. This may lead to unexpected results when running different *initial* pipelines against the same mapset.

1.4.6 Errors & Logging

In the event that a task raises an exception, the output of all tasks, including GRASS GIS output, is automatically written to file for inspection. This log may be written to a specified location and will always be outputted using the `--log` option.

1.5 Reference

1.5.1 Toml configuration options

Pipeline

Property	Type	Description
`gisdbase`	str	Initial grass database directory.
`location`	str	Initial grass location.
`mapset`	str	Initial grass mapset (default: `'PERMANENT'`).
`tasks`	List[*Task*]	Tasks to run against the mapset.

Task

Property	Type	Description
`message`	str	Text to display when the task is run.
`pipeline`	str	Path to a pipeline file.
`task`	str	Built-in task name (see *Built-in Tasks*) or a reference to an importable python function eg. `package.module:function`.
`inputs`	List[str]	List of input resources.
`outputs`	List[str]	List of output resources.
`removes`	List[str]	List of resources removed by the task.
`always`	bool	Option to always run the task/pipeline.
`params`	dict	Task/pipeline keyword arguments.

- Either `pipeline` or `task` must be defined.

Pipeline task `params`

Property	Type	Description
`gisdbase`	str	Grass database directory (not implemented).
`location`	str	Grass location (not implemented).
`mapset`	str	Grass mapset (not implemented).
`vars`	dict	Variables passed into the pipeline.

- Switching database, location and mapset automatically, when calling another pipeline, is not yet implemented.

1.5.2 Built-in Tasks

grass (*module=None*, ***kwargs*)
Run a GRASS GIS command.

Please refer to the relevant version of documentation for `grass.pygrass.modules.Module` for more information.

Keyword Arguments

- **module** (*str*) – GRASS GIS command name
- ****kwargs** – Keyword arguments passed to `grass.pygrass.modules.Module`

script (*cmd=None*)
Run an arbitrary shell command.

Keyword Arguments **cmd** (*list*) – A sequence of program arguments

1.6 Contribute

- Issue Tracker
- Source Code
- Documentation

Python Module Index

S

Index

G

S