tmuxp Documentation

0.1-dev

Tony Narlock

2014 01 15

Contents

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

tmuxptmux python tmux(1) (>= 1.8) tmux

-
- teamocil tmuxinator
- JSONYAML
- bash, zsh tcsh
- tmux(1.8git) travis.yml Travis CItmuxp
- , , “_
-

Explore:

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 1

(About)

tmuxp

tmuxptmux(ORM) YAML, JSON `dict` tmuxinator teamocil

(Quickstart) (Examples)

API Reference (Developing and Testing)

BSD-licensed http://github.com/tony/tmuxp.

1.1 tmuxinator / teamocil (Differences from tmuxinator / teamocil)

: teamocil / tmuxinator github

1.1.1 (Similarities)

tmux(session)

YAML YAML

1.1.2 (Missing)

Stability tmuxinator teamocil tmuxp

ERB / teamocil ERB

tmuxp `tmux >= 1.8` Teamocil tmuxinator

1.1.3 (Differences)

tmuxp python teamocil tmuxinator ruby

Workspace building process teamocil tmuxinator shell tmuxp ORM

1.2 (Additional Features)

CLI tmuxptab(session) *(Other commands)*

Import config Teamocil / Tmuxinator [1] *(Import)*

(session)YAMLJSON [1] *(Freeze sessions)*

JSON JSON *(Examples)*

ORMAPI - tmux(>=1.8) ID(pane)(windows)(session) tumx `Server`, `Session`, `Window`, `Pane` *(Internals)*

`$ tmuxp convert <filename>` JSON/YAML

1.3 (Minor tweaks)

- tmuxtmux(session)(window)(pane) *Travis CI*
- tmux(session)
- (session)
- virtualenv / rvm /
- `$ tmuxp load /full/file/path.json`
- `$ tmuxp load . .tmuxp.yaml .tmuxp.json`
- `$ tmuxp -2`, `$ tmuxp -8` *tmux(1)*
- `$ tmuxp -L<socket-name>`, `$ tmuxp -S<socket-path>` socket `$ tmuxp -f<config-file>`

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

[1] -

CHAPTER 2

(Command Line Interface)

2.1 (Freeze sessions)

tmux(session)

Tmuxp (session) `.json` `.yaml`

```
usage: tmuxp freeze [-h] [-L socket-name] [-S socket-path]
                    session_name [session_name ...]
```

Positional arguments:

session_name	(Session)

Options:

-L	tmuxsockettmux
-S	tmuxsockettmux

2.2 (Load session)

`$HOME/.tmuxp` *Bash(Bash completion)*

```
$ tmuxp load <filename>
```

`.tmuxp.yaml` `.tmuxp.json`:

```
$ tmuxp load .
```

```
usage: tmuxp load [-h] [-L socket-name] [-S socket-path] [-2 | -8] [--list]
                  [config]
```

Positional arguments:

config	

Options:

-L	tmuxsockettmux
-S	tmuxsockettmux
-2	tmux256

-8 -288

--list=False

2.3 (Import)

2.3.1 teamocil(From teamocil)

```
usage: tmuxp import teamocil [-h] (--list | config)
```

Positional arguments:

config ~/.teamocil yaml

Options:

--list=False ~/.teamocil

2.3.2 tmuxinator(From tmuxinator)

```
usage: tmuxp import tmuxinator [-h] (--list | config)
```

Positional arguments:

config ~/.tmuxinator yaml

Options:

--list=False ~/.tmuxinator

2.4 YAMLJSON(Convert between YAML and JSON)

tmuxp `.yaml .json .json .yaml`

```
usage: tmuxp convert [-h] config
```

Positional arguments:

config /

2.5 Bash(Bash completion)

bash `.bashrc`:

```
$ source tmuxp.bash
```

tcsh `.tcshrc`:

```
$ complete tmuxp 'p/*/`tmuxp.tcsh`/'
```

zsh `.zshrc`:

```
$ source tmuxp.zsh
```

2.6 (Other commands)

```
usage: tmuxp kill-session [-h] [-L socket-name] [-S socket-path]
                          session_name [session_name ...]
```

Positional arguments:

session_name	(session)

Options:

-L	tmuxsockettmux
-S	tmuxsockettmux

```
usage: tmuxp attach-session [-h] [-L socket-name] [-S socket-path] [-2 | -8]
                            session_name [session_name ...]
```

Positional arguments:

session_name	(session)

Options:

-L	tmuxsockettmux
-S	tmuxsockettmux
-2	tmux256
-8	-288

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 3

(Quickstart)

3.1 (CLI)

:

(Examples), *(Command Line Interface)*, *Bash(Bash completion)*.

tmuxp(session)

‘‘$HOME/.tmuxp‘‘ `.tmuxp.py` `.tmuxp.json` `.tmuxp.yaml`

:

1. `session_name`
2. `windows`
3. `windows` ‘‘panes‘‘

`~/.tmuxp/example.yaml`

```
session_name: 2-pane-vertical
windows:
  - window_name: my test window
    panes:
      - pwd
      - pwd
```

tmuxp:

```
$ tmuxp load -l
```

`$HOME/.tmuxp` tmuxp `example.yaml`

```
$ tmuxp load example.yaml
```

tmuxp(session)

3.2 Pythonics

:

tmuxp python API *(Developing and Testing)* *(Internals)*.

ORM - (Object Relational Mapping)

AL - (Abstration Layer)

3.2.1 python

tmuxp python api	*tmux(1)*
`Server.new_session()`	`$ tmux new-session`
`Server.list_sessions()`	`$ tmux list-sessions`
`Session.list_windows()`	`$ tmux list-windows`
`Session.new_window()`	`$ tmux new-window`
`Window.list_panes()`	`$ tmux list-panes`
`Window.split_window()`	`$ tmux split-window`
`Pane.send_keys()`	`$ tmux send-keys`

3.2.2 tmux ORM

tmuxptmux(SQLAlchemy engine)(session)

- `Server` `Session`
- `Session` `Window`
- `Window` `Pane`

tmux tmux 1.8's `pane_id`, `window_id` `session_id` python

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 4

(Examples)

4.1 (Short hand / inline)

tmuxp

short hand

```
did you know
you can inline
```

```
single commands
```

```
for panes
```

4.1.1 YAML

```
session_name: shorthands
windows:
  - window_name: long form
    panes:
    - shell_command:
      - echo 'did you know'
      - echo 'you can inline'
    - shell_command: echo 'single commands'
    - echo 'for panes'
```

4.1.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "echo 'did you know'",
            "echo 'you can inline'"
          ]
```

```
        },
        {
          "shell_command": "echo 'single commands'"
        },
        "echo 'for panes'"
      ],
      "window_name": "long form"
    }
  ],
  "session_name": "shorthands"
}
```

4.2 (Blank panes)

```
pwd null,'blank','pane' ''
```

4.2.1 YAML

```
session_name: Blank pane test
windows:
  # Emptiness will simply open a blank pane, if no shell_command_before.
  # All these are equivalent
  - window_name: Blank pane test
    layout: tiled
    panes:
    -
    - pane
    - blank
    - null
    - shell_command:
    - shell_command:
      -
  # an empty string will be treated as a carriage return
  - window_name: Empty string (return)
    panes:
    - ''
    - shell_command: ''
    - shell_command:
      - ''
```

4.2.2 JSON

```
{
  "windows": [
    {
      "panes": [
        null,
        "pane",
        "blank",
        null,
        {
          "shell_command": null
        },
```

```
        {
          "shell_command": [
            null
          ]
        }
      ],
      "layout": "tiled",
      "window_name": "Blank pane test"
    },
    {
      "panes": [
        "",
        {
          "shell_command": ""
        },
        {
          "shell_command": [
            ""
          ]
        }
      ],
      "window_name": "Empty string (return)"
    }
  ],
  "session_name": "Blank pane test"
}
```


4.3 (Start Directory)

`tmux new-window -c <start-directory>`.

4.3.1 YAML

```
session_name: start directory
start_directory: /var/
windows:
  - window_name: should be /var/
    panes:
      - shell_command:
        - echo "\033c
        - it trickles down from session-level"
      - pwd
  - window_name: should be /var/log
    start_directory: log
    panes:
      - shell_command:
        - echo '\033c
        - window start_directory concatenates to session start_directory
        - if it is not absolute'
      - pwd
  - window_name: should be ~
    start_directory: '~'
    panes:
    - shell_command:
```

```
      - 'echo \\033c ~ has precedence. note: remember to quote ~ in YAML'
    - pwd
  - window_name: should be /proc
    start_directory: /proc
    panes:
    - echo '\033c absolute paths also have precedence.'
    - pwd
  - window_name: should be config's dir
    start_directory: ./
    panes:
      - shell_command:
        - echo '\033c
        - ./ is relative to config file location
        - ../ will be parent of config file
        - ./test will be \"test\" dir inside dir of config file'
      - shell_command:
        - echo '\033c
        - This way you can load up workspaces from projects and maintain
        - relative paths.'
```

4.3.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "echo \"\\033c",
            "it trickles down from session-level\""
          ]
        },
        "pwd"
      ],
      "window_name": "should be /var/"
    },
    {
      "panes": [
        {
          "shell_command": [
            "echo '\\033c",
            "window start_directory concatenates to session start_directory",
            "if it is not absolute'"
          ]
        },
        "pwd"
      ],
      "start_directory": "log",
      "window_name": "should be /var/log"
    },
    {
      "panes": [
        {
          "shell_command": [
            "echo \\\\033c ~ has precedence. note: remember to quote ~ in YAML"
          ]
```

```
        },
        "pwd"
      ],
      "start_directory": "~",
      "window_name": "should be ~"
    },
    {
      "panes": [
        "echo '\\033c absolute paths also have precedence.'",
        "pwd"
      ],
      "start_directory": "/proc",
      "window_name": "should be /proc"
    },
    {
      "panes": [
        {
          "shell_command": [
            "echo '\\033c",
            "./ is relative to config file location",
            "../ will be parent of config file",
            "./test will be \\\"test\\\" dir inside dir of config file'"
          ]
        },
        {
          "shell_command": [
            "echo '\\033c",
            "This way you can load up workspaces from projects and maintain",
            "relative paths.'"
          ]
        }
      ],
      "start_directory": "./",
      "window_name": "should be config's dir"
    }
  ],
  "session_name": "start directory",
  "start_directory": "/var/"
}
```

4.4 (2 split panes)

2 pane

4.4.1 YAML - (Short form)

```
session_name: 2-pane-vertical
windows:
  - window_name: my test window
    panes:
      - pwd
      - pwd
```

4.4.2 JSON - (Short form)

```
{
  "windows": [
    {
      "panes": [
        "pwd",
        "pwd"
      ],
      "window_name": "my test window"
    }
  ],
  "session_name": "2-pane-vertical"
}
```

4.4.3 YAML - (Christmas Tree)

```
session_name: 2-pane-vertical-long
windows:
  - window_name: test
    panes:
    - shell_command:
      - cd ~
      - pwd
      - top
    - shell_command:
      - cd /var/www
      - pwd
  - window_name: second window
    shell_command_before:  cd /var/www
    panes:
    - shell_command: pwd
    - shell_command:
      - pwd
```

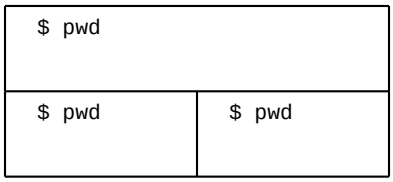
4.4.4 JSON - (Christmas Tree)

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "cd ~",
```

```
            "pwd",
            "top"
          ]
        },
        {
          "shell_command": [
            "cd /var/www",
            "pwd"
          ]
        }
      ],
      "window_name": "test"
    },
    {
      "panes": [
        {
          "shell_command": "pwd"
        },
        {
          "shell_command": [
            "pwd"
          ]
        }
      ],
      "shell_command_before": "cd /var/www",
      "window_name": "second window"
    }
  ],
  "session_name": "2-pane-vertical-long"
}
```

4.5 (3 panes)

3 panes



4.5.1 YAML

```
session_name: 3-panes
windows:
- window_name: dev window
  layout: main-vertical
  shell_command_before:
    - cd ~/
  panes:
    - shell_command:
```

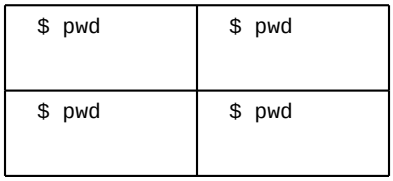
```
        - cd /var/log
        - ls -al | grep \.log
    - pwd
    - pwd
```

4.5.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "cd /var/log",
            "ls -al | grep \\.log"
          ]
        },
        "pwd",
        "pwd"
      ],
      "shell_command_before": [
        "cd ~/"
      ],
      "layout": "main-vertical",
      "window_name": "dev window"
    }
  ],
  "session_name": "3-panes"
}
```

4.6 (4 panes)

4 panes



4.6.1 YAML

```
session_name: 4-pane-split
windows:
- window_name: dev window
  layout: tiled
  shell_command_before:
    - cd ~/
  panes:
```

```
    - shell_command:
        - cd /var/log
        - ls -al | grep \.log
    - pwd
    - pwd
    - pwd
```

4.6.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "cd /var/log",
            "ls -al | grep \\.log"
          ]
        },
        "pwd",
        "pwd",
        "pwd"
      ],
      "shell_command_before": [
        "cd ~/"
      ],
      "layout": "tiled",
      "window_name": "dev window"
    }
  ],
  "session_name": "4-pane-split"
}
```

4.7 (Automatic Rename)

4.7.1 YAML

```
session_name: test window options
start_directory: '~'
windows:
- layout: main-horizontal
  options:
    automatic-rename: on
  panes:
  - shell_command:
    - man echo
    start_directory: '~'
  - shell_command:
    - echo "hey"
  - shell_command:
    - echo "moo"
```

4.7.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "man echo"
          ],
          "start_directory": "~"
        },
        {
          "shell_command": [
            "echo \"hey\""
          ]
        },
        {
          "shell_command": [
            "echo \"moo\""
          ]
        }
      ],
      "layout": "main-horizontal",
      "options": {
        "automatic-rename": true
      }
    }
  ],
  "session_name": "test window options",
  "start_directory": "~"
}
```

4.8 (Main pane height)

4.8.1 YAML

```
session_name: main-pane-height
start_directory: '~'
windows:
- layout: main-horizontal
  options:
    main-pane-height: 30
  panes:
  - shell_command:
    - top
    start_directory: '~'
  - shell_command:
    - echo "hey"
  - shell_command:
    - echo "moo"
  window_name: my window name
```

4.8.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "top"
          ],
          "start_directory": "~"
        },
        {
          "shell_command": [
            "echo \"hey\""
          ]
        },
        {
          "shell_command": [
            "echo \"moo\""
          ]
        }
      ],
      "layout": "main-horizontal",
      "options": {
        "main-pane-height": 30
      },
      "window_name": "editor"
    }
  ],
  "session_name": "main pane height",
  "start_directory": "~"
}
```

4.9 (Super-advanced dev environment)

:

(Developing and Testing) tmuxp(tmuxp developer config)

4.9.1 YAML

```
session_name: tmuxp
start_directory: ./ # load session relative to config location (project root).
windows:
- window_name: tmuxp
  layout: main-horizontal
  options:
    main-pane-height: 35
  shell_command_before:
    - command -v virtualenv >/dev/null 2>&1 || { pip install virtualenv; }
    - '[ -d .env -a -f .env/bin/activate ] && source .env/bin/activate || virtualenv .env'
    - '[ ! -d .env/build ] || rm -rf .env/build'
  panes:
  - shell_command:
```

```
    - reset
    - vim
    - :Ex
    focus: true
  - shell_command:
    - echo hi
  - shell_command:
    - command -v .env/bin/tmuxp >/dev/null 2>&1 || { pip install -e .; }
    - command -v watching_testrunner >/dev/null 2>&1 || { pip install watching_testrunner; }
    - watching_testrunner --basepath ./ --pattern="*.py" 'python run_tests.py'
- window_name: docs
  layout: main-horizontal
  options:
    main-pane-height: 35
  shell_command_before:
    - command -v virtualenv >/dev/null 2>&1 || { pip install virtualenv; }
    - '[ -d .env -a -f .env/bin/activate ] && source .env/bin/activate || virtualenv .env'
    - '[ ! -d .env/build ] || rm -rf .env/build'
    - command -v .env/bin/tmuxp >/dev/null 2>&1 || { pip install -e .; }
    - cd ./doc
  panes:
  - shell_command:
    - reset
    - vim
    - :Ex
    focus: true
  - pwd
  - echo 'docs built to <http://0.0.0.0:8000/_build/html>'; python -m SimpleHTTPServer
  - shell_command:
    - command -v sphinx-quickstart >/dev/null 2>&1 || { pip install -r requirements.pip; }
    - command -v watching_testrunner >/dev/null 2>&1 || { pip install watching_testrunner; }
    - watching_testrunner --basepath ./ --pattern="*.rst" 'make html'
    - python -m SimpleHTTPServer
```

4.9.2 JSON

```
{
  "windows": [
    {
      "panes": [
        {
          "shell_command": [
            "reset",
            "vim",
            ":Ex"
          ],
          "focus": true
        },
        {
          "shell_command": [
            "echo hi"
          ]
        },
        {
          "shell_command": [
            "command -v .env/bin/tmuxp >/dev/null 2>&1 || { pip install -e .; }",
            "command -v watching_testrunner >/dev/null 2>&1 || { pip install watching_testrunner; }",
```

```
            "watching_testrunner --basepath ./ --pattern=\"*.py\" 'python run_tests.py'"
          ]
        }
      ],
      "shell_command_before": [
        "command -v virtualenv >/dev/null 2>&1 || { pip install virtualenv; }",
        "[ -d .env -a -f .env/bin/activate ] && source .env/bin/activate || virtualenv .env",
        "[ ! -d .env/build ] || rm -rf .env/build"
      ],
      "layout": "main-horizontal",
      "window_name": "tmuxp",
      "options": {
        "main-pane-height": 35
      }
    },
    {
      "panes": [
        {
          "shell_command": [
            "reset",
            "vim",
            ":Ex"
          ],
          "focus": true
        },
        "pwd",
        "echo 'docs built to <http://0.0.0.0:8000/_build/html>'; python -m SimpleHTTPServer",
        {
          "shell_command": [
            "command -v sphinx-quickstart >/dev/null 2>&1 || { pip install -r requirements.pip; }",
            "command -v watching_testrunner >/dev/null 2>&1 || { pip install watching_testrunner; }",
            "watching_testrunner --basepath ./ --pattern=\"*.rst\" 'make html'",
            "python -m SimpleHTTPServer"
          ]
        }
      ],
      "shell_command_before": [
        "command -v virtualenv >/dev/null 2>&1 || { pip install virtualenv; }",
        "[ -d .env -a -f .env/bin/activate ] && source .env/bin/activate || virtualenv .env",
        "[ ! -d .env/build ] || rm -rf .env/build",
        "command -v .env/bin/tmuxp >/dev/null 2>&1 || { pip install -e .; }",
        "cd ./doc"
      ],
      "layout": "main-horizontal",
      "window_name": "docs",
      "options": {
        "main-pane-height": 35
      }
    }
  ],
  "session_name": "tmuxp",
  "start_directory": "./"
}
```

4.10 (Kung fu)

: tmuxp(session)python *API* *Reference* `tmuxp.WorkspaceBuilder` `dict` `tmuxp.config.validate_schema()`

---

tmuxp github

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 5

(Internals)

:

API Reference

tmuxptmux

util.TmuxRelationalObject Server, Session, Window Pane

(object)	(child)	(parent)
Server	Session	None
Session	Window	Server
Window	Pane	Session
Pane	None	Window

tmux(server)(server)socket (server)tmux(server)

(server)(session) Ctrl-a s (session)

(session)(window)(pane)(session_id, window_id, pane_id) util.TmuxMappingObject Server

(Object)	(Prefix)	(Example)
Server	N/A	N/A, uses socket-name and socket-path
Session	$	$13
Window	@	@3243
Pane	%	%5433

5.1 TmuxPythonics(Similarities to Tmux and Pythonics)

tmuxpythonAPI tmuxptmuxpython AP

tmuxptmux FORMATTERS Session, Window Pane

Tmuxp(pane)(window)(session)

Tmux(session)(window)(pane)ID

(pane) % %1234

(window) @ @2345

(session) $, $

Tmuxp(window)

Tmux window_id

{pane,window}_index VS a {pane,window,session}_id ?

Pane

window#

(pane),(window),(session) Tmuxp `Server.list_sessions()`, `Session.list_windows()`, `Window.list_panes()`

5.2 (Idiosyncrasies)

Tmuxppython `new-window` (-)(_)

5.3 (Reference)

- tmux http://www.openbsd.org/cgi-bin/man.cgi?query=tmux&sektion=1
- tmux http://sourceforge.net/p/tmux/tmux-code/ci/master/tree/

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 6

(Developing and Testing)

./tmuxp/testsuite unittest

./run_tests.py $ tmux -L test_case socket_name tmux(server)

6.1 git(Install the latest code from git)

git:

```
$ git clone git@github.com:tony/tmuxp.git
$ cd tmuxp
```

virtualenvvirtualenv:

```
$ virtualenv .env
```

(tty/terminal)virtualenv:

```
$ source .env/bin/activate
```

tmuxp:

```
$ pip install -e .
```

pippython -e --editable,

```
$ tmuxp
```

6.2 (Test Runner)

python *PATH* .env python tmuxp

:

```
$ ./run_tests.py
```

github(issue on github)

6.2.1 (Test runner options)

: v0.0.20, `--tests tmuxp.testsuite`

```
$ ./run_tests.py --tests test_config.ImportExportTest
```

:

```
$ ./run_tests.py --tests tmuxp.testsuite.test_config.ImportExportTest
```

```
$ ./run_tests.py --help
```

`test_config`:

`unittest.TestSuite`:

```
$ ./run_tests.py test_config
```

`unittest.TestCase`:

```
$ ./run_tests.py --tests test_config.ImportExportTest
```

:

```
$ ./run_tests.py --tests test_config.ImportExportTest.test_export_json
```

:

```
$ ./run_tests.py --tests ImportExportTest.test_export_json \
    ImportExportTest.test_window
```

(Visual testing)

tmux(session)

:

- 1: `$ tmux -L test_case`
- 2: `$ cd` tmuxp `virtualenv` (tmuxp):

  ```
  $ python ./run_tests.py --tests tests_workspacebuilder
  ```

1 (session) tmuxp

(Verbosity and logging)

`./run_tests.py` supports two options, these are *optional* flags that may be added to for *(Test runner options)* and *(Visual testing)*. `./run_tests.py` *(Test runner options)* *(Visual testing)*

1. : `-l --log-level`, `debug`, `info`, `warn`, `error`, `fatal`. `INFO`

   ```
   $ ./run_tests.py --log-level debug
   ```

   :

   ```
   $ ./run_tests.py -l debug
   ```

2. :

   --verbosity 0, 1 2. : 2.

   ```
   $ ./run_tests.py --verbosity 0
   ```

6.3 (Run tests on save)

Tmux

: pypi watching_testrunner

pypi watching_testrunner:

```
$ pip install watching_testrunner
```

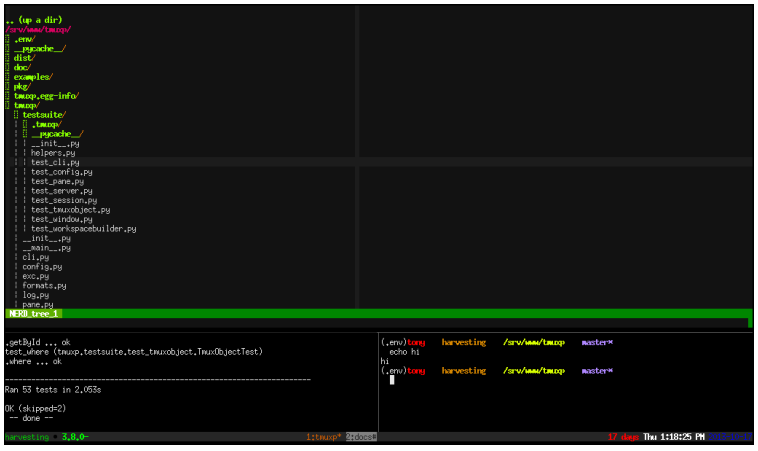
.py :

```
$ watching_testrunner --basepath ./ --pattern="*.py" python run_tests.py
```

(Visual testing) :

```
$ watching_testrunner --basepath ./ --pattern="*.py" python run_tests.py --visual
```

6.3.1 tmuxp(tmuxp developer config)



git(Install the latest code from git) tmxp:

```
$ tmuxp load .
```

.tmuxp.yaml

```
session_name: tmuxp
start_directory: ./ # load session relative to config location (project root).
windows:
- window_name: tmuxp
  layout: main-horizontal
  options:
    main-pane-height: 35
  shell_command_before:
```

```
    - command -v virtualenv >/dev/null 2>&1 || { pip install virtualenv; }
    - '[ -d .env -a -f .env/bin/activate ] && source .env/bin/activate || virtualenv .env'
    - '[ ! -d .env/build ] || rm -rf .env/build'
  panes:
  - shell_command:
    - reset
    - vim
    - :Ex
    focus: true
  - shell_command:
    - echo hi
  - shell_command:
    - command -v .env/bin/tmuxp >/dev/null 2>&1 || { pip install -e .; }
    - command -v watching_testrunner >/dev/null 2>&1 || { pip install watching_testrunner; }
    - watching_testrunner --basepath ./ --pattern="*.py" 'python run_tests.py'
- window_name: docs
  layout: main-horizontal
  options:
    main-pane-height: 35
  shell_command_before:
    - command -v virtualenv >/dev/null 2>&1 || { pip install virtualenv; }
    - '[ -d .env -a -f .env/bin/activate ] && source .env/bin/activate || virtualenv .env'
    - '[ ! -d .env/build ] || rm -rf .env/build'
    - command -v .env/bin/tmuxp >/dev/null 2>&1 || { pip install -e .; }
    - cd ./doc
  panes:
  - shell_command:
    - reset
    - vim
    - :Ex
    focus: true
  - pwd
  - echo 'docs built to <http://0.0.0.0:8000/_build/html>'; python -m SimpleHTTPServer
  - shell_command:
    - command -v sphinx-quickstart >/dev/null 2>&1 || { pip install -r requirements.pip; }
    - command -v watching_testrunner >/dev/null 2>&1 || { pip install watching_testrunner; }
    - watching_testrunner --basepath ./ --pattern="*.rst" 'make html'
    - python -m SimpleHTTPServer
```

6.3.2 Travis CI

Tmuxp travis-ci /

travistmuxptmux1.8python2.7 travistmuxp .travis.yml :

```
language: python

python:
  - 2.6
  - 2.7
  - 3.3
matrix:
  allow_failures:
    - python: 2.6
env:
  - TMUX_VERSION=master
  - TMUX_VERSION=1.8
```

```
before_install:
 - sudo apt-get update -qq
install:
  - "pip install -e ."
before_script:
  - sudo apt-get install -qq libevent-dev libncurses-dev
  - git clone git://git.code.sf.net/p/tmux/tmux-code tmux
  - cd tmux
  - git checkout $TMUX_VERSION
  - sh autogen.sh
  - ./configure && make && sudo make install
  - cd ..
  - tmux -V
#script: python run_tests.py
script: python setup.py test
```

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 7

API Reference

7.1 Server Object

class tmuxp.**Server**(*socket_name=None*, *socket_path=None*, *config_file=None*, *colors=None*, ***kwargs*)

tmuxp.util.TmuxRelationalObject

The *tmux(1)* server.

- Server._sessions [Session, ...]
  - Session._windows [Window, ...]
    - Window._panes [Pane, ...]
      - Pane

When instantiated, provides the t global. stores information on live, running tmux server.

childIdAttribute = 'session_id'
unique child ID key

socket_name = None
[-L socket-name]

socket_path = None
[-S socket-path]

config_file = None
[-f file]

colors = None
-2 or -8

tmux(**args*, ***kwargs*)
Return util.tmux send tmux commands with sockets, colors.

util.tmux

_list_sessions()
Return list of sessions in dict form.

Retrieved from $ tmux(1) list-sessions stdout.

The list is derived from stdout in util.tmux which wraps Subprocess.Popen().

list of dict

_sessions
Property / alias to return `_list_sessions()`.

list_sessions()
Return list of `Session` from the `tmux(1)` session.

`list` of `Session`

sessions
Property / alias to return `list_sessions()`.

children
Alias of `sessions`.

_list_windows()
Return list of windows in `dict` form.

Retrieved from `$ tmux(1) list-windows` stdout.

The `list` is derived from `stdout` in `util.tmux` which wraps `Subprocess.Popen()`.

list

_update_windows()
Update internal window data and return `self` for chainability.

`Server`

_list_panes()
Return list of panes in `dict` form.

Retrieved from `$ tmux(1) list-panes` stdout.

The `list` is derived from `stdout` in `util.tmux` which wraps `Subprocess.Popen()`.

list

_update_panes()
Update internal pane data and return `self` for chainability.

`Server`

attached_sessions()
Return active `Session` object.

This will not work where multiple tmux sessions are attached.

`Server`

has_session(*target_session*)
Return True if session exists. `$ tmux has-session`.

Param target_session: str of session name.

bool

kill_server()
`$ tmux kill-server`.

findWhere(*attrs*)
Return object on first match.

Based on .findWhere() from underscore.js.

getById(*id*)
Return object based on .get() from backbone.js.

id (*string*) –

object

kill_session(*target_session=None*)
Kill the tmux session with `$ tmux kill-session`, return `self`.

Param target_session: str. note this accepts `fnmatch(3)`. 'asdf' will kill 'asdfasd'.

`Server`

where(*attrs*, *first=False*)
Return objects matching child objects properties.

Based on .where() from underscore.js.

attrs (*dict*) – tmux properties to match

list

switch_client(*target_session*)
`$ tmux switch-client`.

Param target_session: str. name of the session. fnmatch(3) works.

attach_session(*target_session=None*)
`$ tmux attach-session` aka alias: `$ tmux attach`.

Param target_session: str. name of the session. fnmatch(3) works.

new_session(*session_name=None*, *kill_session=False*, *attach=False*, **args*, ***kwargs*)
Return `Session` from `$ tmux new-session`.

Uses `-P` flag to print session info, `-F` for return formatting returns new Session object.

`$ tmux new-session -d` will create the session in the background `$ tmux new-session -Ad` will move to the session name if it already exists. todo: make an option to handle this.

- **session_name** (*string*) – session name:

  ```
  $ tmux new-session -s <session_name>
  ```

- **detach** (*bool*) – create session background:

  ```
  $ tmux new-session -d
  ```

- **attach_if_exists** (*bool*) – if the session_name exists, attach it. if False, this method will raise a `tmuxp.exc.TmuxSessionExists` exception
- **kill_session** (*bool*) – Kill current session if `$ tmux has-session` Useful for testing workspaces.

`Session`

7.2 Session Object

class `tmuxp.`**Session**(*server=None*, ***kwargs*)
`tmuxp.util.TmuxMappingObject`, `tmuxp.util.TmuxRelationalObject`

tmux(1) session.

Holds Window objects.

tmux(**args*, ***kwargs*)
Return Server.tmux().

Server.tmux

attach_session(*target_session=None*)
Return $ tmux attach-session aka alias: $ tmux attach.

Param target_session: str. name of the session. fnmatch(3) works.

kill_session()
$ tmux kill-session.

switch_client(*target_session=None*)
$ tmux kill-session.

Param target_session: str. note this accepts fnmatch(3). 'asdf' will kill asdfasd

rename_session(*new_name*)
Rename session and return new Session object.

rename_session (*string*) – new session name

Session

new_window(*window_name=None*, *start_directory=None*, *attach=True*)
Return Window from $ tmux new-window.

: By default, this will make the window active. For the new window to be created and not set to current, pass in attach=False.

---

window_name – window name.

```
$ tmux new-window -n <window_name> -c <start_directory>
```

- **start_directory** (*string*) – specifies the working directory in which the new created.
- **attach** – make new window the current window after creating it, default True.
- **type** – bool

Window

kill_window(*target_window=None*)
$ tmux kill-window.

Kill the current window or the window at target-window. removing it from any sessions to which it is linked.

target_window (*string*) – the target window.

_windows
Property / alias to return _list_windows().

list_windows()
Return a list of Window from the tmux(1) session.

Window

windows
Property / alias to return `list_windows()`.

children
Alias of `windows`.

attached_window()
Return active `Window` object.

`Window`

select_window(*target_window*)
Return `Window` selected via `$ tmux select-window`.

Param window: `target_window` also 'last-window' (`-l`), 'next-window' (`-n`), or 'previous-window' (`-p`)

`Window`

Todo assure `-l`, `-n`, `-p` work.

attached_pane()
Return active `Pane` object.

set_option(*option*, *value*)
Set option `$ tmux set-option <option> <value>`.

todo: needs tests

- **option** (*string*) – the window option. such as 'default-shell'.
- **value** (*string or bool*) – window value. True/False will turn in 'on' and 'off'.

show_options(*option=None*)
Return a dict of options for the window.

For familiarity with tmux, the option `option` param forwards to pick a single option, forwarding to `Session.show_option()`.

option (*string*) – optional. show a single option.

`dict`

show_option(*option*)
Return a list of options for the window.

Todo test and return True/False for on/off string

option (*string*) – option to return.

string, int or bool

findWhere(*attrs*)
Return object on first match.

Based on .findWhere() from underscore.js.

getById(*id*)
Return object based on .get() from backbone.js.

id (*string*) –

object

where (*attrs*, *first=False*)
Return objects matching child objects properties.

Based on .where() from underscore.js.

attrs (*dict*) – tmux properties to match

list

7.3 Window Object

class tmuxp.**Window** (*session=None*, ***kwargs*)
tmuxp.util.TmuxMappingObject, tmuxp.util.TmuxRelationalObject

tmux(1) window.

tmux (*cmd*, **args*, ***kwargs*)
Return Server.tmux() defaulting target_window as target.

Send command to tmux with window_id as target-window.

Specifying ('-t', 'custom-target') or ('-tcustom_target') in args will override using the object's window_id as target.

Server.tmux

select_layout (*layout=None*)
Wrapper for $ tmux select-layout <layout>.

even-horizontal: Panes are spread out evenly from left to right across the window.

even-vertical: Panes are spread evenly from top to bottom.

main-horizontal: A large (main) pane is shown at the top of the window and the remaining panes are spread from left to right in the leftover space at the bottom.

main-vertical: Similar to main-horizontal but the large pane is placed on the left and the others spread from top to bottom along the right.

tiled: Panes are spread out as evenly as possible over the window in both rows and columns.

custom: custom dimensions (see *tmux(1)* manpages).

layout (*string*) – string of the layout, 'even-horizontal', 'tiled', etc.

set_window_option (*option*, *value*)
Wrapper for $ tmux set-window-option <option> <value>.

value (*string or bool*) – window value. True/False will turn in 'on' and 'off'.

show_window_options (*option=None*)
Return a dict of options for the window.

For familiarity with tmux, the option option param forwards to pick a single option, forwarding to Window.show_window_option().

option (*string*) – optional. show a single option.

dict

show_window_option (*option*)
Return a list of options for the window.

todo: test and return True/False for on/off string

option (*string*) – option to return.

string, int

rename_window(*new_name*)
Return `Window` object `$ tmux rename-window <new_name>`.

new_name (*string*) – name of the window

kill_window()
Kill the current `Window` object. `$ tmux kill-window`.

move_window(*destination*)
Move the current `Window` object `$ tmux move-window`.

destination – the `target window` or index to move the window to.

select_pane(*target_pane*)
Return selected `Pane` through `$ tmux select-pane`.

target_pane (*string*) – `target_pane`, or `-U`,‘‘-D‘‘, `-L`, `-R` or `-l`.

`Pane`

split_window(*attach=True*)
Split window and return the created `Pane`.

: *tmux(1)* will move window to the new pane if the `split-window` target is off screen. tmux handles the `-d` the same way as `new-window` and `attach` in `Session.new_window`.

By default, this will make the window the pane is created in active. To remain on the same window and split the pane in another target window, pass in `attach=False`.

Used for splitting window and holding in a python object.

- **attach** – make new window the current window after creating it, default True.
- **type** – bool

`Pane`

attached_pane()
Return the attached `Pane`.

`Pane`

_panes
Property / alias to return `_list_panes()`.

list_panes()
Return list of `Pane` for the window.

list of `Pane`

panes
Property / alias to return `list_panes()`.

children
Alias of `panes`.

findWhere(*attrs*)
Return object on first match.

Based on .findWhere() from underscore.js.

getById(*id*)
Return object based on .get() from backbone.js.

id (*string*) –

object

where(*attrs*, *first=False*)
Return objects matching child objects properties.

Based on .where() from underscore.js.

attrs (*dict*) – tmux properties to match

list

7.4 Pane Object

class tmuxp.**Pane**(*window=None*, ***kwargs*)
tmuxp.util.TmuxMappingObject, tmuxp.util.TmuxRelationalObject

tmux(1) Pane.

window – Window

tmux(*cmd*, **args*, ***kwargs*)
Return Server.tmux() defaulting to target_pane as target.

Send command to tmux with pane_id as target-pane.

Specifying ('-t', 'custom-target') or ('-tcustom_target') in args will override using the object's pane_id as target.

Server.tmux

send_keys(*cmd*, *enter=True*)
$ tmux send-keys to the pane.

- **cmd** (*str*) – Text or input into pane
- **enter** (*bool*) – Send enter after sending the input.

clear()
Clear pane.

reset()
Reset and clear pane history.

set_width(*width*)
Set width of pane.

width (*int*) – pane width, in cells.

set_height(*height*)
Set height of pane.

height (*int*) – pane height, in cells.

resize_pane(**args*, ***kwargs*)
$ tmux resize-pane of pane and return self.

target_pane (*string*) – target_pane, or -U,‘‘-D‘‘, -L, -R.

Pane

enter()
Send carriage return to pane.

`$ tmux send-keys` send Enter to the pane.

findWhere(*attrs*)
Return object on first match.

Based on .findWhere() from underscore.js.

getById(*id*)
Return object based on .get() from backbone.js.

id (*string*) –

object

where(*attrs*, *first=False*)
Return objects matching child objects properties.

Based on .where() from underscore.js.

attrs (*dict*) – tmux properties to match

list

7.5 Internals

class tmuxp.util.**TmuxRelationalObject**
Base Class for managing tmux object child entities.

Manages collection of child objects (a `Server` has a collection of `Session` objects, a `Session` has collection of `Window`)

Children of `TmuxRelationalObject` are going to have a `self.children`, `self.childIdAttribute` and `self.list_children`.

Object	.children	.childIdAttribute	method
Server	self._sessions	'session_id'	Server.list_sessions()
Session	self._windows	'window_id'	Session.list_windows()
Window	self._panes	'pane_id'	Window.list_panes()
Pane			

findWhere(*attrs*)
Return object on first match.

Based on .findWhere() from underscore.js.

getById(*id*)
Return object based on .get() from backbone.js.

id (*string*) –

object

where(*attrs*, *first=False*)
Return objects matching child objects properties.

Based on .where() from underscore.js.

attrs (*dict*) – tmux properties to match

list

class tmuxp.util.**TmuxMappingObject**
Base: collections.MutableMapping.

Convenience container. Base class for Pane, Window, Session and Server.

Instance attributes for useful information *tmux(1)* uses for Session, Window, Pane, stored self._TMUX. For example, a Window will have a window_id and window_name.

class tmuxp.util.**tmux**(**args*, ***kwargs*)
subprocess for *tmux(1)*.

Usage:

```
proc = tmux('new-session', '-s%' % 'my session')

if proc.stderr:
    raise exc.TmuxpException('Command: %s returned error: %s' % (proc.cmd, proc.stderr))

print('tmux command returned %s' % proc.stdout)
```

Equivalent to:

```
$ tmux new-session -s my session
```

static util.**has_required_tmux_version**()
Return if tmux meets version requirement. Version >1.8 or above.

static util.**oh_my_zsh_auto_title**()
Give warning and offer to fix DISABLE_AUTO_TITLE.

see: https://github.com/robbyrussell/oh-my-zsh/pull/257

static util.**which**(*exe=None*)
Return path of bin. Python clone of /usr/bin/which.

from salt.util - https://www.github.com/saltstack/salt - license apache

exe (*string*) – Application to search PATHs for.

string

7.6 Command Line

static cli.**startup**(*config_dir*)
Initialize CLI.

config_dir (*string*) – Config directory to search

static cli.**prompt**(*name*, *default=None*)
Return user input from command line.

prompt(), prompt_bool() and prompt_choices() are from flask-script. See the flask-script license.

- **name** – prompt text
- **default** – default value if no input provided.

string

static cli.**prompt_bool**(*name*, *default=False*, *yes_choices=None*, *no_choices=None*)
Return user input from command line and converts to boolean value.

- **name** – prompt text
- **default** – default value if no input provided.
- **yes_choices** – default 'y', 'yes', '1', 'on', 'true', 't'
- **no_choices** – default 'n', 'no', '0', 'off', 'false', 'f'

bool

static cli.**prompt_choices**(*name*, *choices*, *default=None*, *resolve=<function lower at 0x7f7fb086d398>*, *no_choice=('none',)*)
Return user input from command line from set of provided choices.

- **name** – prompt text
- **choices** – list or tuple of available choices. Choices may be single strings or (key, value) tuples.
- **default** – default value if no input provided.
- **no_choice** – acceptable list of strings for "null choice"

str

static cli.**setup_logger**(*logger=None*, *level='INFO'*)
Setup logging for CLI use.

logger (Logger) – instance of logger

static cli.**get_parser**()
Return argparse.ArgumentParser instance for CLI.

static cli.**load_workspace**(*config_file*, *args*)
Build config workspace.

- **config_file** – full path to config file
- **type** – string

7.7 Configuration

7.7.1 Finding

static config.**is_config_file**(*filename, extensions=['.yml', '.yaml', '.json', '.ini']*)
Return True if file has a valid config file type.

- **filename** (*string*) – filename to check (e.g. mysession.json).
- **extensions** (*list or string*) – filetypes to check (e.g. ['.yaml', '.json']).

bool

static config.**in_dir**(*config_dir='/home/docs/.tmuxp', extensions=['.yml', '.yaml', '.json', '.ini']*)
Return a list of configs in config_dir.

- **config_dir** (*string*) – directory to search
- **extensions** (*list*) – filetypes to check (e.g. ['.yaml', '.json']).

list

static config.**in_cwd**()
Return list of configs in current working directory.

If filename is .tmuxp.py, .tmuxp.json, .tmuxp.yaml or .tmuxp.ini.

list

7.7.2 Import and export

static config.**validate_schema**(*sconf*)
Return True if config schema is correct.

sconf (*dict*) – session configuration

bool

static config.**expand**(*sconf*, *cwd=None*)
Return config with shorthand and inline properties expanded.

This is necessary to keep the code in the WorkspaceBuilder clean and also allow for neat, short-hand configurations.

As a simple example, internally, tmuxp expects that config options like shell_command are a list (array):

```
'shell_command': ['htop']
```

tmuxp configs allow for it to be simply a string:

```
'shell_command': 'htop'
```

Kaptan will load JSON/YAML/INI files into python dicts for you.

- **sconf** (*dict*) – the configuration for the session
- **cwd** – directory to expand relative paths against. should be the dir of the config directory.

dict

static config.**inline**(*sconf*)
Return config in inline form, opposite of config.expand().

sconf (*dict*) – unexpanded config file

dict

static config.**trickle**(*sconf*)
Return a dict with "trickled down" / inherited config values.

This will only work if config has been expanded to full form with config.expand().

tmuxp allows certain commands to be default at the session, window level. shell_command_before trickles down and prepends the shell_command for the pane.

sconf (*dict*) – the session configuration

dict

static config.**import_teamocil**(*sconf*)
Return tmuxp config from a teamocil yaml config.

Todo change 'root' to a cd or start_directory

Todo width in pane -> main-pain-width

Todo with_env_var

Todo clear

Todo cmd_separator

sconf (*dict*) – python dict for session configuration

static config.**import_tmuxinator**(*sconf*)
Return tmuxp config from a tmuxinator yaml config.

sconf (*dict*) – python dict for session configuration

dict

7.8 Workspace Builder

class tmuxp.**WorkspaceBuilder**(*sconf*, *server=None*)
Load workspace from session dict.

Build tmux workspace from a configuration. Creates and names windows, sets options, splits windows into panes.

The normal phase of loading is:

1. *kaptan* imports json/yaml/ini. .get() returns python dict:

```
    import kaptan
    sconf = kaptan.Kaptan(handler='yaml')
    sconf = sconfig.import_config(self.yaml_config).get()

or from config file with extension::

    import kaptan
    sconf = kaptan.Kaptan()
    sconf = sconfig.import_config('path/to/config.yaml').get()

kaptan automatically detects the handler from filenames.
```

2. config.expand() sconf inline shorthand:

```
from tmuxp import config
sconf = config.expand(sconf)
```

3. config.trickle() passes down default values from session -> window -> pane if applicable:

```
sconf = config.trickle(sconf)
```

4.(You are here) We will create a `Session` (a real `tmux(1)` session) and iterate through the list of windows, and their panes, returning full `Window` and `Pane` objects each step of the way:

```
workspace = WorkspaceBuilder(sconf=sconf)
```

It handles the magic of cases where the user may want to start a session inside tmux (when *$TMUX* is in the env variables).

build(*session=None*)
Build tmux workspace in session.

Optionally accepts `session` to build with only session object.

Without `session`, it will use `Server` at `self.server` passed in on initialization to create a new Session object.

session (`Session`) –

- session to build workspace in

iter_create_panes(*w*, *wconf*)
Return `Pane` iterating through window config dict.

Run `shell_command` with `$ tmux send-keys`.

- **w** (`Window`) – window to create panes for
- **wconf** (`dict`) – config section for window

`Pane`

iter_create_windows(*s*)
Return `Window` iterating through session config dict.

Generator yielding `Window` by iterating through `sconf['windows']`.

Applies `window_options` to window.

session – `Session` from the config

`Window`

7.9 Exceptions

exception `tmuxp.exc.`**TmuxpException**
Base Exception for Tmuxp Errors.

Also for Python 2.6 compat: http://stackoverflow.com/a/6029838

exception `tmuxp.exc.`**TmuxSessionExists**
Session does not exist in the server.

exception `tmuxp.exc.`**EmptyConfigException**
Configuration is empty.

exception `tmuxp.exc.`**ConfigError**
Error parsing tmuxp configuration dict.

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 8

Changelog

Here you can find the recent changes to tmuxp.

8.1 0.1-dev

8.1.1 2013-11-07

- [internal] Remove old logger (based on tornado's log.py), replace with new, simpler one.
- [cli] [import] fix teamocil import.
- [import] [tests]: support import teamocil `root` to `start_directory`.

8.1.2 2013-11-06

- [cli] [tests] tagged v0.0.37. Many fixes. Python 2.6 support. Will switch to per-version changelog after 0.1 release..
- [config] support for blank panes (null, `pane`, `blank`) and panes with empty strings.
- [freeze] tmuxp freeze supports exporting to blank panes.
- [freeze] tmuxp freeze will now return a blank pane for panes that would previously return a duplicate shell command, or generic python, node interpreter.

8.1.3 2013-11-05

- [cli] Support for `[-L socket-name]` and `[-S socket-path]` in autocompletion and when loading. Note, switching client into another socket may cause an error.
- Documentation tweaking to *API Reference*, *The Tao of tmux*.
- New *Roadmap*.
- pep257, pep8.

8.1.4 2013-11-04

- [internal] [docs] - pep257, pep8.
- tagged version `v0.0.36`.

8.1.5 2013-11-02

- [docs] Many documentation, pep257, pep8 fixes
- [internal] move old `Server` methods `__list_panes()`, `__list_windows` and `__list_sessions` into the single underscore.
- [cli] #12 bug fix for `$ tmuxp freeze` by @finder.
- [cli] Support for spaces in `$ tmuxp attach-session` and `$ tmuxp kill-session`, and `$ tmuxp freeze`.
- [config] [tests] support for relative paths of `start_directory`. Add an update config in *Start Directory* on *(Examples)*.

8.1.6 2013-11-01

- [internal] [tests] New servers for `Server` arguments `socket_name`, `socket_path`, `config_file`.
- [internal] `Server` support for `-2` with `colors=256` and `colors=8`.
- [cli] `$ tmuxp -2` for forcing 256 colors and `tmuxp -8` for forcing 88.
- [config] [tests] Concatenation with `start_directory` via `config.trickle()` if window `start_directory` is alphanumeric / relative (doesn't start with `/`). See *(Examples)* in *start directory*.
- [import] Fix bug with import teamocil and tmuxinator
- [import] Improve quality of tmuxinator imports. Especially `session_name` and `start_directory`.
- [cli] Allow saving with ~ in file destination.

8.1.7 2013-10-31

- [internal] `util.is_version()`
- [config] [tests]: correctly `config.trickle()` the `start_directory`.
- [config] [tests]: get `start_directory` working for configs
- [internal]: fix :meth:`Window.kill_window()` target to `session_id:window_index` for compatibility and pass tests.
- [docs] [examples]: Example for `start_directory`.
- [internal] fix bug where first and second window would load in mixed order
- [internal] `Window.move_window()` for moving window.
- [docs] major doc overhaul. front page, renamed orm_al.rst to internals.rst.

8.1.8 2013-10-30

- [cli] fix bug where if inside tmux, loading a workspace via switch_client wouldn't work.
- [cli] fix bug where `tmuxp load .` would return an error instead of a notice.
- [cli] `tmuxp freeze <filename>` experimental
- [freeze] [tests] tmuxp now has experimental support for freezing live sessions.
- [internal] `Window.kill_window()`
- [internal] support for `start_directory` (work in progress)

8.1.9 2013-10-29

- [internal] `Window.select_pane()` now accepts `-l`, `-U`, `-D`, `-L`, `-R`.
- [internal] [tests] support for `automatic-rename` option.
- [docs] 3 new *(Examples)*, 'main-pane-height', 'automatic-rename', and 'shorthands'.
- [cli] enhancements to prompts
- [cli] `tmuxp import` for teamocil and tmuxinator now has a wizard and offers to save in JSON or YAML format.
- [cli] [bug] [b6c2e84] Fix bug where tmuxp load w/ session already loaded would switch/attach even if no was entered
- [cli] when workspace loader crashes, give option to kill session, attach it or detach it.
- [cli] tmux 1.8 `set-option` / `set-window-options` command `target-window` fix.
- [internal] `WorkspaceBuilder` now has `.session` attribute accessible publicly.
- [cli] tmux will now use `Session.switch_client()` and `Session.attach_session()` to open new sessions instead of `os.exec`.
- [config] tmuxp now allows a new shorter form for panes. Panes can just be a string. See the shorthand form in the *(Examples)* section.
- [cli] [config] support loading `.yml`.

8.1.10 2013-10-28

- [cli] fix `tmuxp load .` fixed
- [cli] fix `tmuxp convert <file>` fixed.
- [internal] pep257 fixes.
- [internal] [tests] - `Pane` now has `Pane.set_width()` and `Pane.set_height()`.
- [tests] `./run_tests.py --tests` now automatically prepends `tmuxp.testsuite` to names.
- [internal] `Window.tmux()` and `Pane.tmux()` will automatically add their `{window/pane}_id` if one isn't specific.

8.1.11 2013-10-27

- [cli] argcomplete overhaul for CLI bash completion.
- [cli] `tmuxp load`, `tmuxp convert` and `tmuxp import` now support relative and full filenames in addition to searching the config directory.

8.1.12 2013-10-26

- [import] [tests] initial version of tmuxinator and teamocil config importer. it does not support all options and it not guaranteed to fully convert window/pane size and state.
- [internal] `config.in_dir()` supports a list of `extensions` for filetypes to search, i.e. `['.yaml', '.json']`.
- [internal] `config.is_config_file()` now supports `extensions` argument as a string also.
- [cli] fix `$ tmuxp load -l` to work correctly alongside `$ tmuxp load filename`.

8.1.13 2013-10-25

- [cli] fix bug where `-v` and `--version` wouldn't print version.
- [cli] property handle case where no tmux server exists when `attach-session` or `kill-session` is used.
- [import] [tests] test fixtures and inital work for importing tmuxinator and teamocil configs

8.1.14 2013-10-24

- [internal] clean out old code for `automatic-rename` option. it will be reimplemented fresh.
- [cli] check for `oh-my-zsh` when using `$SHELL zsh`. Prompt if `DISABLE_AUTO_TITLE` is unset or set to `true`.
- [cli] tmuxp can now `$ tmuxp convert <file>` from JSON <=> YAML, back and forth.
- [docs] New examples in JSON. Update the *(Examples)* page in the docs.
- [dev] `.tmuxp.json` now exists as a config for tmuxp development and as an example.
- [cli] Fix bug where `tmuxp kill-session` would give bad output
- [cli] Fix bug in tab completion for listing sessions with no tmux server is active.

8.1.15 2013-10-23

- [cli] zsh/bash/tcsh completion improvements for tab-completion options
- [cli] tmuxp `kill-session` with tab-completion.
- [cli] tmuxp `attach-session` with tab-completion. Attach session will `switch-client` for you if you are inside of of a tmux client.
- [cli] tmuxp `load` for loading configs.
- [tests] unit test fixes.

8.1.16 2013-10-21

- [cli] Make 1.8 the official minimym version, give warning notice to upgrade tmux if out of date
- Fix regression causing unexpected build behavior due to unremoved code supporting old tmux versions.
- [docs] Added 2 new examples to the *(Examples)* page.
- [docs] Examples now have graphics
- [cli] [internal] `$ tmuxp -v` will print the version info.

8.1.17 2013-10-19

- [internal] tmuxp will now give warning and sys.exit() with a message if `tmux` not found in system PATH
- [internal] major internal overhaul of `Server`, `Session` , `Window`, and `Pane` continues.
  - `Server` has @property `Server.sessions()`, which is forward to `Server.list_sessions()` (kept to keep tmux commands in serendipty with api), `Server._list_sessions()` returns dict object from `Server.__list_sessions()` tmux command. `Server.__list_sessions()` exists to keep the command layered so it can be tested against in a matrix with travis and compatibility methods can be made.
  - `Session` now has @proprety `Session.windows()` returning a list of `Window` objects via `Session.list_windows()`. @property `Session._windows()` to `Session._list_windows()` to return a list of dicts without making objects.
  - `Window` now has @proprety `Window.panes()` returning a list of `Pane` objects via `Window.list_panes()`. @property `Window._panes()` to `Window._list_panes()` to return a list of dicts without making objects.

8.1.18 2013-10-18

- [internal] major internal overhaul of `Server`, `Session`, `Window`, and `Pane`.
  - `Session`, `Window` and `Pane` now refer to a data object in `Server` internally and always pull the freshest data.
  - A lot of code and complexity regarding setting new data for objects has been reduced since objects use their unique key identifier to filter their objects through the windows and panes in `Server` object.
  - `Server` object is what does the updating now.
- [project] some research into supporting legacy tmux versions. tmux 1.6 and 1.7 support seem likely eventually if there is enough demand.
- [internal] python 3 support

8.1.19 2013-10-17

- [docs] updated README docs with new project details, screenshots
- [dev] [docs] - new example `.tmuxp.yaml` file updated to include development workflow. Removed nodemon as the tool for checking files for now.
- [cli] Support for switching sessions from within tmux. In both cases after the the session is built and if session already exists.

8.1.20 2013-10-16

- [cli] use `util.which()` from salt.util to find tmux binary.
- [pypi / packaging] add MANIFEST.in, fix issue where package would not install because missing file
- [cli] bash / zsh completion.
- [docs] New page on *(Internals)*.
- [docs] Updates to *The Tao of tmux* page.
- [docs] add vim modeline for rst to bottom of this page
- [internal] [tests] Server is now a subclass of `util.TmuxObject`.
- [internal] [tests] subclasses of `util.TmuxRelationalObject`, `Server`, `Session`, `Window`, `Pane` now have `util.TmuxRelationalObject.getById()` (similar to .get() in backbone.js collection), `util.TmuxRelationalObject.where()` and `util.TmuxRelationalObject.findWhere()` (.where() and .findWhere() in underscore.js), to easily find child objects.
- [internal] tmux object mapping has been split into `util.TmuxMappingObject`. The mapping and the relational has been decoupled to allow `Server` to have children while not being a dict-like object.
- [internal] `Server`, `Session`, `Window`, `Pane` now explicitly mixin subclasses.

8.1.21 2013-10-15

- [docs] new theme
- [docs] initial examples, misc. updates, front page update.
- [cli] support for `$ tmux .` to load `.tmuxp.{yaml/json/py}` in current working directory.
- [cli] support for `socket-name` (`-L`) and `socket-path` (`socket-path`)
- [config] [builder] Support for 1-command pane items.

```
session_name: my session
windows:
  - window_name: hi
    panes:
    - bash
    - htop
```

- [cli] If session name is already exists, prompt to attach.

8.1.22 2013-10-14

- [cli] can now -l to list configs in current directory and $HOME/.tmuxp
- [cli] tmuxp can now launch configs and build sessions
- [internal] new exceptions
- [config] [tests] `config.check_consistency()` to verify and diagnose issues with config files.
- [cli] [tests] `cli.startup()`
- [config] [tests] `config.is_config_file()`
- [config] [tests] `config.in_dir()`

- [config] [tests] `config.in_cwd()`

8.1.23 2013-10-13

- [cli] [tests] `config.inline()` to produce far far better looking config exports and tests.
- `Pane.resize_pane()` and tests
- documentation fixes and updates
- [internal] `Session.refresh()`, `Window.refresh()`, `Pane.refresh()`.
- [internal] `Server.find()`, `Session.find()`, `Window.find()`.

8.1.24 2013-10-12

- [tests] Test documentation updates
- [internal] [tests] Builder is now `WorkspaceBuilder` + tests. - WorkspaceBuilder can build panes - WorkspaceBuilder can build windows and set options
- [internal] [tests] `Window.show_window_options()`, `Window.show_window_option()`, `Window.set_window_option()`
- [internal] [tests] `Session.show_options()`, `Session.show_option()`, `Session.set_option()`

8.1.25 2013-10-11

- [builder] More preparation for builder / session maker utility.
- [tests] Major test runner and test suite overhaul.
- [docs] Documentation for development environment and test runner updated.
- [tests] Travis now tests against tmux 1.8 and latest source. Door open for future testing against python 3 and earlier tmux versions in the future.
- [internal] Quiet logger down in some areas
- [internal] __future__ imports for future python 3 compatibility
- [internal] setup.py import __version__ via regex from tmuxp package
- [cli] move beginnings of cli to tmuxp.cli

8.1.26 2013-10-09

- New logging module
- Removed dependency logutils
- Removed dependency sh

8.1.27 2013-10-08

- switch to semver

8.1.28 born

- changedlog added
- sphinx documentation

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 9

Roadmap

9.1 0.1 milestone

- pep8, pep257 for tests. (ongoing)
- if no `window_name` in config, option `automatic_rename: on` by default and test.
- example of doctest/usage example of creating a new tmux session from object
- example of accessing a current tmux session from object
- example of loading a tmux session from a :py:obj:`dict`. example of the schema.
- remove duplicated code in `tmuxp.cli`.
- Determine the workflow for `$ tmuxp load filename` for sure. Document it.
- Handle case where switching client via `$ tmuxp load` or `$ tmuxp attach-session` into another socket may cause an error.

9.1.1 Done

- `shell_command_before` for teamocil.
- teamocil and tmuxinator import support for blank panes
- Add help docs to CLI commands and options.
- tmuxp -L and -S autocomplete to correct server `socket-name` and `socket-path`.
- Get sphinx-argparse in good enough shape for docs.
- Rename functions - `util.version()` to `util.has_required_tmux_version()`. (done) - `config.check_consistency()` to `config.validate_schema()`. (done)
- Python 2.6 support
- Remove doc for `run_tests.py` visual test runner. Just have instruction for running test builder package.
- tmuxp freeze:
  - offer to save `session_name` as default file if filename doesn't already exist in config dir.
  - Remove -zsh command, python, etc if just in shell.

9.2 Future

- Automatically grab `[-L socket-name]` and `[-S socket-path]` from where tmuxp was ran. `os.environ.get('$TMUX')`.
- Automatically grab current sesession, window, session from where script is ran. `$ tmuxp kill-window` should kill current window.

  Or, As an alternative, create a mapping that pipes commands right into tmux like a layer cake.

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 10

The Tao of tmux

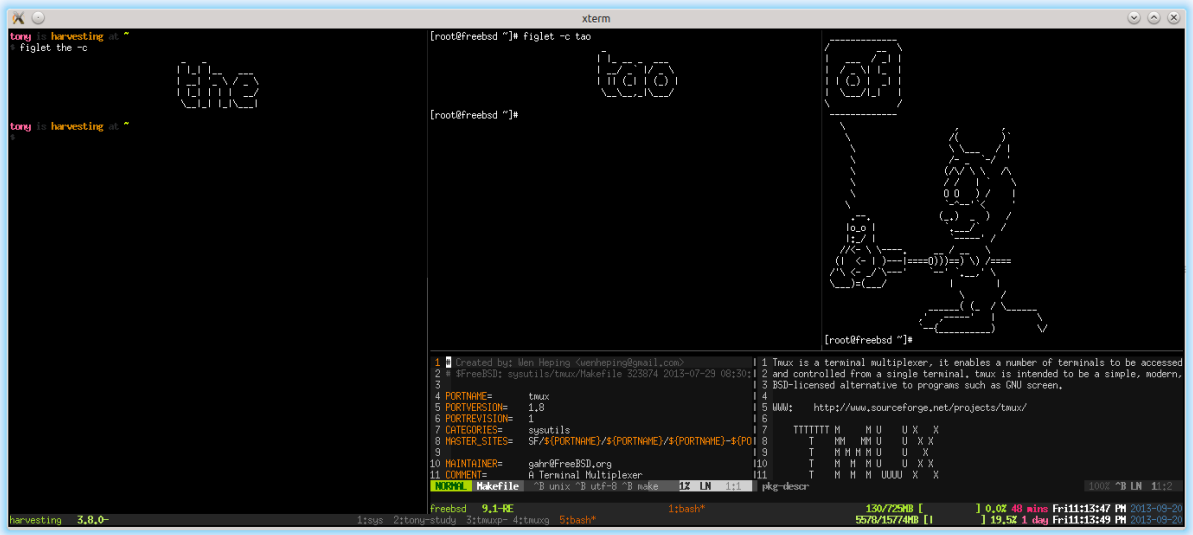


Figure 10.1: BSD-licensed terminal multiplexer.

tmux is geared for developers and admins who interact regularly with CLI (text-only interfaces)

In the world of computers, there are 2 realms:

1. The text realm
2. The graphical realm

Tmux resides in the text realm. This is about fixed-width fonts and that old fashioned black terminal.

tmux is to the console what a desktop is to gui apps. It's a world inside the text dimension. Inside tmux you can:

- multitask inside the terminal, run multiple applications.
- have multiple command lines (pane) in the same window
- have multiple windows (window) in the workspace (session)
- switch between multiple workspaces, like virtual desktops

10.1 Overview

10.1.1 For Terminals only.

No graphics.

Uses:

- window-manager for text-based applications
- keep applications in a background process

10.1.2 Text-based window manager

tmux	**"Desktop"-Speak**	**Plain English**
Multiplexer	Multitasking	Do more than one thing at once
Session	Desktop	Where stuff gets done
Window	Virtual Desktop or screen	Has windows inside
Pane	Application	Performs operations

10.1.3 Multiple terminals in one screen

It allows multiple applications or terminals to run at once.

Being able to run 2 or more terminals on one screen is convenient. This way one screen can be used to edit a file, and another may be used to `$ tail -F` a logfile.

```
$ bash  $ bash
```

```
$ bash  $ bash
$ vim   $ bash
```

tmux supports as manys terminals as you want.

10.1.4 It allows multiple layouts to view the apps

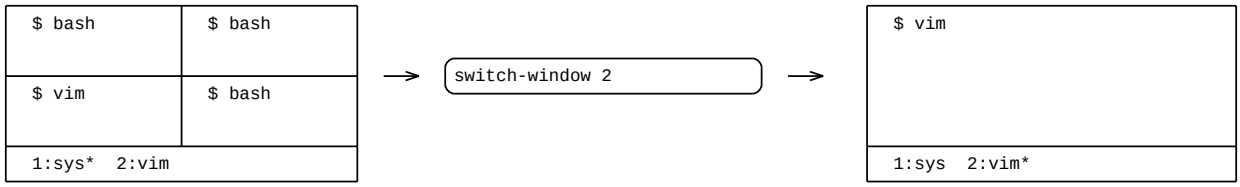
Different applications are viewable better in different layouts.

It allows switching between layouts such as...

10.1.5 Organize apps based on your needs

You can categorize and keep many terminals / applications separated into multiple windows.

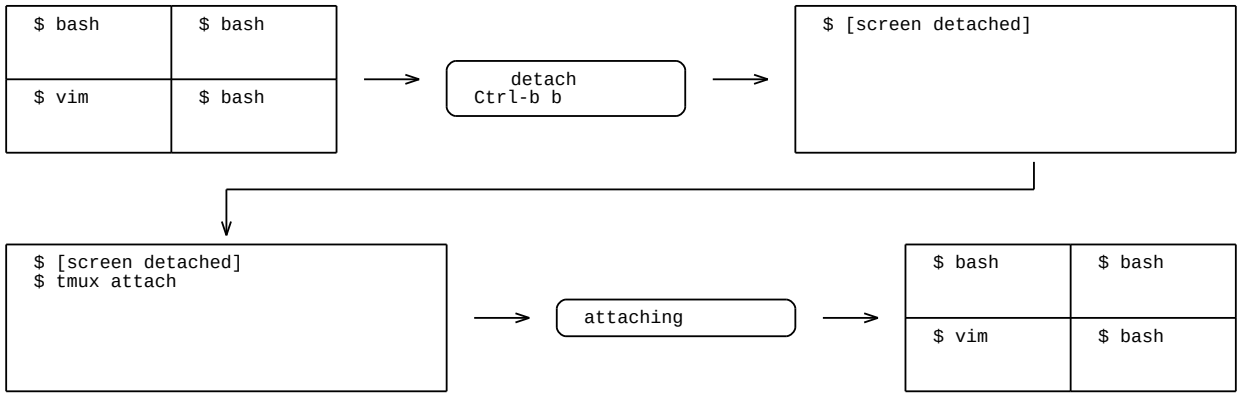
In addition to being able to split the terminal into multiple panes, you can create new windows as much as you want.



You can switch between the windows you create.

10.1.6 Resume everything later

You can leave tmux and all applications running (detach), log out, make a sandwich, and re-(attach), all applications are still running!



10.2 Core Concepts

10.2.1 Your workflow

You can keep tmux on a server with your latest work, come back and resume your "train of thought" and work.

Multitasking. More important than any technical jargon - it's preserving the thinking you have, whether you were in the midst of a one-off task, or a common task.

If you do a task commonly, it may help to use an application which manages tmux workspaces.

10.2.2 Server

A server contains *Session*'s.

tmux starts the server automatically if it's not running.

In advanced cases, multiple can be run by specifying `[-L socket-name]` and `[-S socket-path]`.

10.2.3 Client

Attaches to a tmux *Server*.

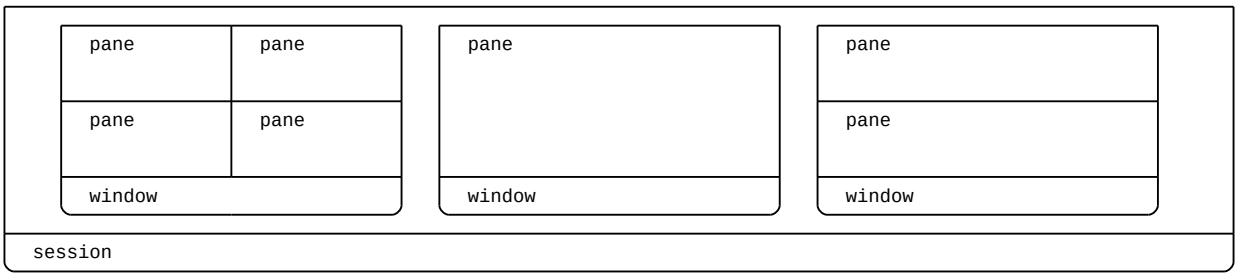
10.2.4 Session

Inside a tmux *Server*.

The session has 1 or more *Window*. The bottom bar in tmux show a list of windows. Normally they can be navigated with `Ctrl-a [0-9]`, `Ctrl-a n` and `Ctrl-a p`.

Sessions can have a `session_name`.

Uniquely identified by `session_id`.



10.2.5 Window

Entity of a *Session*.

Can have 1 or more *Pane*.

panes can be organized with a layouts.

windows can have names.

10.2.6 Pane

Linked to a *Window*.

a pseudoterminal.

10.2.7 Target

A target, cited in the manual as `[-t target]` can be a session, window or pane.

10.2.8 License

: tmuxp is usable but still needs your help reporting errors, bugs and usability feedback. If you encounter an error, please post on the Issue tracker.

CHAPTER 11

(Glossary)

tmuxp tmuxpythontmux

tmux(1) tmux(1)tmuxptmuxptmux

kaptan python kaptan on github

t