Up Documentation

Release 0.1.0

Aaron Spaulding

Sep 27, 2017

Contents

Contents:

CHAPTER 1

Getting Started

CHAPTER 2

Up - A next generation status monitor

Sometimes you just need to know, is it up? The goal of this project is to create an easy to use, but highly customizable status monitor.

Setup

First start by installing the environment.

```
$ mkdir example-status

$ cd example-status

$ virtualenv . -p python3 --no-site-packages

$ bin/pip install up
```

Now you need to create the *upfile.py*. It goes in the same folder as everything else. From here you can setup what you want to monitor.

```
from up import status, source, sink

class ExampleStatus(status.StatusMonitor):

    source = source.HTTPStatusSource('Example Status', 'https://example.com/')
    sink = sink.StdOutStatusSink()
```

You can now run it like this.

```
$ bin/up
Example Status: UP
```

Monitoring Multiple URL's

Up uses a "tinker-toy" pattern allowing you to combine sources to build whatever kind of monitor you need. A *StatusTreeSource* will let you combine multiple sources into one.

```
from up import status, source, sink

class ExampleStatus(status.StatusMonitor):

    # You can also try a ThreadedTreeSource which runs the monitors
    # in parallel.
    source = source.StatusTreeSource('Example Status', [
        source.HTTPStatusSource('PROD', 'https://example.com/'),
        source.HTTPStatusSource('QA', 'https://qa.example.com/')
    ])
    sink = sink.StdOutStatusSink()
```

Up will query each of the sources and give you a simplified status.

```
$ bin/up
Example Status: HALF UP
```

For more information use -v.

```
$ bin/up -v
Example Status: HALF UP (50%)
    PROD: UP
    QA: DOWN
```

Checking the status of GitHub

Up comes with a source that reads from GitHub's status API.

```
from up import status, source, sink

class ExampleStatus(status.StatusMonitor):

    source = source.GitHubStatusSource('GitHub Status')
    sink = sink.StdOutStatusSink()
```

```
$ bin/up -v
GitHub Status: UP
```

Developers Setup

```
$ virtualenv . -p python3 --no-site-packages

$ bin/python setup.py develop
```

Changelog

- **Next** Nothing Yet.
- 0.2.1 - Fix templates and static resources missing from egg
- 0.2.0 - Detect ConnectionError and set status to DOWN; Expose Web Interface; Experimental SNMP monitoring (will most likely change)
- 0.1.0 - Initial release

CHAPTER 3

Up– a next generation status monitor

class up.status.**StatusMonitor**
A status monitor. Inherit from this class to create your own.

main(*note*, *mood*, *verbosity*)

sink = None

source = None

class up.source.**StatusSource**(*name*)
A base class for a status. All statuses should extend from this class.

prepare()
Abstract method, implementing classes should use this hook to poll the status and update *self.status*.

timed_prepare()
Call prepare and record the amount of time it takes to finish.

class up.source.**StatusTreeSource**(*name*, *children=None*)
A tree source allows you to organize your statuses. It will allow you to see the combined status of all of the statuses underneath it.

calculate()
Calculates the percentage of the children that are up.

prepare()
Calls *timed_prepare* on all of its children and sets its status to the percentage of children that are up.

class up.source.**ThreadedTreeSource**(*name*, *children=None*)
Similar to *StatusTreeSource*, *ThreadedTreeSource* runs all of the *timed_prepare* calls in parallel. This is useful for statuses that require network access.

prepare()

class up.source.**ThreadedTreeSource**(*name*, *children=None*)
Similar to *StatusTreeSource*, *ThreadedTreeSource* runs all of the *timed_prepare* calls in parallel. This is useful for statuses that require network access.

prepare()

class up.source.**HTTPStatusSource**(*name*, *url*)
Makes a request to a url and uses the HTTP status code to determine of the server is up.

Redirects are followed before determining the status.If the status code is 2xx the server is considered to be up. Status codes 4xx, 5xx, truncated connections, or otherwise mangled responses are considered down.

You can also use this as a basis for parsed statuses.

prepare()

class up.source.**GitHubStatusSource**(*name='GitHub'*, *url='https://status.github.com/api/status.json'*)
Using the GitHub status api you can retrieve the current status of GitHub.

GitHub Statuses: Good maps to UP Minor maps to HALF UP Major maps to DOWN

prepare()

class up.sink.**StatusSink**
A base class for a sink. Sinks are used for outputting the the statuses gathered by sources.

add_annotation(*note*)
Add an annotation to the output.

process_status(*status*, *deep=0*, *surname=''*)
Implementing classes should implement this hook to output an individual status.

Parameters

- **status** – a status object to be output.
- **deep** – an int of the number of *StatusTreeSource* that have been traversed down.
- **surname** – The names of all of the parent sources concatenated with a '/'.

set_mood(*mood*)
The mood of any messaging.

set_status(*source*, *deep=0*, *surname=''*)

set_verbosity(*verbosity*)
If the sink outputs to stdout, use this value to control the amount of output.

class up.sink.**TreeStatusSink**(*children*)
Output a source to multiple sinks.

add_annotation(*note*)

set_mood(*mood*)

set_status(*source*)

set_verbosity(*verbosity*)

class up.sink.**MongoStatusSink**(*domain*, *port*)
Output a source to MongoDB. Each status is stored as a separate document.

Up-web requires statuses to be stored with this sink.

DB_NAME = 'up'

add_annotation(*note*)

process_status(*status*, *deep=0*, *surname=''*)

class up.sink.**StdOutStatusSink**
Output a source to stdout with fancy colors for easy skimming.

add_annotation(*note*)

messages = {'pessimist': ['\x1b[91mGET TO WORK!\x1b[0m', '\x1b[91mMOSTLY DOWN\x1b[0m', '\x1b[93mHALF

process_status (*status*, *deep=0*, *surname=''*)

CHAPTER 4

Up Web– for the bigger picture

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index