# debmetrics Documentation

***Release 0.0.1***

**Joseph Bisch**

March 22, 2016

Contents

Contents:

CHAPTER 1

# Documentation for the Code

CHAPTER 2

# How to Add a New Metric

You need the following to add a new metric:

- a manifest
- a script (pull or push)
- a graph script (optional)

## 2.1 Manifest

See the manifests directory for example manifests.

Manifest files are parsed by Python's configparser.

The following are the different options available:

**type** pull|push

What kind of metric is it? Pull metric or push metric.

**freq** crontab format

How frequently to call the metric's script.

**script** a filesystem location

The location of the script to be ran on the filesystem. Not currently used. Right now the name of the script must match the name of the manifest file.

**fields** field1:type1, field2:type2, ...

A comma separated list of fields and the type where the field and type are separated by a colon. Used by SQLAlchemy to create the db structure and insert and retreive data from the db.

**display_fields** field1, field2, field3, ...

A comma separated list of fields. Use by debmetrics to display on the web interface. Unlike fields, the field names do not have to be valid Python variables.

**format** csv|json

The format that the script returns data in.

**override_ts** true|false

If false a ts column is automatically added that stores a timestamp. If true the ts column is not added.

**delete_before_insert** truelfalse

If true then delete all existing data in the db table before each time new data form the script is added to the db. For example, the releases metric uses this.

**graph_type** default|custom

If default, a timeseries graph is generated and displayed. If custom a graph script will run that generates a custom graph. It is designed for more complex metrics that don't work well with the default timeseries graph. Currently there is no way to display custom graphs.

**description** The description is some text that gets displayed to the user. Currently the static interface is the only one that displays the description.

**token** Only used by push metrics. Must match the token used in the push script. Used to prevent unauthorized pushing of data.

## 2.2 Pull Script

See the debmetrics/pull_scripts directory for example pull scripts.

Your pull script should write data to stdout. For example, if using csv as the format:

```
writer = csv.writer(sys.stdout)
writer.writerow(['field1', 'field2',...]) # replace with actual field names
writer.writerow([data[0], data[1],...])
```

Make sure to end your script with exit(0) if everything went okay.

## 2.3 Push Script

The push script goes on a remote server.

It is best to demonstrate push scripts with an example:

```
#!/usr/bin/python

import re
import csv
import urllib
import urllib2
import StringIO
from BeautifulSoup import BeautifulSoup

def run():
    url = 'https://bugs.debian.org/release-critical/'
    soup = BeautifulSoup(urllib2.urlopen(url))
    date = soup.findAll('h2')[0]
    p = date.findNext('p')
    nums = re.findall('\d+', p.text)
    url = 'http://metrics.debian.net/push'
    si = StringIO.StringIO()
    cw = csv.writer(si)
    cw.writerow(['ts', 'rc_bugs', 'with_patch', 'with_fix', 'ignored', 'concern_current_stable', 'co
    cw.writerow([date.text] + nums)
    data = si.getvalue().strip('\r\n')
    values = {'data': data,
```

```
                'metric': 'rc_bug_count',
                'format': 'csv',
                'token': '1'}
    data = urllib.urlencode(values)
    req = urllib2.Request(url, data)
    response = urllib2.urlopen(req)
    print response.read()

if __name__ == '__main__':
    run()
```

The token should match the one in the manifest file.

## 2.4 Graph Script

At this time there is no documentation on graph scripts.

CHAPTER 3

# Hacking on Debmetrics

It is easy to hack on debmetrics or just setup a local instance.

- git clone git://anonscm.debian.org/qa/debmetrics.git
- cd debmetrics
- mkdir graphs
- make
- ./minified_grabber
- crontab crontab
- adjust .debmetrics.ini to point to Postgres instance
- python create_all.py

## 3.1 Debmetrics Structure

| | |
|---|---|
| app.py: | All the flask code is contained in app.py |
| create_all.py: | Creates the db structure from the model files |
| crontab: | A crontab file that runs the pull_runner.py every 5 minutes |
| debmetrics.wsgi: | A sample wsgi file for use with Apache |
| minified_grabber: | Downloads js and css files |
| pull_runner.py: | Runs all the pull_scripts, inserts data in db, and generates graphs |
| push_runner.py: | Handles data from push scripts and generates graphs |
| debmetrics/base.py: | A universal SQLAlchemy base for the entire debmetrics project |
| debmetrics/config_reader.py: | Reads the config file (.debmetrics.ini) |
| debmetrics/graph_helper.py: | Implements the time_series_graph() function |
| debmetrics/graph_scripts/: | Stores the custom graph scripts |
| debmetrics/manifest2index.py: | Generates an index file from the manifest file names |
| debmetrics/manifest2orm.py: | Generates the ORM from the manifest files |
| debmetrics/models/: | Stores the autogenerated models |
| debmetrics/pull_scripts/: | Stores the pull scripts |
| debmetrics/runner_helper.py: | A helper for pull_runner.py and push_runner.py |
| docs/: | Sphinx documentation |
| manifests/: | Stores the manifests |
| static/: | The flask static directory. Serves images, js, and css. |
| templates/: | The flask template directory |
| tests/: | The tests |

CHAPTER 4

# Running Debmetrics Tests

To run the tests, run *nosetests3* from anywhere within the directory structure of debmetrics.

Before running the tests, you should edit .debmetrics.ini to have the TEST flag set True. The TEST flag causes debmetrics to use the metrics_test schema instead of the metrics schema like normal. That way the normal data is not disturbed by the test data. After setting the flag, delete all the model files in debmetrics/models/ and regenerate the models by running make in the root of debmetrics.

After you are done running the tests, you should set the TEST flag to False and repeat the above steps.

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search