Greenwave Documentation

Release 0.1.0

Red Hat, Inc. and others

Jun 06, 2017

Contents:

Greenwave is a service to decide whether a software artifact can pass certain gating points in a software delivery pipeline, based on test results stored in ResultsDB and waivers stored in WaiverDB.

CHAPTER 1

Greenwave APIs

Greenwave is a web application built using Flask and SQLAlchemy.

It provides a *HTTP REST API* for applications to use.

1.1 HTTP REST API

POST /api/v1.0/decision

Make a decision after evaluating all applicable policies based on test results. The request must be *application/json*.

JSON Parameters

- **product_version** (*string*) – The product version string used for querying WaiverDB.
- **decision_context** (*string*) – The decision context string.
- **subject** (*array*) – A list of items about which the caller is requesting a decision used for querying ResultsDB. For example, a list of build NVRs.

Status Codes

- 200 OK – A decision was made.
- 400 Bad Request – Invalid data was given.

CHAPTER 2

Contribution Guidelines

Please follow the following contribution guidelines when contributing a pull request.

2.1 Code style

We follow the PEP 8 style guide for Python. The test suite includes a test that enforces the required style, so all you need to do is run the tests to ensure your code follows the style. If the unit test passes, you are good to go!

2.2 Unit tests

Greenwave uses the Python unittest framework for unit tests.

Patches should be accompanied by one or more tests to demonstrate the feature or bugfix works. This makes the review process much easier since it allows the reviewer to run your code with very little effort, and it lets developers know when they break your code.

2.2.1 Running Tests

Tests are run with py.test via tox. You can run individual environments by using the `-e` flag. For example, `tox -e lint` runs the linter.

2.2.2 Test Organization

The test organization is as follows:

1. Each module in the application has a corresponding test module. These modules is organized in the test package to mirror the package they test. That is, `greenwave/app.py` has a test module in located at `greenwave/tests/test_app.py`
2. Within each test module, follow the unittest code organization guidelines.

3. Include documentation blocks for each test case that explain the goal of the test.

2.3 Documentation

Greenwave uses sphinx to create its documentation. New packages, modules, classes, methods, functions, and attributes all should be documented using "Google style" docstrings.

Python API documentation is automatically generated from the code using Sphinx's autodoc extension. HTTP REST API documentation is automatically generated from the code using the httpdomain extension.

2.4 Development Environment

Set up a Python virtual environment and then install Greenwave:

```
$ pip install -r dev-requirements.txt
$ pip install -e .
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

HTTP Routing Table

/api

`POST /api/v1.0/decision`, 3

Python Module Index

g

Index

G