TESTING CFENGINE POLICY

NICK ANDERSON

Created: 2017-02-06 Mon 15:24

FORK ME ON GITHUB!

https://github.com/nickanderson/presentation-testing-cfengine-policy

INTRODUCTIONS

MY NAME IS NICK.

- Wife, 2 kids, and a dog
- Sysadmin/Infrastructure Engineer
- You can find me online
  - nick@cmdln.org | nick.anderson@cfengine.com
  - twitter: @cmdln_
  - cmdln.org
  - linkedin.com/in/hithisisnick

How about you?

WHY TEST?

Implementations are ephemeral. Documented reasoning is priceless. – Mark Burgess

- **Inspect what you expect**
  - Prove policy behaves as expected
  - Catch what you can as early as possible

WHO IS TESTING CFENGINE POLICY?

- **CFEngine:** Documentation Examples, Core acceptance tests, MPF acceptance tests, Standard library utility bundles
- **Evolve Thinking:** Evolve Thinking Free library
- **Normation:** NCF
- **Others:** CFEngine Provisioner for Test Kitchen, Marco Marongiu, Jarle Bjørgeengen

CORE ACCEPTANCE TESTS

Found in tests/acceptance

THE MOST SIMPLE TEST

This test will fail unless the system has the `linux` class defined.

```
bundle agent main
{
  classes:
    "pass" expression => "linux";

  reports:
    pass::
      "$(this.promise_filename) Pass";

    !pass::
      "$(this.promise_filename) FAIL";
}
```

```
R: /home/nickanderson/src/presentations/testing-cfengine-policy/cfengine3-16585LMB Pass
```

BUNDLE META INFO

Controls interpretation of a test result

- Use in a bundle named `test`
- Requires inclusion of `default.cf.sub` and `default($(this.promise_filename))` for the bundlesequence.

```
body common control
{
    inputs => { "../default.cf.sub" };
    bundlesequence  => { default("$(this.promise_filename)") };
}
```

NOTEABLE BUNDLE META VARS

`description`
: Describes what is being tested.

`test_skip_unsupported`
: Skips a test because it makes no sense on that platform (e.g. symbolic links on Windows).

`test_skip_needs_work`
: Skips a test because the test itself is not adapted to the platform (even if the functionality exists).

`*test_soft_fail`
: Requires meta tag representing the associated issue ID. Runs the test, but failure does not fail the build. **Good for incoming bug reports**.

`*test_suppress_fail`
: Failures are counted, but won't block the build.

* **Requires meta tag representing the associated issue ID**

BUNDLE META INFO EXAMPLE

```
bundle agent test
{
  meta:
    "description" string => "This tests ....";
    "test_soft_fail"
      string => "any", # Class expression describing platforms (hard classes)
      meta => { "CFE-XXX" };
}
```

STAGED TESTS

- Not expected to pass, and skipped unless running `testall` with `--staging`
- Can be placed in staging directory (not run automatically)
- Now preferring the use of bundle meta info to not fail the build (run automatically)
  - But do not fail the build in our CI system

UNSAFE TESTS

- Modify the system outside of `/tmp`
- Should be placed in a directory named `unsafe`
- Can be run with `--unsafe` option to `testall`

PARALLEL AND SERIAL TESTS

- Run `n` tests in parallel `./testall -jobs=[n]`
- Tests with `serial` in the name are run in strict lexical order

TIMED TESTS

- Allows tests to wait for extended period of time
- Use `dcs_wait( $(this.promise_filename), <seconds>)`

FAULT TESTS

- Are expected to fault, for example invalid syntax
- Should have suffix of `.x.cf`

NETWORK TESTS

- Use external networked resources
- Should be placed in a directory named '`network`'
- Can be disaled with '`--no-network`' option to `testall`

RUNNING CORE ACCEPTANCE TEST

```
./testall --bindir=/var/cfengine/bin
```

WRITING A CORE ACCEPTANCE TEST

- Start with self contained policy to excercise and validate the behaviour.
- Include `default.cf.sub` in `body common control`
- Use `default("$(this.promise_filename)")` for the `bundlesequence` in `body common control`
- Split test into approrpirate bundles

SIMPLE EXAMPLE TEST

```
body common control
{
    inputs => { "../default.cf.sub" };
    bundlesequence  => { default("$(this.promise_filename)") };
}
bundle agent init
{
  files:
    "$(G.testfile)"
      delete => tidy;
}
bundle agent test
{
  meta:
    "description" string => "Test that a file gets created";

  files:
    "$(G.testfile)"
      create => "true",
      classes => scoped_classes_generic("namespace", "testfile");
}
bundle agent check
{
  methods:
    "" usebundle => dcs_passif( "testfile_repaired", $(this.promise_filename) );
}
```

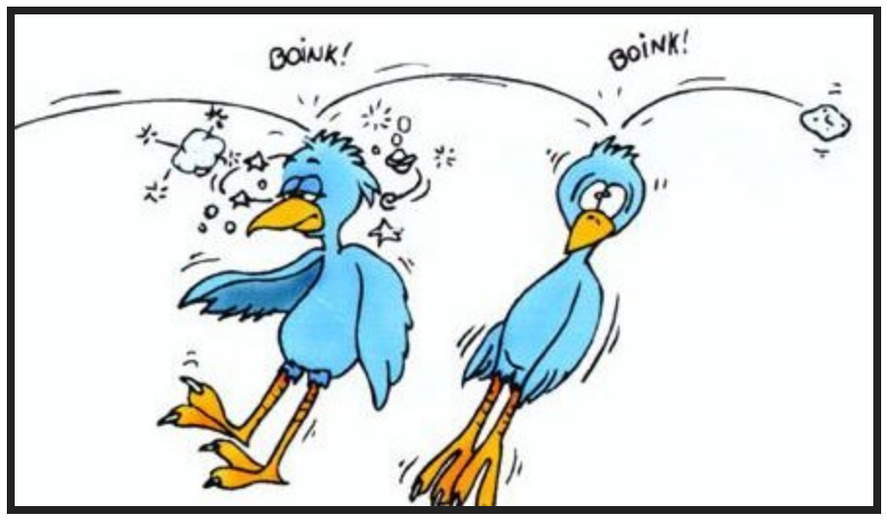
RUNNING THE TEST

```
$ ./testall example.cf
================================================================================
Testsuite started at 2016-01-31 17:06:40
--------------------------------------------------------------------------------
Total tests: 1

CRASHING_TESTS: enabled
NETWORK_TESTS: enabled
STAGING_TESTS: disabled
UNSAFE_TESTS: disabled
LIBXML2_TESTS: enabled

./example.cf Pass

================================================================================
Testsuite finished at 2016-01-31 17:06:41 (1 seconds)

Passed tests:  1
Failed tests:  0
Skipped tests: 0
Soft failures: 0
Total tests:   1
```

• • •


BUT WAIT
THERE'S MORE
memegenerator.net

IMPROVE DOCUMENTATION AND TESTING WITH TEST SUPPORT FOR EXAMPLES



CFENGINE CORE EXAMPLES WITH TEST SUPPORT

- Example with test support Optional `prep` section to prepare the environment for testing.
  - Required `cfengine3` section containing policy to excercise the test
  - Required `example_output`
- Example doc usage
- Example doc result

TESTING YOUR OWN POLICIES WITH TAP OR JUNIT

- Utility bundles in `$(sys.libdir)/testing.cf`

IMPLEMENTING A SIMPLE TEST WITH TAP AND JUNIT OUTPUT

```
body file control { inputs => { "$(sys.libdir)/stdlib.cf", "$(sys.libdir)/testing.cf" }; }

bundle agent main
{
  classes:
    "BUNDLE_CLASS" expression => "any";

  methods:
    "Check namespace scoped class"
      usebundle => testing_ok_if("NAMESPACE_CLASS",
                                 "Checking to see if 'NAMESPACE_CLASS' is defined",
                                 "'NAMESPACE_CLASS' is *not* defined.", "Extra trace info", "TA

    "Check bundle scoped class"
      inherit => "true",
      usebundle => testing_ok_if("BUNDLE_CLASS",
                                 "Checking to see if 'BUNDLE_CLASS' is defined",
                                 "'BUNDLE_CLASS' is *not* defined.", "Extra trace info", "TAP")

    "TAP Summary Report"
      usebundle => testing_tap_report("/tmp/test_result.txt");

    "JUnit Summary Report"
      usebundle => testing_junit_report("/tmp/test_result.xml");
reports:
    "Content of /tmp/test_result.txt:$(const.n)"
      printfile => cat("/tmp/test_result.txt");

    "Content of /tmp/test_result.xml:$(const.n)"
```

```
        printfile => cat("/tmp/test_result.xml");
}
```

```
R:
not ok Checking to see if 'NAMESPACE_CLASS' is defined
R:
ok Checking to see if 'BUNDLE_CLASS' is defined
R: Content of /tmp/test_result.txt:

R: 1..2
R: 1 not ok Checking to see if 'NAMESPACE_CLASS' is defined
R: 2 ok Checking to see if 'BUNDLE_CLASS' is defined
R: Content of /tmp/test_result.xml:

R: <?xml version="1.0" encoding="UTF-8"?>
R: <testsuite tests="2" failures="1" timestamp="2017-01-20T13:43:26">
R:
R:   <testcase name="BUNDLE_CLASS">Checking to see if 'BUNDLE_CLASS' is defined</testcase>
R:
R:   <testcase name="NAMESPACE_CLASS_failed">
R:     <failure message="'NAMESPACE_CLASS' is *not* defined.">Checking to see if 'NAMESPACE_CLAS
R:   </testcase>
R:
R:
R:
R:
R: </testsuite>
R:
R: <!-- not implemented (yet):
R: 1) errors: <error message="my error message">my crash report</error>
R: 2) STDOUT: <system-out>my STDOUT dump</system-out>
R: 3) STDERR: <system-err>my STDERR dump</system-err>
R:  -->
```

ADDITIONAL RESOURCES

In no particular order:

- Behind the scenes: How do we test CFEngine
- Test dummies on sale!
- Policy testing using TAP
- Testing CFEngine policy by counting classes
- CFEngine Policy Servers with Docker
- Using Vagrant with CFEngine for Development and Testing
- CFEngine Enterprise Vagrant Environment
- Vagrant: Virtual machine provisioning made easy

MASTERFILES ACCEPTANCE TESTS

https://github.com/cfengine/masterfiles/pull/860/files

Created by Nick Anderson.