# Testing CFEngine Policy

Nick Anderson

[dark] **2015-09-29** [light]

CFEngine

# Core Acceptance Tests - Parallel/Serial

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

CFEngine

# Core Acceptance Tests - Timed

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

CFEngine

# Core Acceptance Tests - Fault

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

# Core Acceptance Tests - Network

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

CFEngine

# Writing a test

Tests will automatically run the following bundles if present:

- `init`
- `test`
- `check`
- `destroy` ($(G.testdir) is automatically cleaned up by `testall)

`inputs` in `body common control` should include the path to `default.cf.sub`, and `bundlesequence` should be set to `default("$(this.promise_filename)")`.

**NOTE:** Since the class `ok` is used in most tests, never create a persistent class called `ok` in any test. Persistent classes are cleaned up between test runs, but better safe than sorry.

Output `$(this.promise_filename) Pass` for passing and `$(this.promise_filename) FAIL` for failing.

# 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) );
}
```

CFEngine

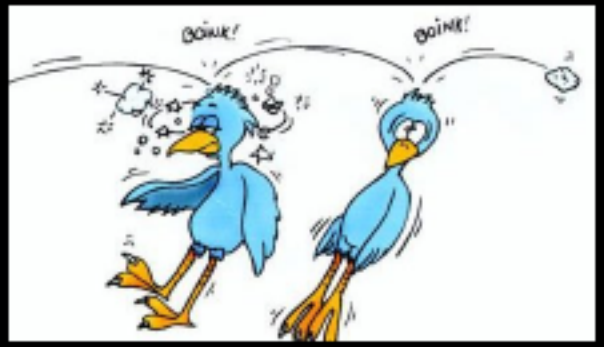
# Running the test

```
$ ./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
```

CFEngine


CFEngine

# Killing 2 birds with one stone




Improve documentation and testing using the test support for examples.

CFEngine

# 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 (for test support) example_output
- Example doc usage
- Example doc result

CFEngine

# Get the presentation source

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

CFEngine

# 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

CFEngine

# 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
    - @cmdln_
    - cmdln.org
    - linkedin.com/in/hithisisnick

**How about you?**

CFEngine

# Why Test?

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, Masterfiles Policy Framework
- **Evolve Thinking:** Evolve Thinking Freelib
- **Normation:** NCF
- **Others:** CFEngine Provisioner for Test Kitchen, Marco Marongiu, Jarle Bjørgeengen

Send me more!

CFEngine

# Core Acceptance Tests

Found in tests/acceptance

- Staged
- Meta Info
- Unsafe
- Parallel/Serial
- Timed
- Fault
- Network

CFEngine

# Core Acceptance Tests - Staged

- 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

# Core Acceptance Tests - Bundle Meta Info

- 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 (usually use this one)
- test_suppress_fail - Runs the test, but will accept failure. Use this when there is a real failure, but it is acceptable for the time being. This variable requires a meta tag on the variable set to "redmine", where is a Redmine issue number. There is a subtle difference between the two. Soft failures will not be reported as a failure in the XML output, and is appropriate for test cases that document incoming bug reports. Suppressed failures will count as a failure, but it won't block the build, and is appropriate for regressions or bad test failures.

**Note:** If you are writing an acceptance test for a (not yet fixed) bug in Redmine, use **test_soft_fail**.

# Core Acceptance Tests - Bundle Meta Info Example

```
1  bundle agent test
2  {
3    meta:
4      "test_soft_fail"
5        string => "any",           # Class expression describing platforms (hard classes)
6        meta => { "redmineXXX" };
7  }
```

CFEngine

# Core Acceptance Tests - Unsafe

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

CFEngine