Objectives:

- Writing an integration test.

- Develop a cookbook with a test-driven approach using the same test cases.

To create integration tests that use Chef recipes, and follows a TDD approach, the apache cookbook will be updated to build a web server by:

- Installing the apache2 package

- Writing a test page

- Starting and enabling the apache2 service

1. In order to test the apache cookbook, the default_test.rb file will be updated

    a. Chef generator also created an example specification (or spec) file.

    b. All integration tests exist in following directory:

```
/home/ubuntu/chef-repo/cookbooks/apache/test/integration/default
```

i. Test Kitchen looks for tests to run under this directory. This corresponds to the values specified in the suites section of the Test Kitchen configuration file (kitchen.yml).

ii. **test/integration/default/default_test.rb** is a file written in Ruby. Because Chef is a domain specific language (DSL), any logic used in this file needs to follow Ruby constructs.

iii. Open default_test.rb:

```
# InSpec test for recipe apache::default

# The InSpec reference, with examples and extensive documentation, can be
# found at https://docs.chef.io/inspec/resources/

unless os.windows?
  # This is an example test, replace with your own test.
  describe user('root'), :skip do
    it { should exist }
  end
end

# This is an example test, replace it with your own test.
describe port(80), :skip do
  it { should_not be_listening }
end
```

iv. The **default test** has two parameters:

- Check for a user resource (**Using Inspec Resource**) named 'root' on the test instance.
- An expectation that the server should not be listening on port 80

v. Within the block any number of **expectations** can be defined regarding a particular **resource**.

By default all operating systems will be examined. So this example will be evaluated and executed against every operating system.

2. Modify the test cases by updating default_test.rb with the following:

    a. Remove the first test case

    b. Make sure apache is listening on port 80

    c. Remove the keyword 'skip'

    d. Add a new test to validate the working website

Updated default_test.rb file:

```
describe port(80) do
  it { should be_listening }
end

describe command('curl http://localhost') do
  its(:stdout) { should match(/Welcome Home/) }
end
```

File Details:

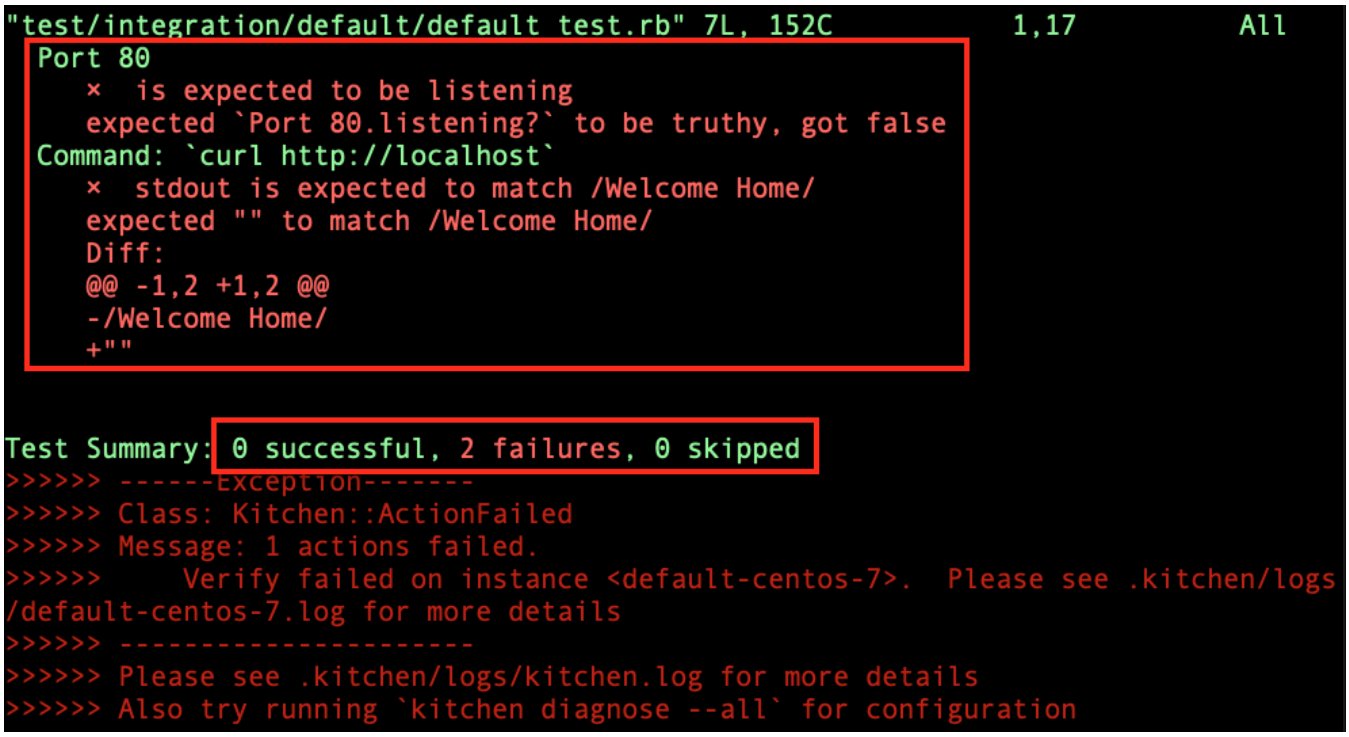
➔ The first test checks that the server is listening on port 80 for incoming connections. This test stops at the ‘end’ of the ‘do’ block

➔ The second test validates that the message ‘Welcome Home’ is displayed.

3. Executing tests using kitchen verify:

   a. Execute `$kitchen verify`.

   The expected output is for 2 test cases to fail:



Failure breakdown:

1. The first test failure states that there was an expectation to be able to listen on port 80 and was unable to do so.
2. The second test failure states that the URL ‘http:localhost’ is expected to have the standard output of ‘Welcome Home’ and instead has an output of an empty string (“”).

Notify your instructor that you are done with the lab

END OF LAB