Objectives:

- Define expectations for multiple platforms
- Implement a multi-platform cookbook

1. In the **specification file** (spec/unit/recipes/default_spec.rb)**,** find the examples for the **Ubuntu** platform as shown below:

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on Ubuntu 20.04' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/master/PLATFORMS.md
    platform 'ubuntu', '20.04'

    it 'converges successfully' do
      expect { chef_run }.to_not raise_error
    end
  end
```

2. The Apache service in Ubuntu is called Apache2, not httpd. It is necessary to add the block shown below to ensure apache's default recipe installs apache service correctly on ubuntu.

```
describe 'apache::default' do
  context 'When all attributes are default, on Ubuntu 20.04' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/master/PLATFORMS.md
    platform 'ubuntu', '20.04'

    it 'converges successfully' do
      expect { chef_run }.to_not raise_error
    end

    it 'installs the apache2 package' do
      expect(chef_run).to install_package('apache2')
    end
```

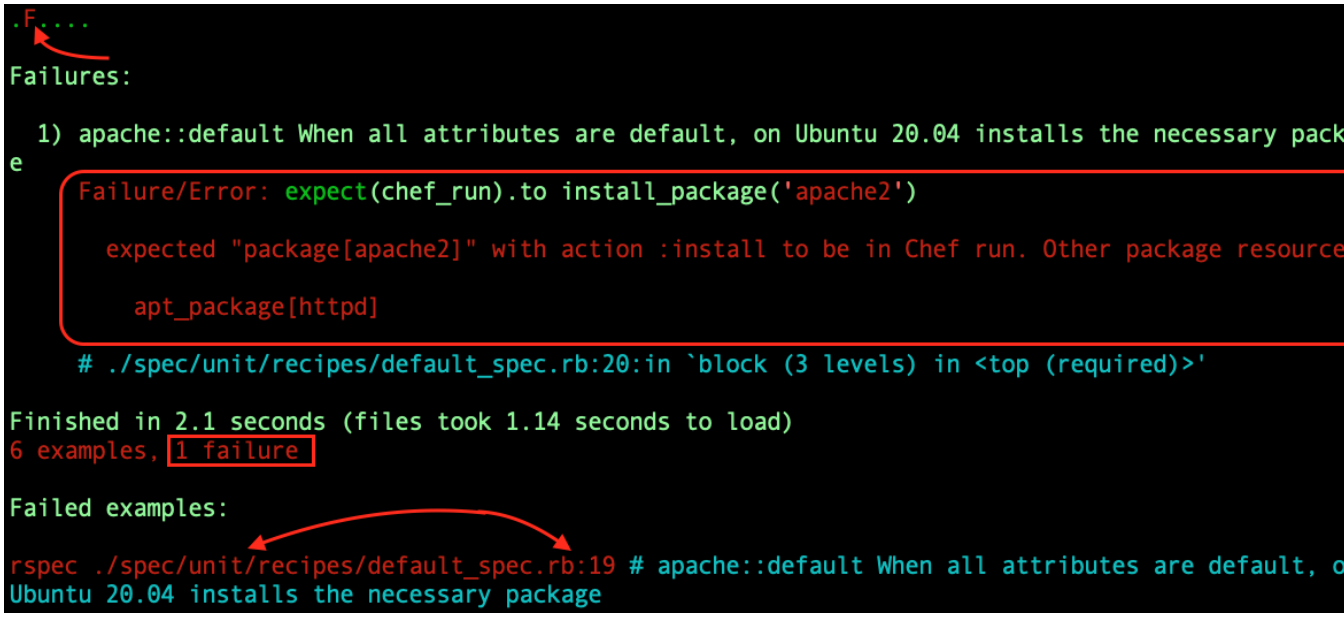
3. After including the new block, execute unit tests with rspec:

   Expected Output :

   When the tests are executed, defining the new platform will not raise an error when it converges, but will raise an error when the expected installed package is ‘httpd’ instead of ‘apache2’.

   It states that the apache2 installation on ubuntu failed, which means if this recipe executes on a platform operating on Ubuntu, this recipe will not work as expected.

   At the bottom of the output, rspec lists the file path of the line and file throwing the error.



4. Update the **attributes/default.rb** file to specify the **package** name depending on whether it is running on a **Ubuntu** or **CentOS** machine.

   a. **Node attributes** will be used to identify the **operating system**.

   b. The node attribute is **conditionally** based on the **platform**

   c. As Ruby is the used as the basis for Chef’s **domain specific language** (DSL), the **conditional** will be written in a context that Ruby recognizes

d. The case statement is the conditional to be used in this example. Please view Chef's documentation on using a case statement for a better understanding: https://docs.chef.io/ruby/#case

e. Update attributes/default.rb with the following case statement :

```
case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
else
  default['apache']['package_name'] = 'httpd'
end

default['apache']['service_name'] = 'httpd'
default['apache']['default_index_html'] = '/var/www/html/index.html'
```

5. Execute ChefSpec:

```
$ chef exec rspec spec/unit/recipes/default_spec.rb
```

Successful Output:

```
......

Finished in 1.66 seconds (files took 1.12 seconds to load)
6 examples, 0 failures
```

6. Updating the service name for Ubuntu:

a. Change the specification file to test for starting and enabling apache2.

b. Test with ChefSpec

c. Address the errors.

d. Update the attributes file by adding to the case statement

7. Update specification file as depicted below:

```
describe 'apache::default' do
  context 'When all attributes are default, on Ubuntu 20.04' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/master/PLATFORMS.md
    platform 'ubuntu', '20.04'

    it 'converges successfully' do
      expect { chef_run }.to_not raise_error
    end

    it 'installs the apache2 package' do
      expect(chef_run).to install_package('apache2')
    end

    it 'starts apache2' do
      expect(chef_run).to start_service('apache2')
    end

    it 'starts apache2' do
      expect(chef_run).to enable_service('apache2')
    end
  end
```

8. Execute ChefSpec:

Expected output:

```
..FF....

Failures:

  1) apache::default When all attributes are default, on Ubuntu 20.04 sta
     Failure/Error: expect(chef_run).to start_service('apache2')

       expected "service[apache2]" with action :start to be in Chef run.

         service[httpd]

     # ./spec/unit/recipes/default_spec.rb:24:in `block (3 levels) in <to

  2) apache::default When all attributes are default, on Ubuntu 20.04 sta
     Failure/Error: expect(chef_run).to enable_service('apache2')

       expected "service[apache2]" with action :enable to be in Chef run.

         service[httpd]

     # ./spec/unit/recipes/default_spec.rb:28:in `block (3 levels) in <to

Finished in 2.17 seconds (files took 1.15 seconds to load)
8 examples, 2 failures
```

9. Update attributes/default.rb to support Ubuntu platform.

10. Remove the line with `default['apache']['service_name'] = 'httpd'` and update the 'platform' case statement with the following:

```
case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
else
  default['apache']['package_name'] = 'httpd'
  default['apache']['service_name'] = 'httpd'
end

default['apache']['default_index_html'] = '/var/www/html/index.html'
```

11. Execute `$chef exec rspec`

    Note: The shortened command for rspec finds EVERY spec file in the current directory and every sub-directory. There is only one spec file in the current directory.

```
........

Finished in 1.96 seconds (files took 1.18 seconds to load)
8 examples, 0 failures
```

12. Rendering a file and checking for its content

    a. Add another 'do' block to render an index.html file and check the content of the file.

```
it 'starts apache2' do
  expect(chef_run).to enable_service('apache2')
end

it 'creates the index.html' do
  expect(chef_run).to render_file('/var/www/html/index.html').with_content('<h1>Welcome Home!</h1>')
end
```

13. Execute rspec:

```
.........

Finished in 2.08 seconds (files took 1.17 seconds to load)
9 examples, 0 failures
```

*Note : The **attributes file** did **not** need to be modified to avoid a test failure because the creation and content of the **index.html** file will be the same **irrespective** of the **Operating System**.*

14. Updating kitchen integration tests to execute on an Ubuntu Operating System.

    a. Update **kitchen.yml** to the following:

```
platforms:
  - name: centos-7
    driver:
      image: dokken/centos-7
      privileged: true
      pid_one_command: /usr/lib/systemd/systemd
      volumes:
        - /sys/fs/cgroup:/sys/fs/cgroup:ro # required by systemd
  - name: ubuntu-20.04
    driver:
      image: dokken/ubuntu-20.04
      privileged: true
      pid_one_command: /usr/lib/systemd/systemd
      volumes:
        - /sys/fs/cgroup:/sys/fs/cgroup:ro # required by systemd
```

15. Converge the kitchen

16. Destroy the kitchen:

```
$kitchen destroy
```

Expected output:

```
-----> Starting Test Kitchen (v2.10.0)
-----> Destroying <default-centos-7>...
       Deleting kitchen sandbox at /home/ubuntu/.dokken/kitchen_sandbox/fa3b5de57d-default-centos-7
       Deleting verifier sandbox at /home/ubuntu/.dokken/verifier_sandbox/fa3b5de57d-default-centos-7
       Finished destroying <default-centos-7> (0m10.39s).
-----> Destroying <default-ubuntu-2004>...
       Deleting kitchen sandbox at /home/ubuntu/.dokken/kitchen_sandbox/fa3b5de57d-default-ubuntu-2004
       Deleting verifier sandbox at /home/ubuntu/.dokken/verifier_sandbox/fa3b5de57d-default-ubuntu-2004
       Finished destroying <default-ubuntu-2004> (0m10.48s).
-----> Test Kitchen is finished. (0m21.77s)
```

17. List the existing kitchens to see the new environment

```
$kitchen list
```

18. The expected output confirms that the kitchen was destroyed in the previous step by showing the 'Last Action' column as 'Not Created' for both kitchen instances :

```
Instance             Driver  Provisioner  Verifier  Transport  Last Action    La
default-centos-7     Dokken  Dokken       Inspec    Dokken     <Not Created>  <N
default-ubuntu-2004  Dokken  Dokken       Inspec    Dokken     <Not Created>  <N
```

a. Run tests by executing the `$kitchen verify` command:

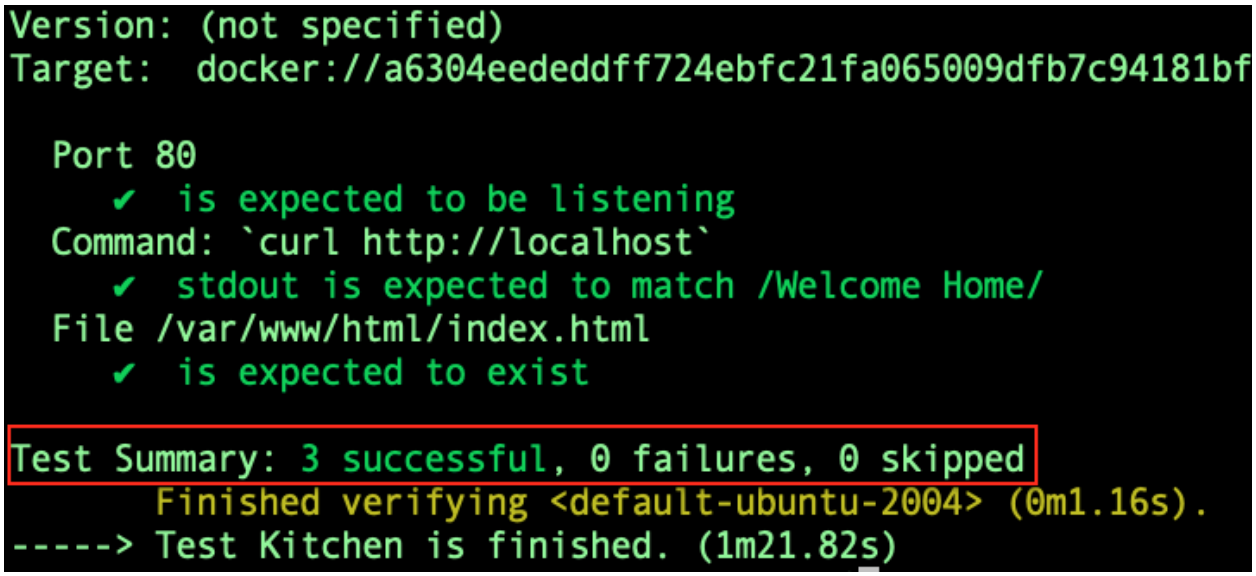
*Note: because the kitchen was **destroyed** in the last step, kitchen verify will automatically run kitchen **create** and kitchen **converge** after checking if the kitchen exists, **then** it will run kitchen verify*

Expected Output:

```
Target:  docker://8d9f5028e016dd161af34c5550ae166cbe1d4c

  Port 80
     ✔  is expected to be listening
  Command: `curl http://localhost`
     ✔  stdout is expected to match /Welcome Home/
  File /var/www/html/index.html
     ✔  is expected to exist

Test Summary: 3 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.85s).
```



Notify your instructor that you are done with the lab

END OF LAB