Lab 02, Objectives:

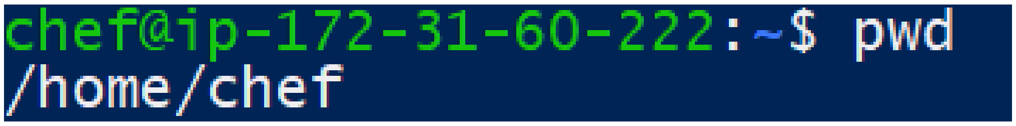
- *Generate a Chef repo*
- *Create a web server cookbook*
- *Configure Test Kitchen*
- *Create a virtual testing environment*

1. Login to the machine provided to you with the following command:

   $ ssh chef@IP_ADDRESS

2. Print your working directory (show what directory you are in):

   $ pwd

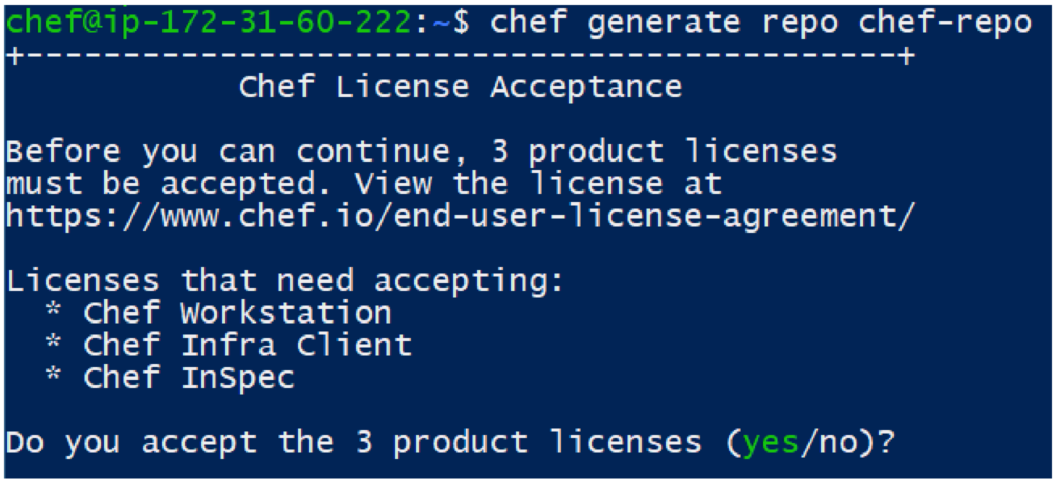


```
chef@ip-172-31-60-222:~$ pwd
/home/chef
```

3. To create a base repo, execute the command below:

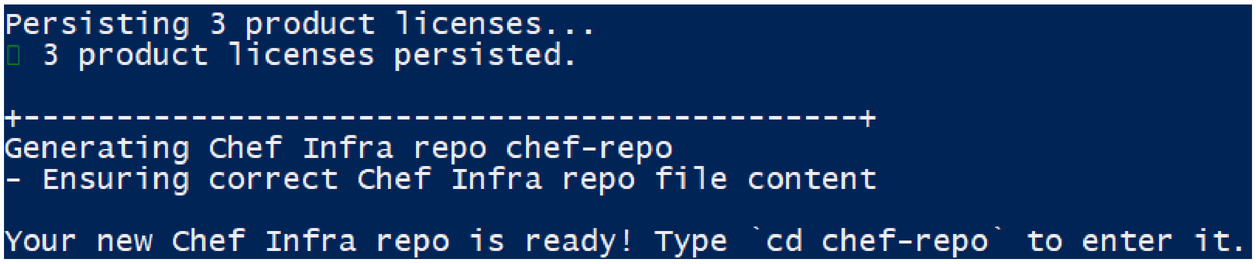
   $ chef generate repo chef-repo

Note: This is the first use of a chef command, so you may be prompted to accept a license first, if so type 'yes'



```
chef@ip-172-31-60-222:~$ chef generate repo chef-repo
+---------------------------------------------+
            Chef License Acceptance

Before you can continue, 3 product licenses
must be accepted. View the license at
https://www.chef.io/end-user-license-agreement/

Licenses that need accepting:
  * Chef Workstation
  * Chef Infra Client
  * Chef InSpec

Do you accept the 3 product licenses (yes/no)?
```

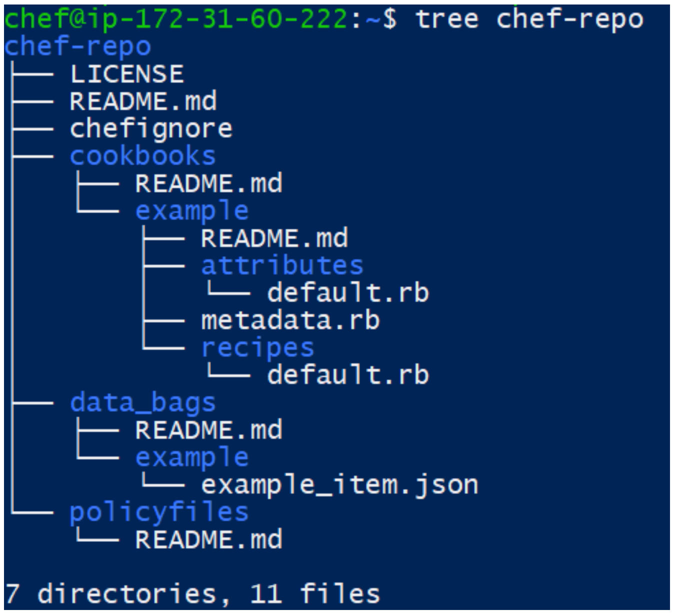
4. Once accepted, you will see following message:



```
Persisting 3 product licenses...
 3 product licenses persisted.

+---------------------------------------------+
Generating Chef Infra repo chef-repo
- Ensuring correct Chef Infra repo file content

Your new Chef Infra repo is ready! Type `cd chef-repo` to enter it.
```

5. View the directory structure by executing the tree command:

$ tree chef-repo



```
chef@ip-172-31-60-222:~$ tree chef-repo
chef-repo
├── LICENSE
├── README.md
├── chefignore
├── cookbooks
│   ├── README.md
│   └── example
│       ├── README.md
│       ├── attributes
│       │   └── default.rb
│       ├── metadata.rb
│       └── recipes
│           └── default.rb
├── data_bags
│   ├── README.md
│   └── example
│       └── example_item.json
└── policyfiles
    └── README.md

7 directories, 11 files
```

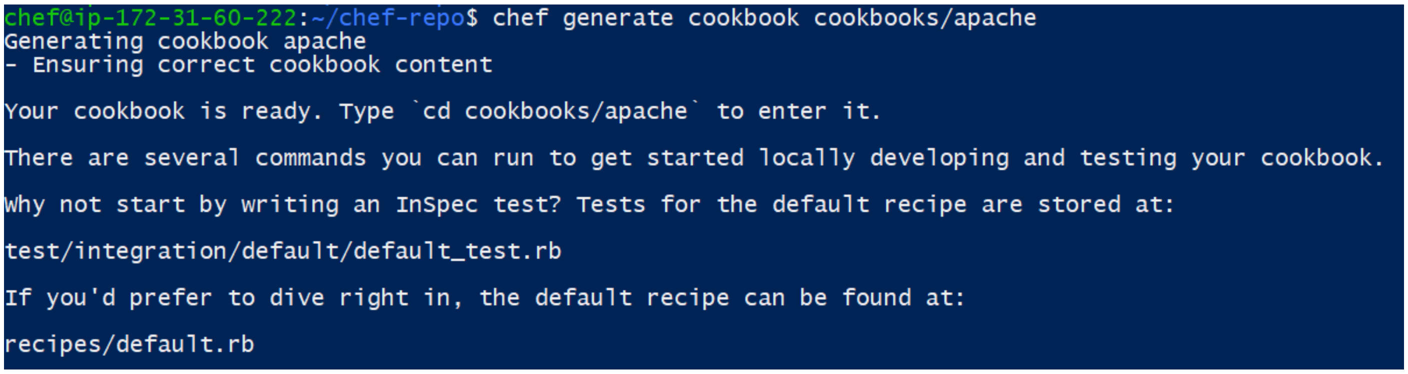
Note: While you can manually create a directory called chef-repo, using the command `**chef generate repo chef-repo**` creates a standardized directory structure that is very useful when using Chef.

6. Change to the 'chef-repo' directory and use the 'chef' command to create a new cookbook named 'apache':

$ cd chef-repo/

$ chef generate cookbook cookbooks/apache

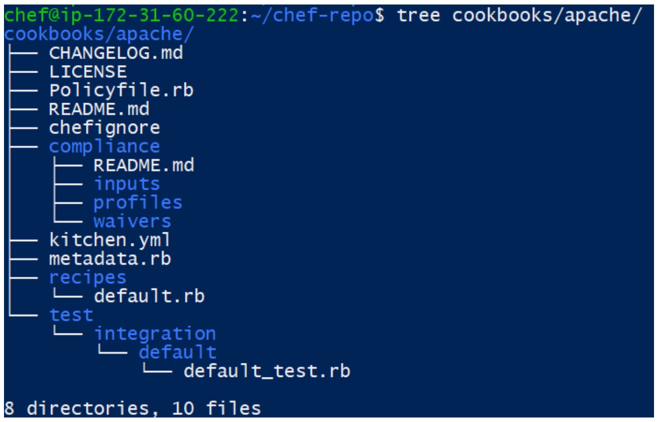
The following output will be displayed:



```
chef@ip-172-31-60-222:~/chef-repo$ chef generate cookbook cookbooks/apache
Generating cookbook apache
- Ensuring correct cookbook content

Your cookbook is ready. Type `cd cookbooks/apache` to enter it.

There are several commands you can run to get started locally developing and testing your cookbook.

Why not start by writing an InSpec test? Tests for the default recipe are stored at:

test/integration/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:

recipes/default.rb
```

7. The contents of the cookbook that Chef generated can now be examined in a file tree format using the "tree" command:

$ tree cookbooks/apache/

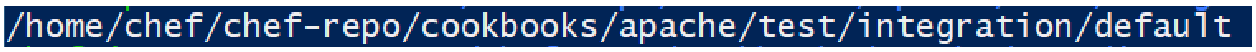


```
chef@ip-172-31-60-222:~/chef-repo$ tree cookbooks/apache/
cookbooks/apache/
├── CHANGELOG.md
├── LICENSE
├── Policyfile.rb
├── README.md
├── chefignore
├── compliance
│   ├── README.md
│   ├── inputs
│   ├── profiles
│   └── waivers
├── kitchen.yml
├── metadata.rb
├── recipes
│   └── default.rb
└── test
    └── integration
        └── default
            └── default_test.rb

8 directories, 10 files
```

8. Integration tests are created in the following directory:

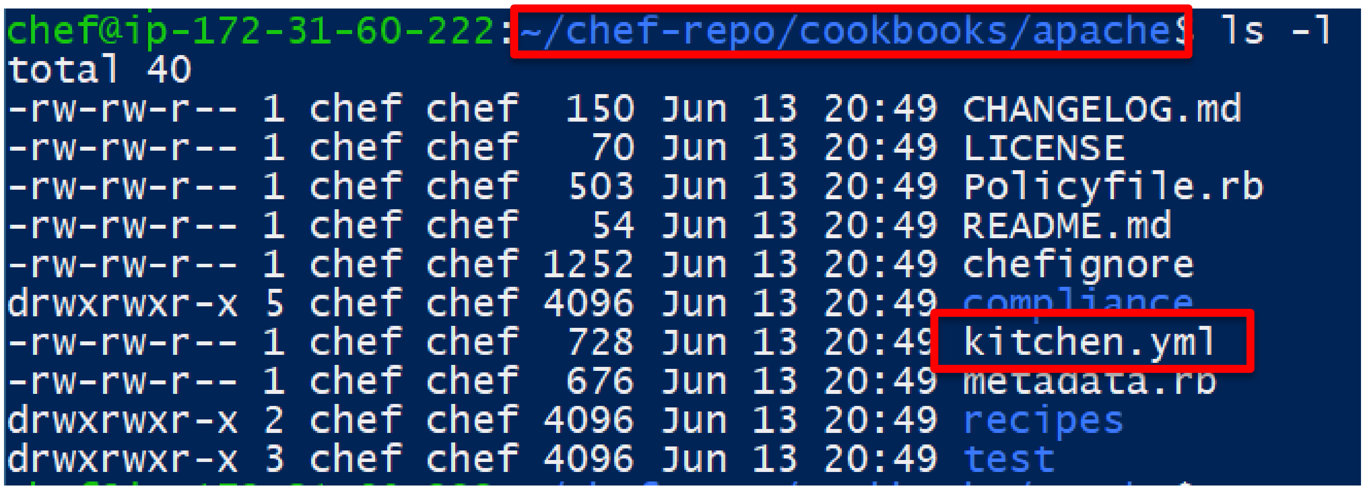
chef-repo/cookbooks/<cookbook_name>/test/integration/default/default_test.rb

In our example of the apache cookbook, the file path will be:



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

9. Move into your apache cookbook. There you will find **kitchen.yml**. This is the configuration file for Test Kitchen.



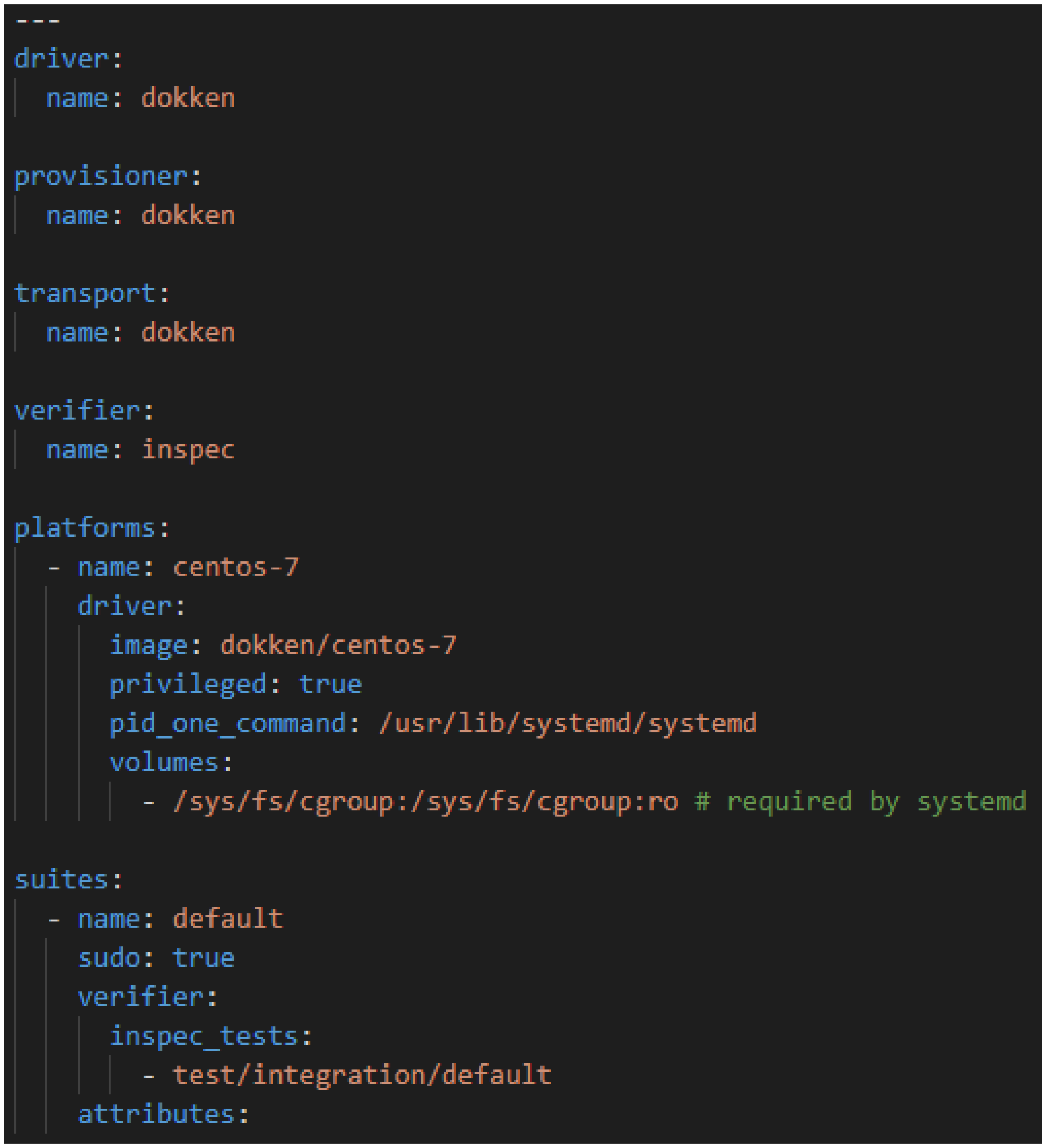
```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ ls -l
total 40
-rw-rw-r-- 1 chef chef  150 Jun 13 20:49 CHANGELOG.md
-rw-rw-r-- 1 chef chef   70 Jun 13 20:49 LICENSE
-rw-rw-r-- 1 chef chef  503 Jun 13 20:49 Policyfile.rb
-rw-rw-r-- 1 chef chef   54 Jun 13 20:49 README.md
-rw-rw-r-- 1 chef chef 1252 Jun 13 20:49 chefignore
drwxrwxr-x 5 chef chef 4096 Jun 13 20:49 compliance
-rw-rw-r-- 1 chef chef  728 Jun 13 20:49 kitchen.yml
-rw-rw-r-- 1 chef chef  676 Jun 13 20:49 metadata.rb
drwxrwxr-x 2 chef chef 4096 Jun 13 20:49 recipes
drwxrwxr-x 3 chef chef 4096 Jun 13 20:49 test
```

In our examples we will be using the Dokken driver and provisioner. The Dokken provisioner will use docker to execute test cases on behalf of Test Kitchen

10. Update **kitchen.yml** file as follows:

Note: yml files need exact indentation. Only use spaces, not tabs. Replicate the file precisely as seen below

```
---
driver:
  name: dokken

provisioner:
  name: dokken

transport:
  name: dokken

verifier:
  name: inspec

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

suites:
  - name: default
    sudo: true
    verifier:
      inspec_tests:
        - test/integration/default
    attributes:
```

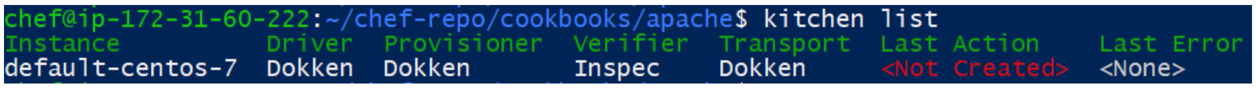


```yaml
---
driver:
  name: dokken

provisioner:
  name: dokken

transport:
  name: dokken

verifier:
  name: inspec

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

suites:
  - name: default
    sudo: true
    verifier:
      inspec_tests:
        - test/integration/default
    attributes:
```

Note: yml files need exact indentation. Each new level is two spaces, using tab will cause errors. Replicate the file precisely as seen above.

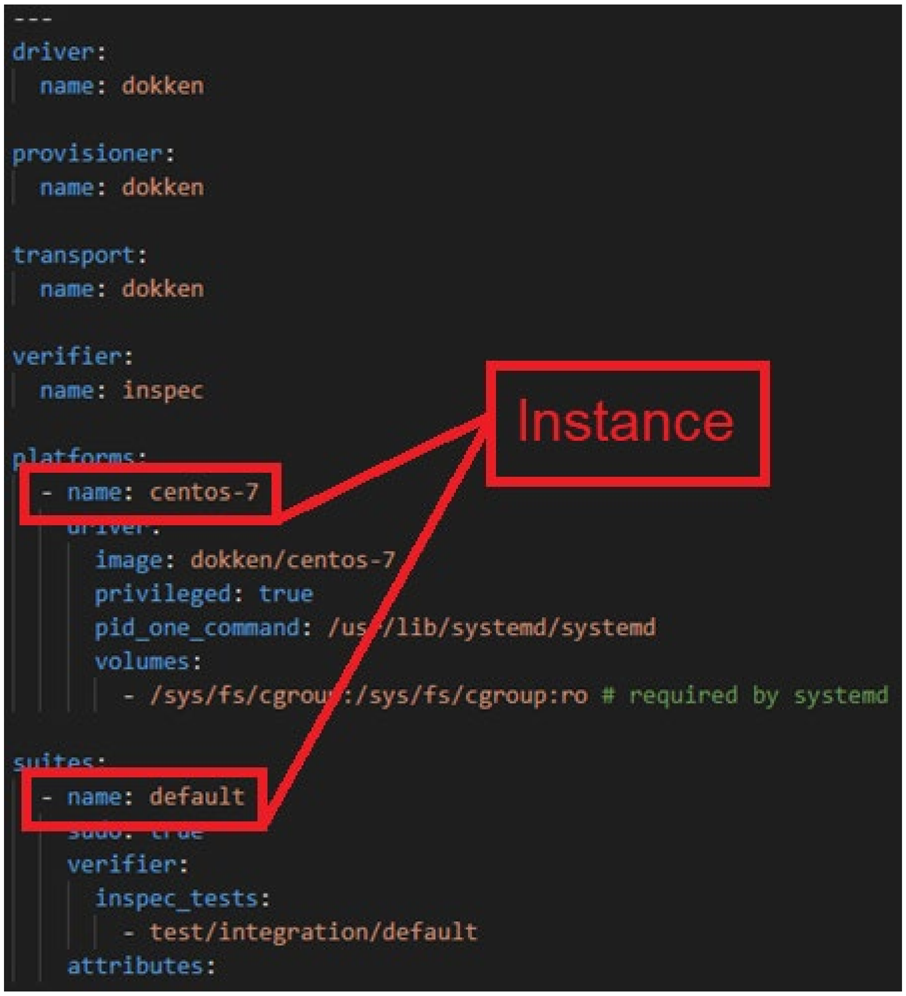
11. List the kitchens that will be created:

$ kitchen list



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ kitchen list
Instance           Driver   Provisioner   Verifier   Transport   Last Action     Last Error
default-centos-7   Dokken   Dokken        Inspec     Dokken      <Not Created>   <None>
```

This shows the list of Kitchen instances that will be created. An instance comprises a single platform (listed in platforms) with a set of testing criteria (listed in suites)



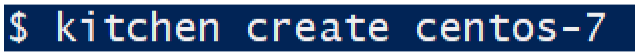
```
---
driver:
  name: dokken

provisioner:
  name: dokken

transport:
  name: dokken

verifier:
  name: inspec

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

suites:
  - name: default
    sudo: true
    verifier:
      inspec_tests:
        - test/integration/default
    attributes:
```

12. Create the environments that were in the **kitchen list** command:

    $ kitchen create

    When the **$ kitchen create** command is executed, a kitchen instance will be created for every instance listed in the table above

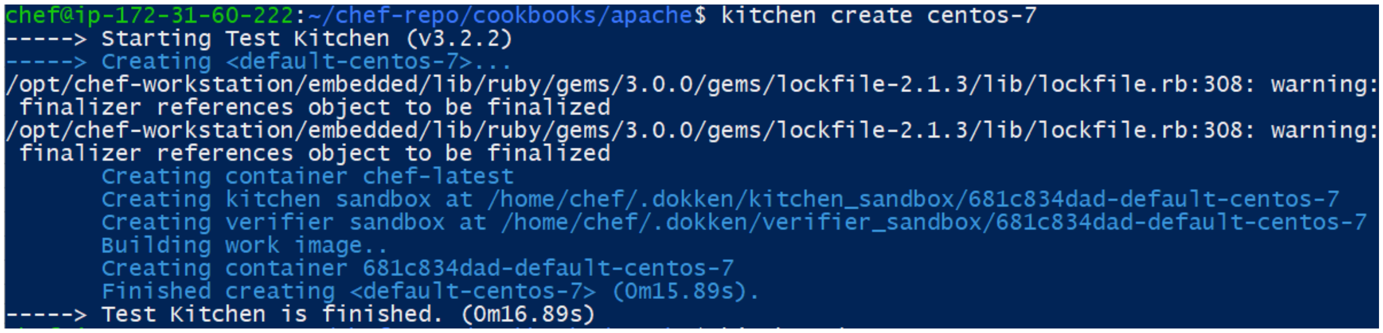
    If you were to execute



```
$ kitchen create centos-7
```

    this would create only one instance using the centos-7 platform

    A successful creation of a kitchen instance will output the following:

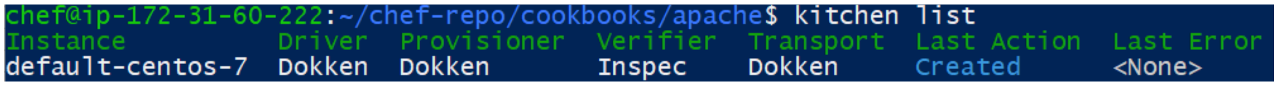


```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ kitchen create centos-7
-----> Starting Test Kitchen (v3.2.2)
-----> Creating <default-centos-7>...
/opt/chef-workstation/embedded/lib/ruby/gems/3.0.0/gems/lockfile-2.1.3/lib/lockfile.rb:308: warning:
 finalizer references object to be finalized
/opt/chef-workstation/embedded/lib/ruby/gems/3.0.0/gems/lockfile-2.1.3/lib/lockfile.rb:308: warning:
 finalizer references object to be finalized
       Creating container chef-latest
       Creating kitchen sandbox at /home/chef/.dokken/kitchen_sandbox/681c834dad-default-centos-7
       Creating verifier sandbox at /home/chef/.dokken/verifier_sandbox/681c834dad-default-centos-7
       Building work image..
       Creating container 681c834dad-default-centos-7
       Finished creating <default-centos-7> (0m15.89s).
-----> Test Kitchen is finished. (0m16.89s)
```

13. Execute the command:

    $ kitchen list

    Under the **'Last Action'** column of the kitchen instances table, it should be **'Created'**



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ kitchen list
Instance          Driver  Provisioner  Verifier  Transport  Last Action  Last Error
default-centos-7  Dokken  Dokken       Inspec    Dokken     Created      <None>
```

14. Install Chef in the kitchen, copy the cookbook(s) over to the kitchen and converge the cookbook(s):

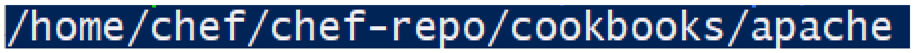
    $ kitchen converge

Converging a kitchen instance will:

a. Create the instance if it has not already been created

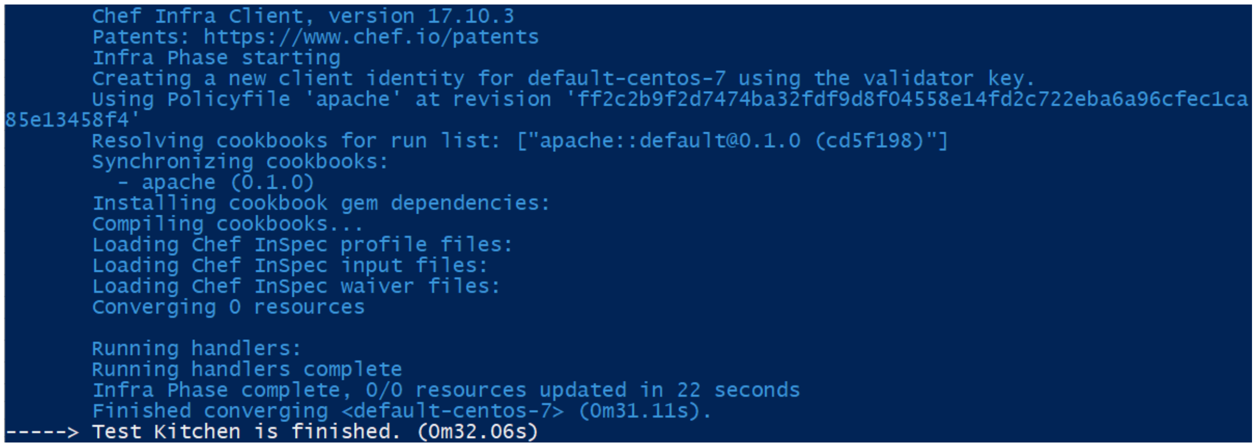
b. Install Chef

c. Apply the cookbook that is at the root of the working directory to that kitchen instance



```
/home/chef/chef-repo/cookbooks/apache
```

d. Expected output from the kitchen converge command:



```
       Chef Infra Client, version 17.10.3
       Patents: https://www.chef.io/patents
       Infra Phase starting
       Creating a new client identity for default-centos-7 using the validator key.
       Using Policyfile 'apache' at revision 'ff2c2b9f2d7474ba32fdf9d8f04558e14fd2c722eba6a96cfec1ca
85e13458f4'
       Resolving cookbooks for run list: ["apache::default@0.1.0 (cd5f198)"]
       Synchronizing cookbooks:
         - apache (0.1.0)
       Installing cookbook gem dependencies:
       Compiling cookbooks...
       Loading Chef InSpec profile files:
       Loading Chef InSpec input files:
       Loading Chef InSpec waiver files:
       Converging 0 resources

       Running handlers:
       Running handlers complete
       Infra Phase complete, 0/0 resources updated in 22 seconds
       Finished converging <default-centos-7> (0m31.11s).
-----> Test Kitchen is finished. (0m32.06s)
```

15. Execute your tests against the state of the instance:

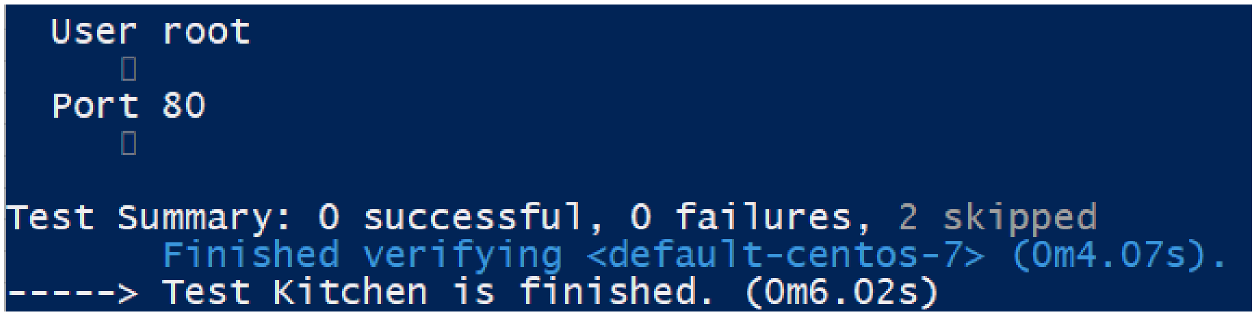
$ kitchen verify

The **$ kitchen verify** command does the following:

a. Create an instance, if needed

b. Converge the instance, if needed

c. Execute a collection of defined tests against the instance



```
  User root
     
  Port 80
     

Test Summary: 0 successful, 0 failures, 2 skipped
       Finished verifying <default-centos-7> (0m4.07s).
-----> Test Kitchen is finished. (0m6.02s)
```

Note: the kitchen **skips** two test cases. These are the default cases that you can see in apache/test/integration/default/defaut_test.rb.

Notice they both have the word "skip" in the test (because they are just sample tests), causing Test Kitchen to ignore them, hence the "2 skipped" notation in the test results

Notify your instructor that you are done with the lab

END OF LAB

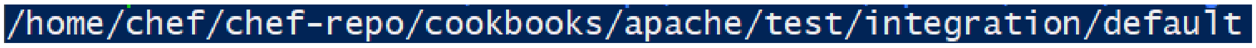
Lab 03, Objectives:

- Write 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:

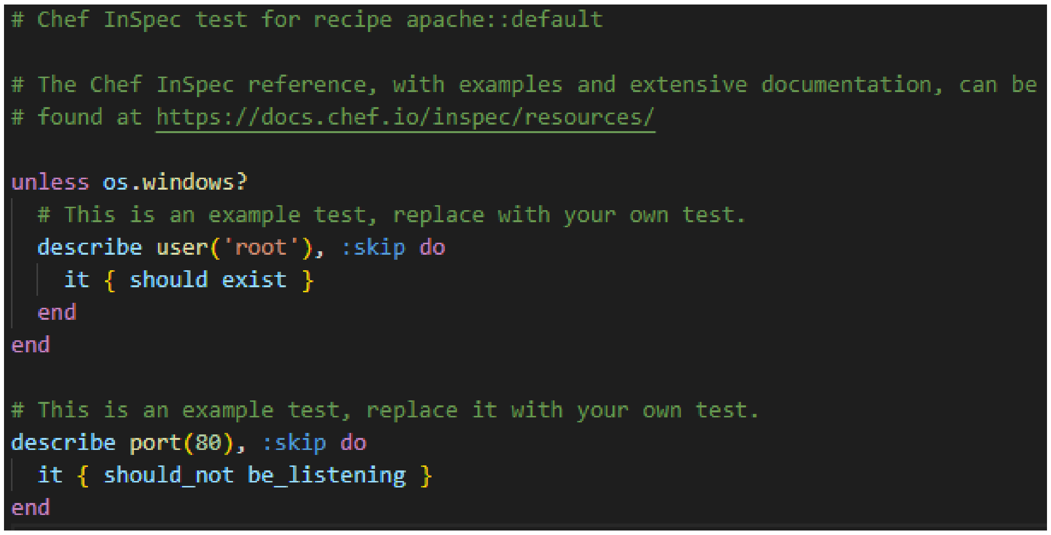
- Updating the **default_test.rb** file in the apache cookbook in order to test the cookbook
- Installing the apache2 package
- Writing a test page
- Starting and enabling the apache2 service

The **chef generate cookbook** command in our previous lab created an example integration test for us. All integration tests exist in following directory:



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

By default, 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). The naming convention for tests is **recipename_test.rb**, i.e. **default_test.rb** is the test for the default recipe. Below is an example of an InSpec test.



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

# The Chef 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
```

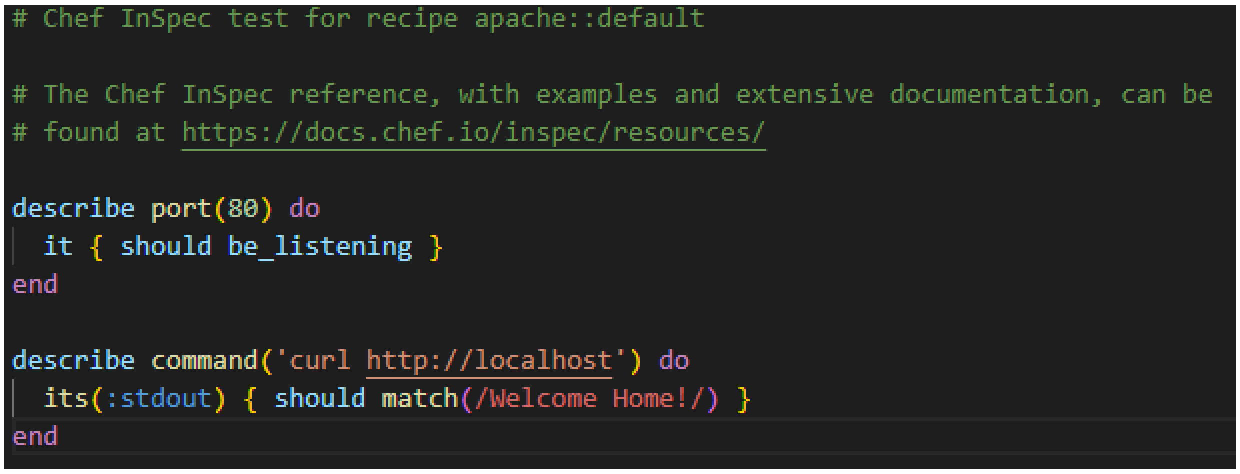
default_test.rb has two sample test blocks:

- Check for the existence of a user resource named 'root'
- Check that the server should not be listening on port 80

Within each block, any number of **expectations** can be defined regarding a particular **resource**

1. Modify the test cases by updating **default_test.rb** with the following:
    a. Remove the first test case
    b. Make sure the server is listening on port 80
    c. Remove the keyword 'skip' from the port 80 test
    d. Add a new test to validate the working website

    default_test.rb file after updates:



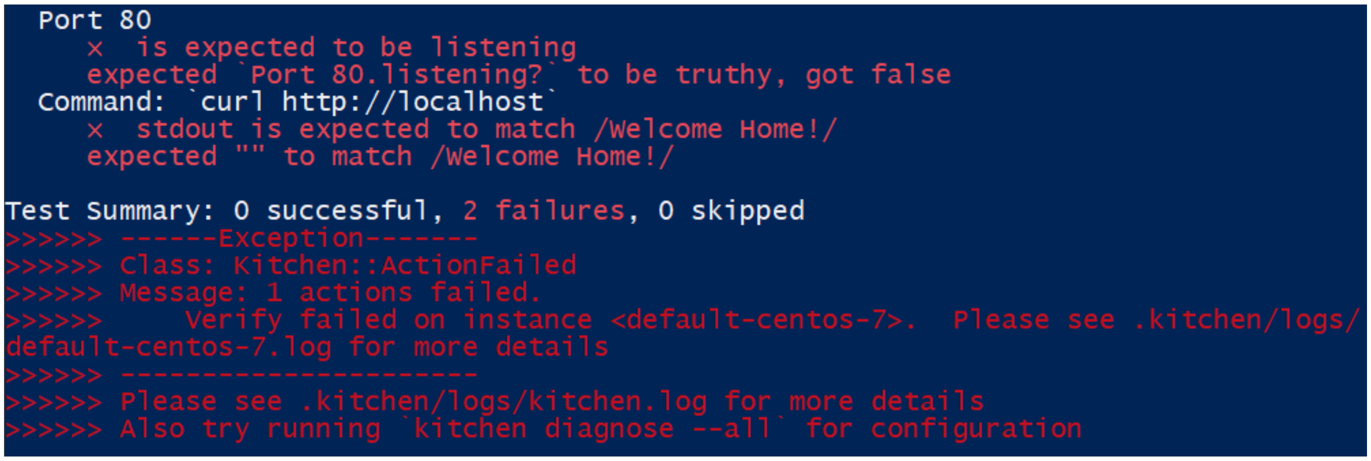
```
# Chef InSpec test for recipe apache::default

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

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

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

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

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

2. Executing tests using kitchen verify:

$ kitchen verify

The expected output is for 2 test cases to fail:



```
  Port 80
     ×  is expected to be listening
     expected `Port 80.listening?` to be truthy, got false
  Command: `curl http://localhost`
     ×  stdout is expected to match /Welcome Home!/
     expected "" to match /Welcome Home!/

Test Summary: 0 successful, 2 failures, 0 skipped
>>>>>> ------Exception-------
>>>>>> Class: Kitchen::ActionFailed
>>>>>> Message: 1 actions failed.
>>>>>>     Verify failed on instance <default-centos-7>.  Please see .kitchen/logs/
default-centos-7.log for more details
>>>>>> ----------------------
>>>>>> Please see .kitchen/logs/kitchen.log for more details
>>>>>> Also try running `kitchen diagnose --all` for configuration
```

Failure breakdown:

The first test failure states that there was an expectation to be able to listen on port 80 and was unable to do so

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

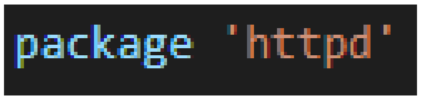
END OF LAB

Lab 04, Objectives:

- Update chef resources in the default recipe to bring the instance to the desired state
- Check desired state by writing test cases

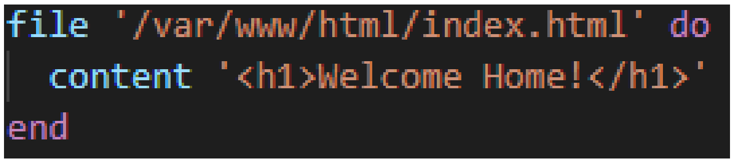
1. Edit the **default.rb** file, found in the directory **/home/chef/chef-repo/cookbooks/apache/recipes** and add the following chef resources:

   a. **Package resource**: to install httpd



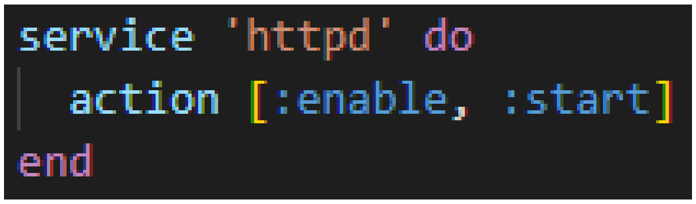
```
package 'httpd'
```

   b. **File resource:** to create the file **"/var/www/html/index.html"**



```
file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h1>'
end
```

   c. **Service resource**: to start and enable service

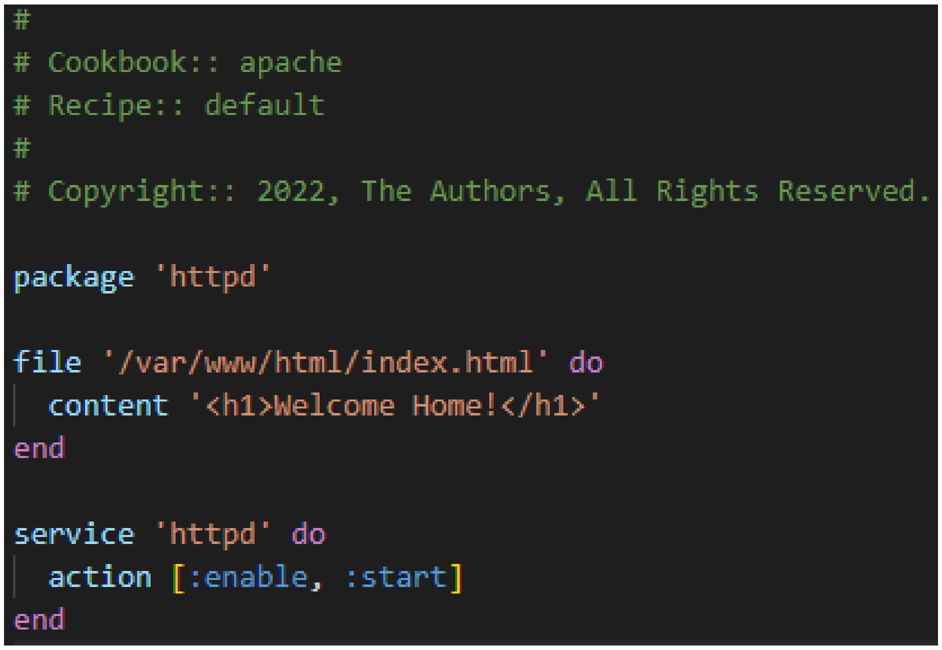


```
service 'httpd' do
  action [:enable, :start]
end
```

2. The **default.rb** file should display the following:

```
# Cookbook:: apache
# Recipe:: default
#
# Copyright:: 2022, The Authors, All Rights Reserved.

package 'httpd'
```

```
file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h1>'
end

service 'httpd' do
  action [:enable, :start]
end
```



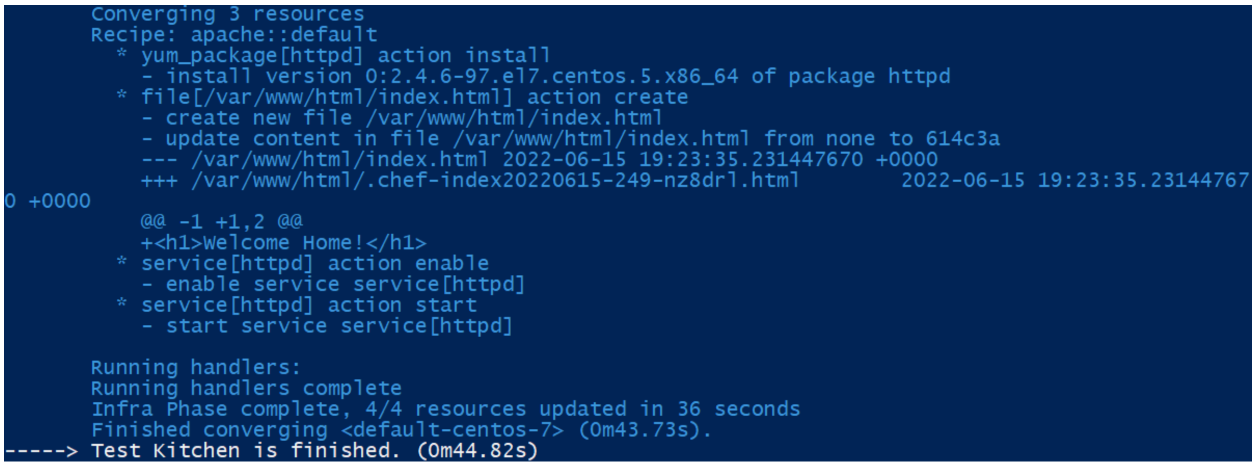
```
#
# Cookbook:: apache
# Recipe:: default
#
# Copyright:: 2022, The Authors, All Rights Reserved.

package 'httpd'

file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h1>'
end

service 'httpd' do
  action [:enable, :start]
end
```

3. Execute the command:

$ kitchen converge

Note: Make sure kitchen commands are executed from the root of the cookbook directory (**~/chef-repo/cookbooks/apache**)

Expected output:

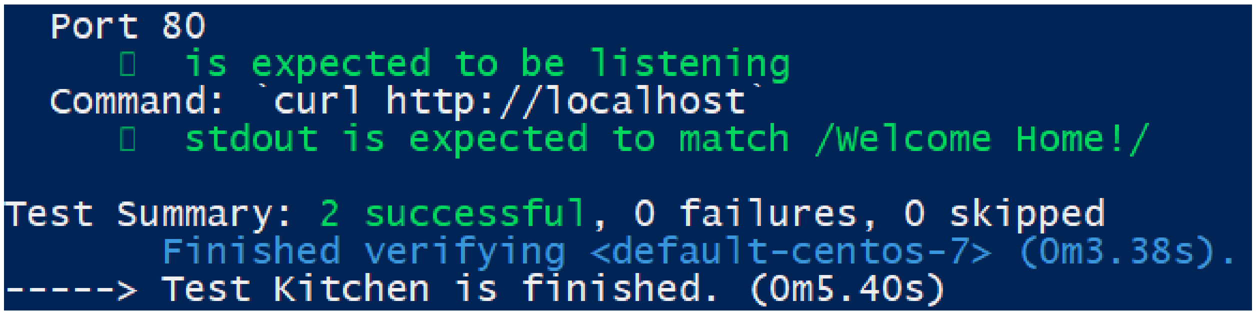


```
       Converging 3 resources
       Recipe: apache::default
         * yum_package[httpd] action install
           - install version 0:2.4.6-97.el7.centos.5.x86_64 of package httpd
         * file[/var/www/html/index.html] action create
           - create new file /var/www/html/index.html
           - update content in file /var/www/html/index.html from none to 614c3a
           --- /var/www/html/index.html 2022-06-15 19:23:35.231447670 +0000
           +++ /var/www/html/.chef-index20220615-249-nz8drl.html        2022-06-15 19:23:35.23144767
0 +0000
           @@ -1 +1,2 @@
           +<h1>Welcome Home!</h1>
         * service[httpd] action enable
           - enable service service[httpd]
         * service[httpd] action start
           - start service service[httpd]

       Running handlers:
       Running handlers complete
       Infra Phase complete, 4/4 resources updated in 36 seconds
       Finished converging <default-centos-7> (0m43.73s).
-----> Test Kitchen is finished. (0m44.82s)
```

4. Execute the command:

$ kitchen verify

Expected output:



```
  Port 80
      is expected to be listening
  Command: `curl http://localhost`
      stdout is expected to match /Welcome Home!/

Test Summary: 2 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.38s).
-----> Test Kitchen is finished. (0m5.40s)
```

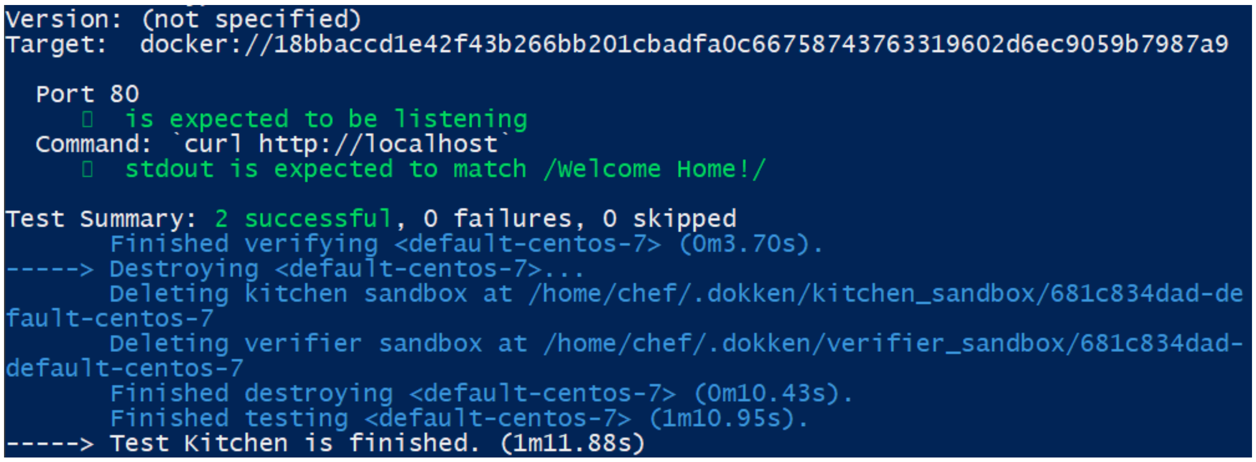
5. Execute the command:

$ kitchen test

This command combines and runs the following:

1. kitchen destroy
2. kitchen create
3. kitchen converge
4. kitchen verify
5. kitchen destroy

Expected output:



```
Version:  (not specified)
Target:   docker://18bbaccd1e42f43b266bb201cbadfa0c66758743763319602d6ec9059b7987a9

  Port 80
     ✓  is expected to be listening
  Command: `curl http://localhost`
     ✓  stdout is expected to match /Welcome Home!/

Test Summary: 2 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.70s).
-----> Destroying <default-centos-7>...
       Deleting kitchen sandbox at /home/chef/.dokken/kitchen_sandbox/681c834dad-de
fault-centos-7
       Deleting verifier sandbox at /home/chef/.dokken/verifier_sandbox/681c834dad-
default-centos-7
       Finished destroying <default-centos-7> (0m10.43s).
       Finished testing <default-centos-7> (1m10.95s).
-----> Test Kitchen is finished. (1m11.88s)
```

6. Update **default_test.rb** to check whether an index.html file (which we created through the default recipe) **exists** or not

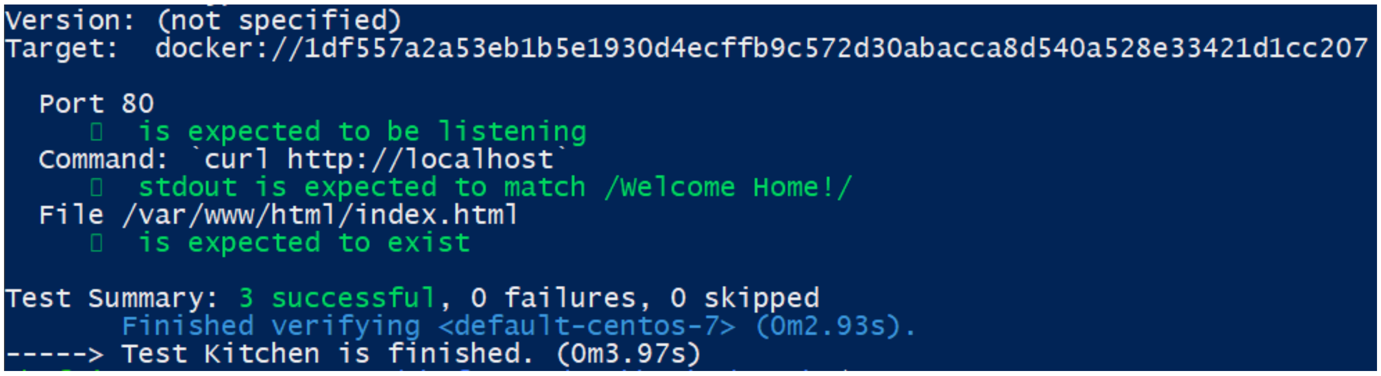
   HINT: Use inspec file resource
   https://docs.chef.io/inspec/resources/file/
   The solution can be found at the end of this lab

What is the next step? How can one make sure that the changes to the test file are working as expected?

7. Execute the command:

   $ kitchen verify

   The test summary should now show 3 successful tests:



```
Version:  (not specified)
Target:   docker://1df557a2a53eb1b5e1930d4ecffb9c572d30abacca8d540a528e33421d1cc207

  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> (0m2.93s).
-----> Test Kitchen is finished. (0m3.97s)
```

Notify your instructor that you are done with the lab

END OF LAB...

...to view the solution, keep on scrolling

. . . .

. . . .

. . . .

. . . .

. . . .

. . . . . .

. . . . .

. . . .

. . .

. .

.

Keep scrolling...

. . . .

. . . .

. . . .

. . . .

. . . .

. . . . . .

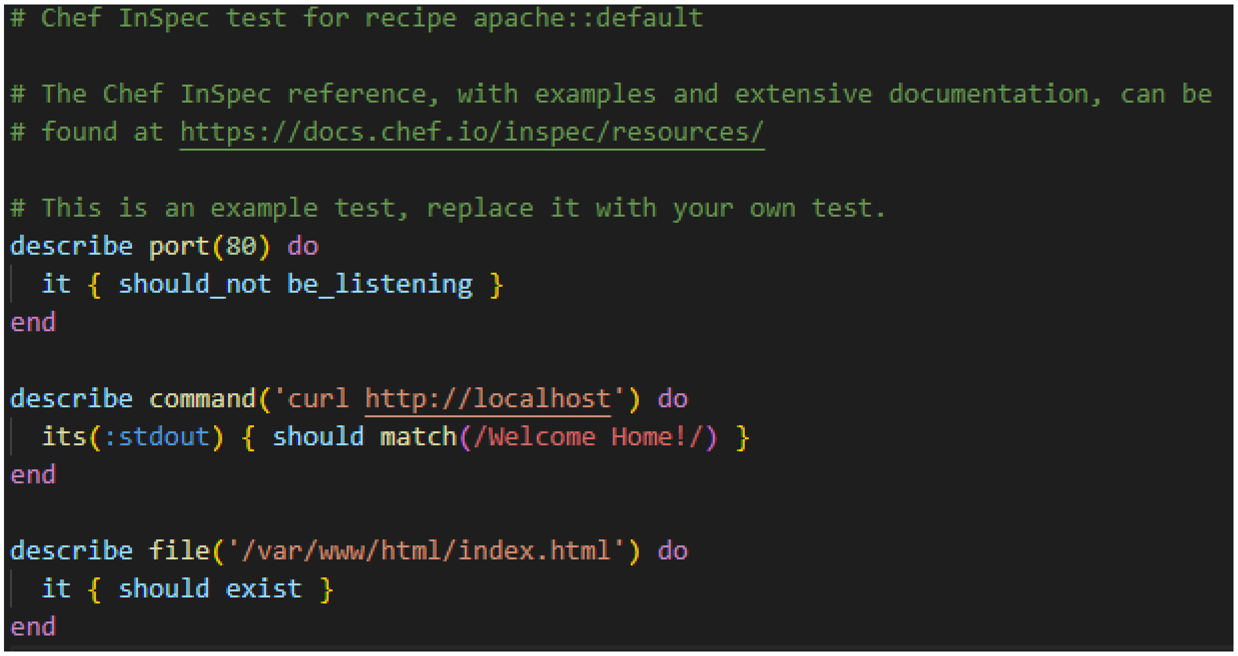
. . . . .

. . . .

.   .   .

.   .

.

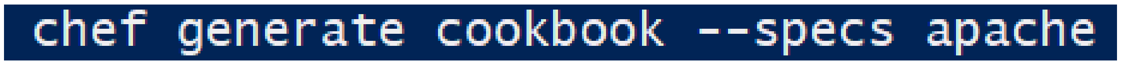


```ruby
# Chef InSpec test for recipe apache::default

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

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

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

describe file('/var/www/html/index.html') do
  it { should exist }
end
```

Lab 05, Objectives:

- Write test cases to test recipes with a faster response time
- Understand the importance and limitations of unit testing
- Verify tests using ChefSpec

1. ChefSpec tests are stored in the **spec/** directory in a cookbook. This directory is not created by default. Execute the following command from the **chef-repo/cookbooks** directory:

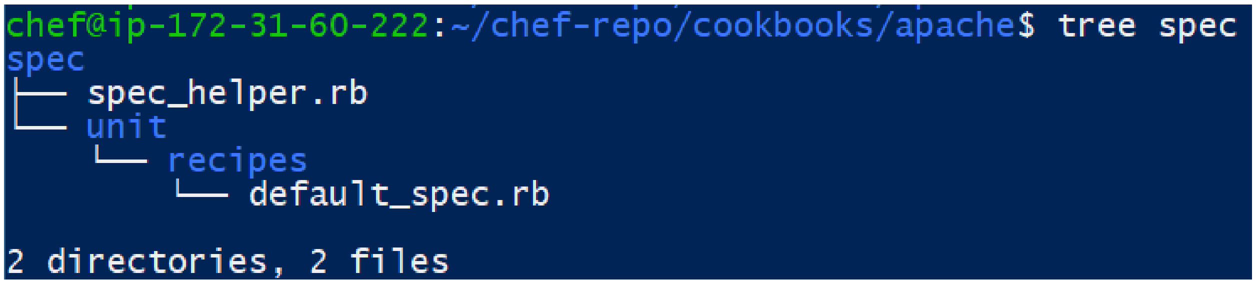


```
chef generate cookbook --specs apache
```

2. Move to the root of the apache cookbook and execute the command:

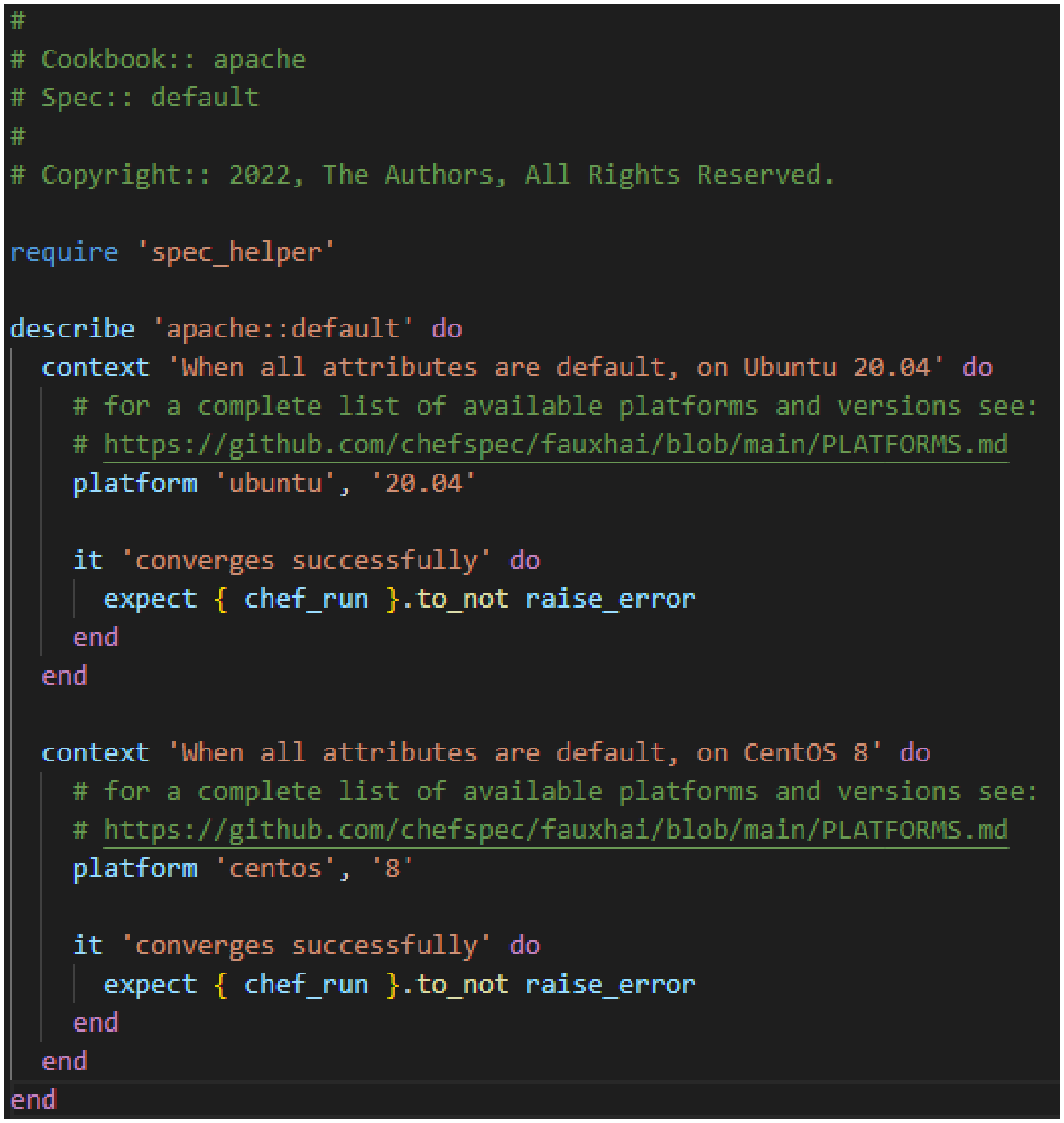
$ tree spec/

Expected output:



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ tree spec
spec
├── spec_helper.rb
└── unit
    └── recipes
        └── default_spec.rb

2 directories, 2 files
```

3. View apache/spec/unit/recipe/default_spec.rb:



```
#
# Cookbook:: apache
# Spec:: default
#
# Copyright:: 2022, The Authors, All Rights Reserved.

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/main/PLATFORMS.md
    platform 'ubuntu', '20.04'

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

  context 'When all attributes are default, on CentOS 8' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/main/PLATFORMS.md
    platform 'centos', '8'

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

Note the two existing tests, showing that it expects the Ubuntu and Centos test environments to **NOT** raise an error

The expectations defined in this initially generated specification file should look a little familiar, because ChefSpec is built on Rspec. Similar to how InSpec is built. ChefSpec requires a little more setup as it creates an in-memory execution

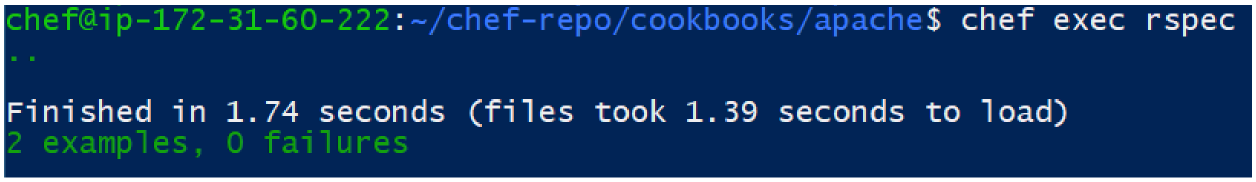
4. From **~/chef-repo/cookbooks/apache** execute:

   $ chef exec rspec

   Note: **spec/unit/recipes/default_spec.rb** is used as the default when a ChefSpec test is not specified.

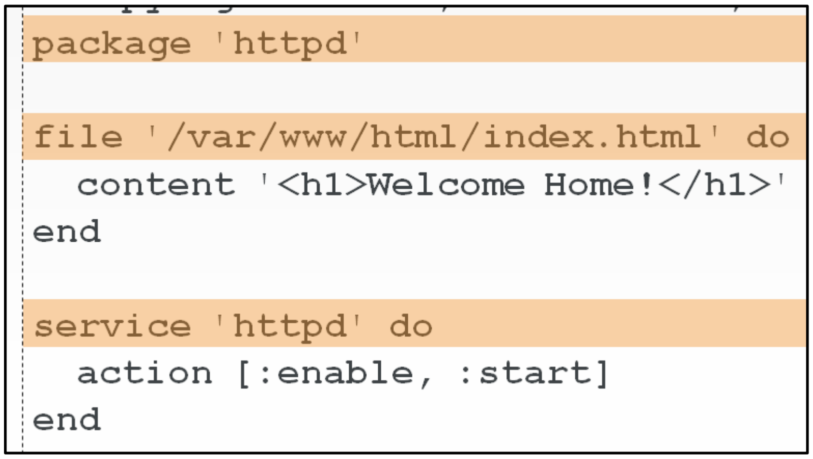
   Expected output:

   The two tests (showing that Ubuntu and Centos test environments do not raise an error) are passing



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..

Finished in 1.74 seconds (files took 1.39 seconds to load)
2 examples, 0 failures
```

   There are 4 resources in the recipe (package, file, service enable and service start). We need to write unit tests for each of these 4 resources:



```
package 'httpd'

file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h1>'
end

service 'httpd' do
  action [:enable, :start]
end
```

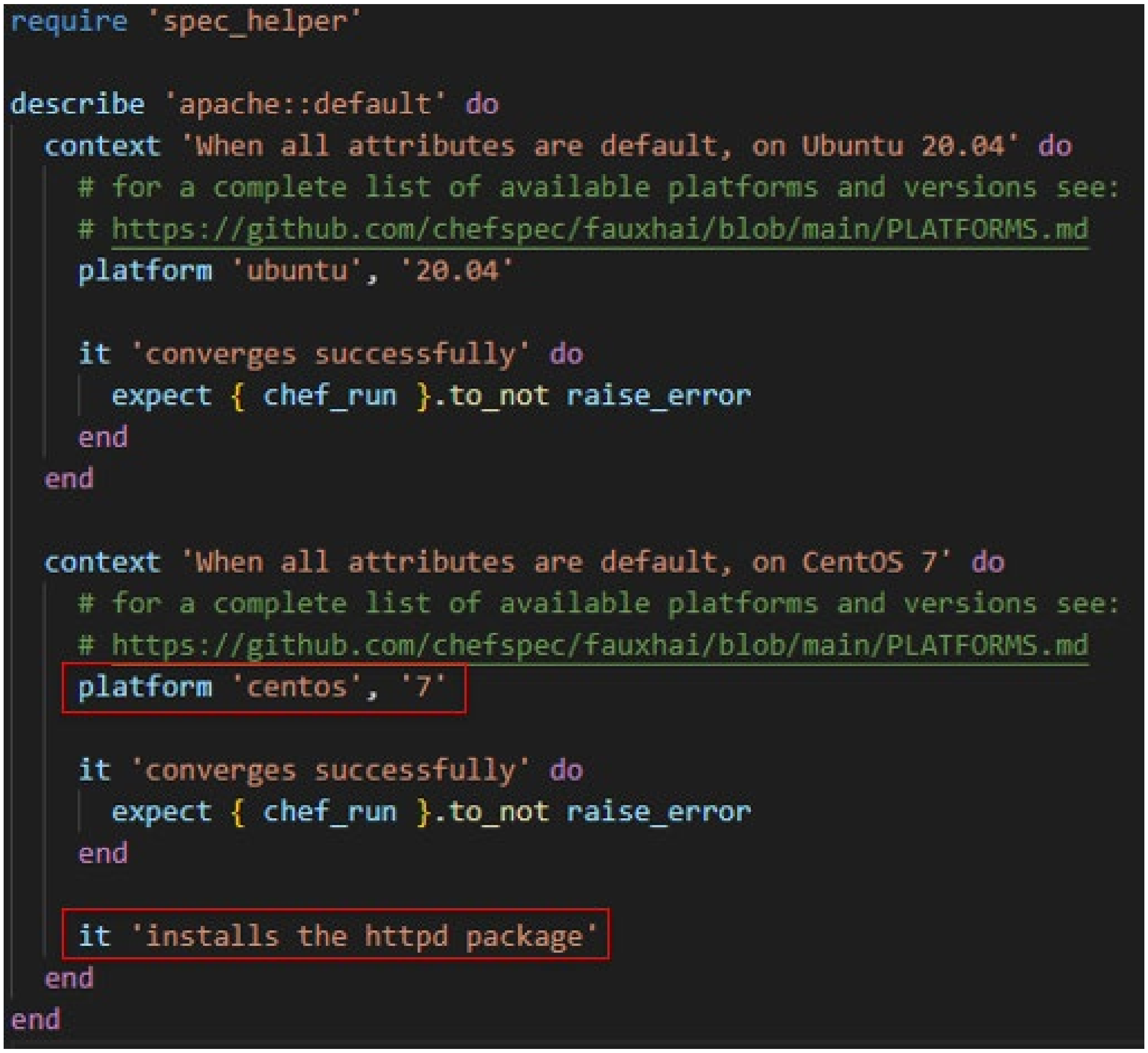
   A. The first test will check if the package is **installed**

   B. The second test will confirm whether the index.html file has been created

   C. The third test will check if the ‘httpd’ service is **enabled** and **started**

5. Update the **spec/unit/recipes/default_spec.rb** file under the CentOS platform as described below:

   Note: we are changing the centos version to 7, to match our Inspec test

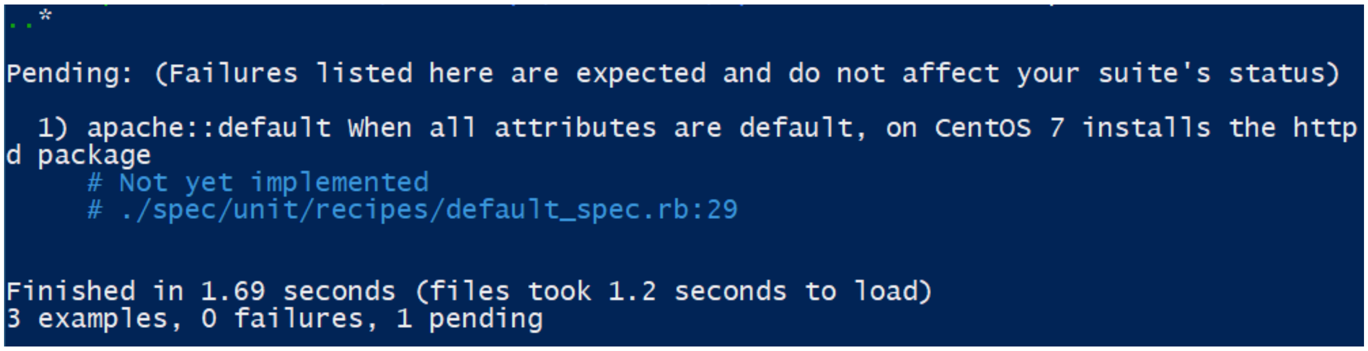


```ruby
require 'spec_helper'

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/main/PLATFORMS.md
    platform 'ubuntu', '20.04'

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

  context 'When all attributes are default, on CentOS 7' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/main/PLATFORMS.md
    platform 'centos', '7'

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

    it 'installs the httpd package'
  end
end
```

6. Save the file and execute the following command:

   $ chef exec rspec

   Expected output:


```
..*

Pending: (Failures listed here are expected and do not affect your suite's status)

  1) apache::default When all attributes are default, on CentOS 7 installs the http
d package
     # Not yet implemented
     # ./spec/unit/recipes/default_spec.rb:29


Finished in 1.69 seconds (files took 1.2 seconds to load)
3 examples, 0 failures, 1 pending
```

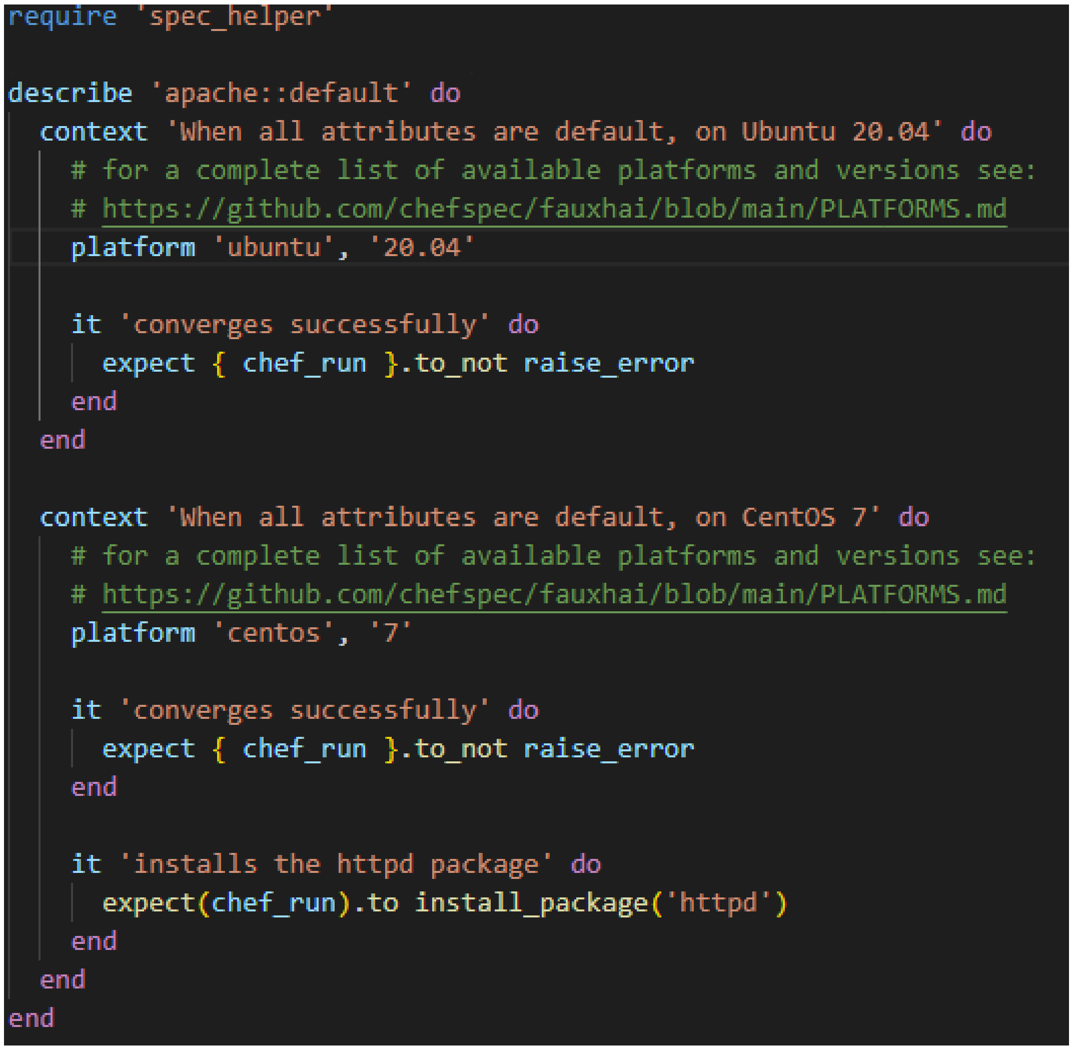
Note: When executing 'rspec' again we should see the new pending example that we defined within the specification file.

RSpec's pending summary is similar to the failure summary. The pending examples are identified and then they are collected together in a list

Also, the summary is displaying that an additional example has been added and it is displayed as pending

7. Update the default_spec.rb file as follows:

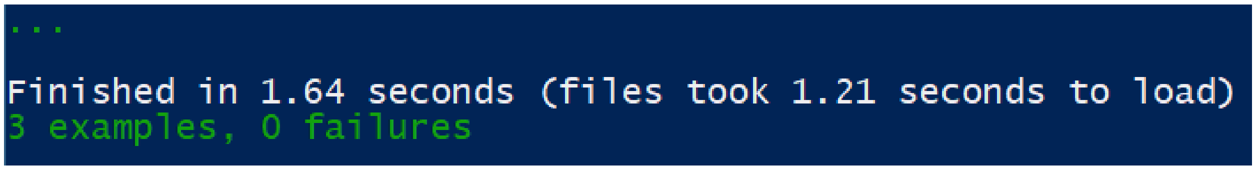
```
require 'spec_helper'

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.m
d
platform 'ubuntu', '20.04'

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

context 'When all attributes are default, on CentOS 7' do
    # for a complete list of available platforms and
versions see:
    #
https://github.com/chefspec/fauxhai/blob/master/PLATFORMS.m
d
  platform 'centos','7'

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

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



```
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/main/PLATFORMS.md
    platform 'ubuntu', '20.04'

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

  context 'When all attributes are default, on CentOS 7' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/main/PLATFORMS.md
    platform 'centos', '7'

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

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

8. Executing 'rspec' one more time and you should see the previously pending example now as a passing example (it is no longer pending)

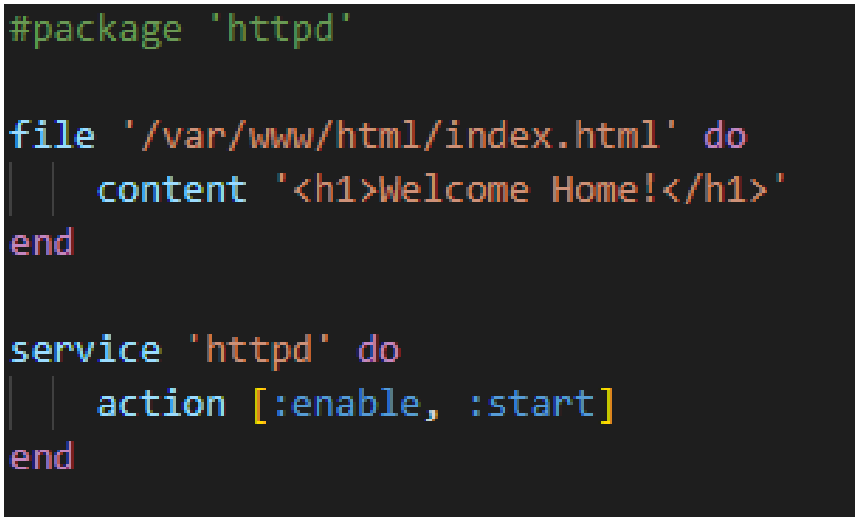
```
$ chef exec rspec
```



```
...

Finished in 1.64 seconds (files took 1.21 seconds to load)
3 examples, 0 failures
```

9. Edit your default recipe to cause a failure:

   a. Comment out
      package 'httpd'
      in recipes/default.rb

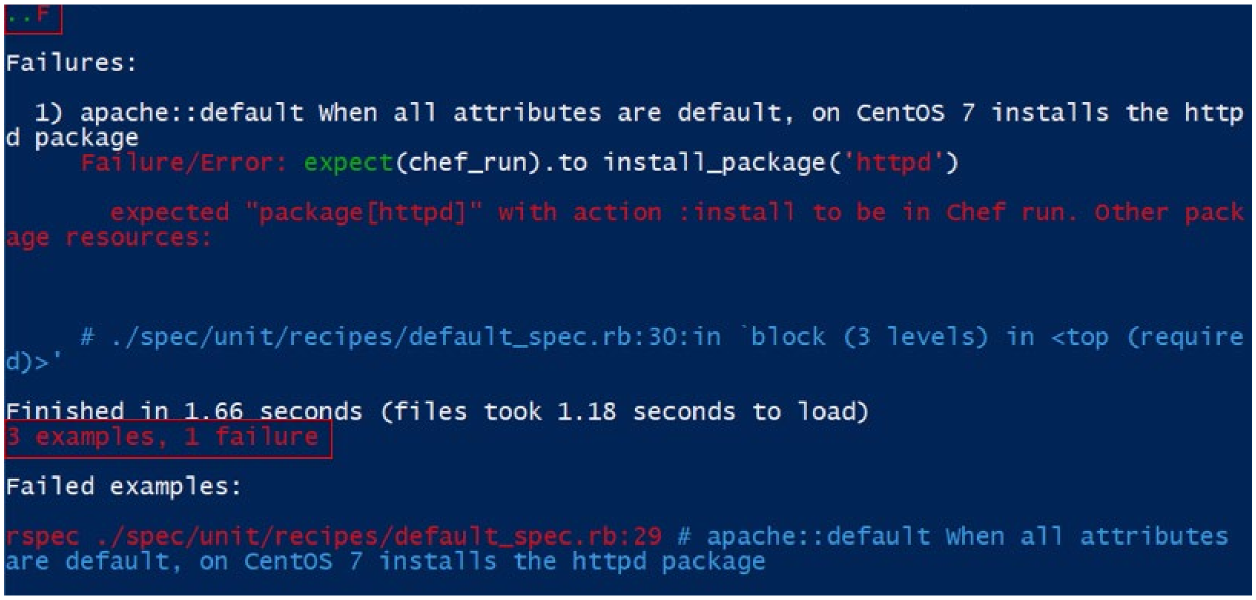


```
#package 'httpd'

file '/var/www/html/index.html' do
    content '<h1>Welcome Home!</h1>'
end

service 'httpd' do
    action [:enable, :start]
end
```

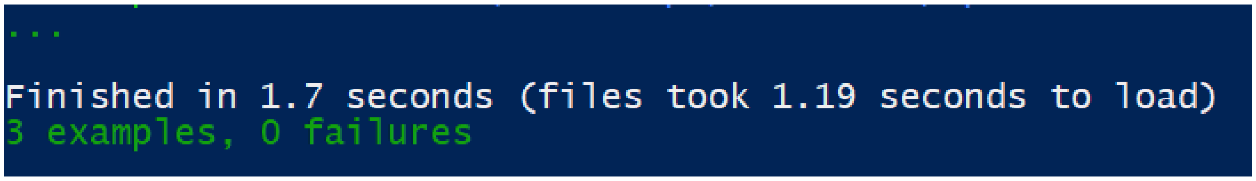
10. Execute rspec:

    $ chef exec rspec

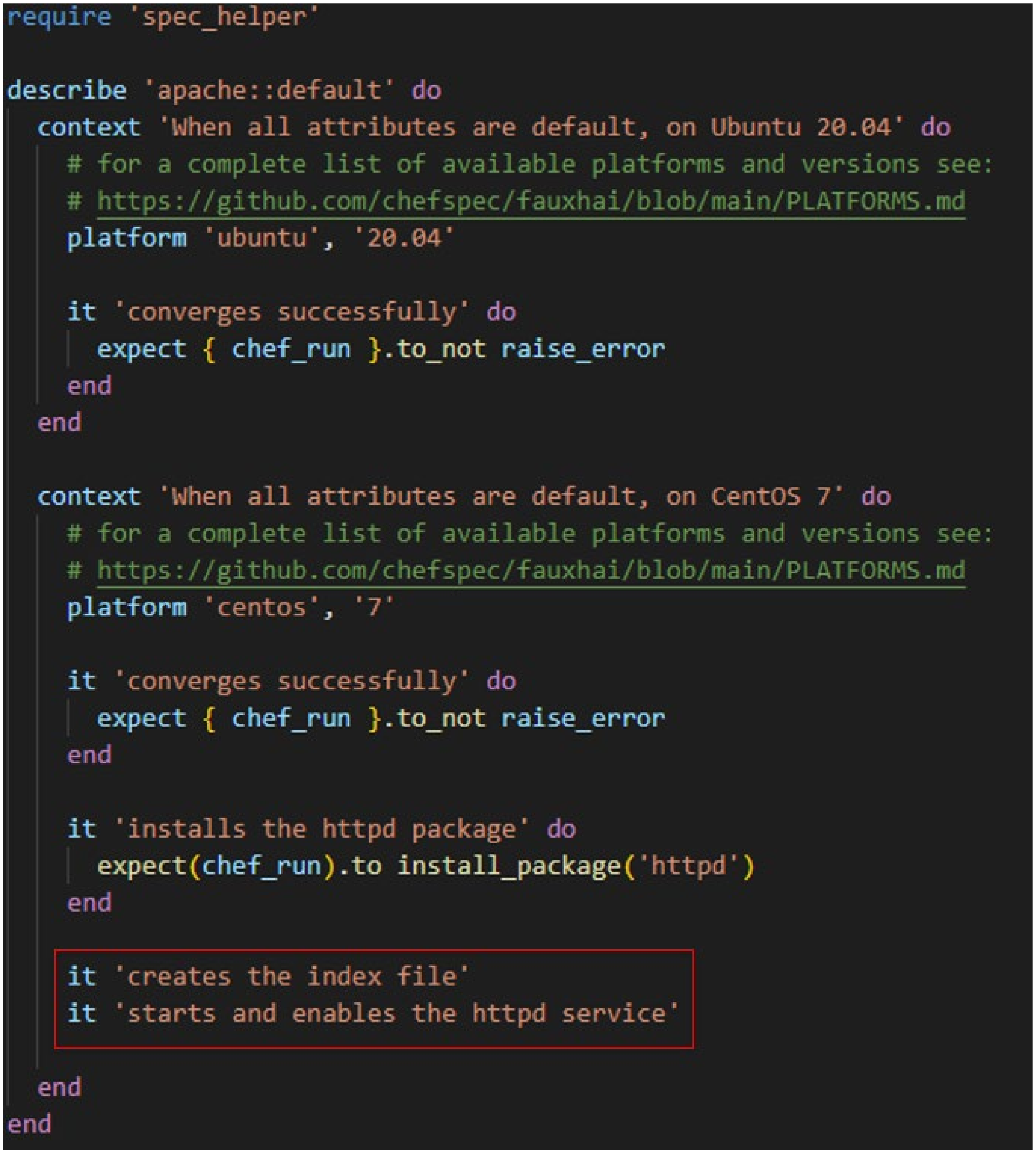
The expected output is two dots in the upper left corner followed by an uppercase F and 3 examples with 1 failure:


```
..F

Failures:

  1) apache::default When all attributes are default, on CentOS 7 installs the http
d package
     Failure/Error: expect(chef_run).to install_package('httpd')

       expected "package[httpd]" with action :install to be in Chef run. Other pack
age resources:


     # ./spec/unit/recipes/default_spec.rb:30:in `block (3 levels) in <top (require
d)>'

Finished in 1.66 seconds (files took 1.18 seconds to load)
3 examples, 1 failure

Failed examples:

rspec ./spec/unit/recipes/default_spec.rb:29 # apache::default When all attributes
are default, on CentOS 7 installs the httpd package
```

11. Uncomment the package resource in your default recipe and test again. You should get:


```
...

Finished in 1.7 seconds (files took 1.19 seconds to load)
3 examples, 0 failures
```

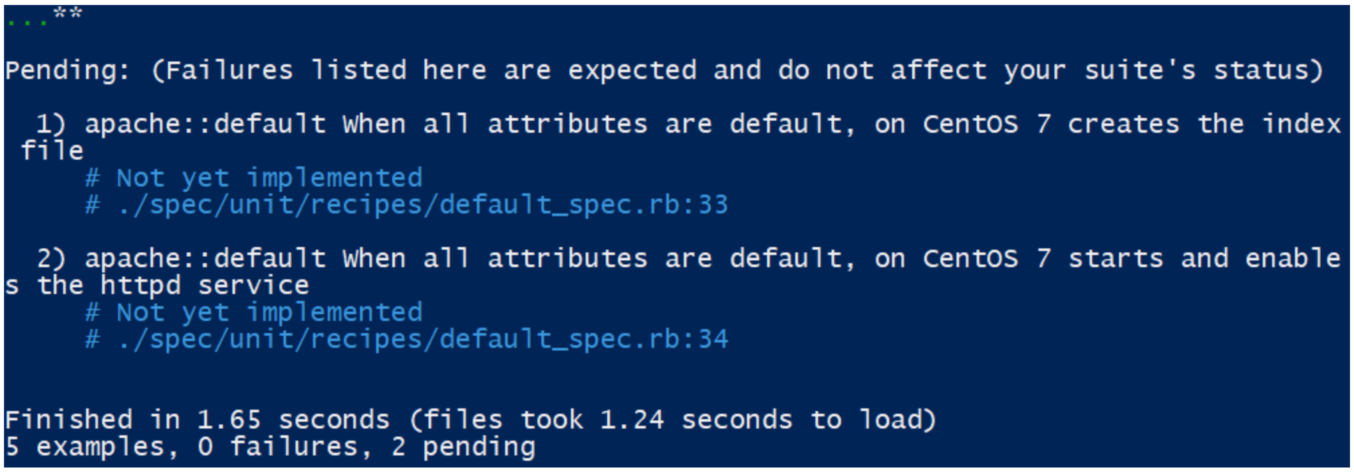
12. Update **default_spec.rb** as follows, adding two new pending tests:



```
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/main/PLATFORMS.md
    platform 'ubuntu', '20.04'

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

  context 'When all attributes are default, on CentOS 7' do
    # for a complete list of available platforms and versions see:
    # https://github.com/chefspec/fauxhai/blob/main/PLATFORMS.md
    platform 'centos', '7'

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

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

    it 'creates the index file'
    it 'starts and enables the httpd service'

  end
end
```

13. Save this file and execute rspec:

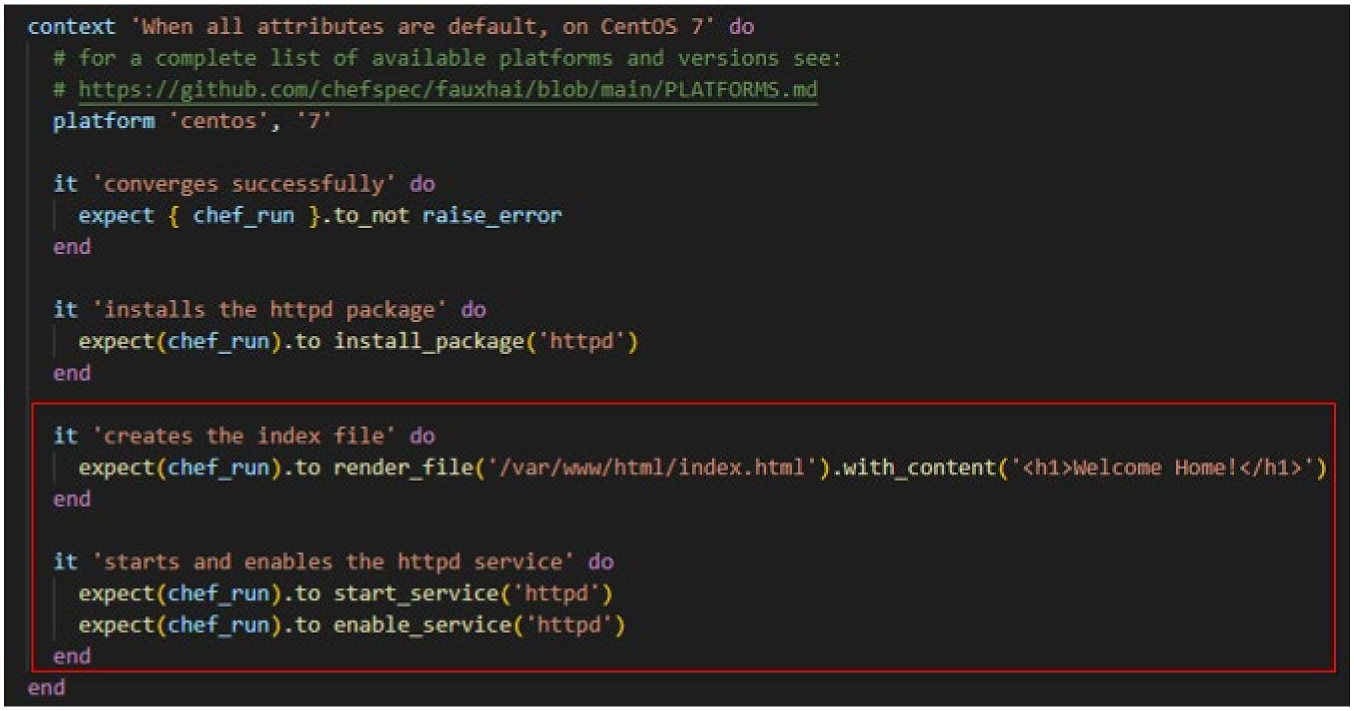
$ chef exec rspec



```
...**

Pending: (Failures listed here are expected and do not affect your suite's status)

  1) apache::default When all attributes are default, on CentOS 7 creates the index
 file
     # Not yet implemented
     # ./spec/unit/recipes/default_spec.rb:33

  2) apache::default When all attributes are default, on CentOS 7 starts and enable
s the httpd service
     # Not yet implemented
     # ./spec/unit/recipes/default_spec.rb:34


Finished in 1.65 seconds (files took 1.24 seconds to load)
5 examples, 0 failures, 2 pending
```

14. Implement the tests by adding the following code:

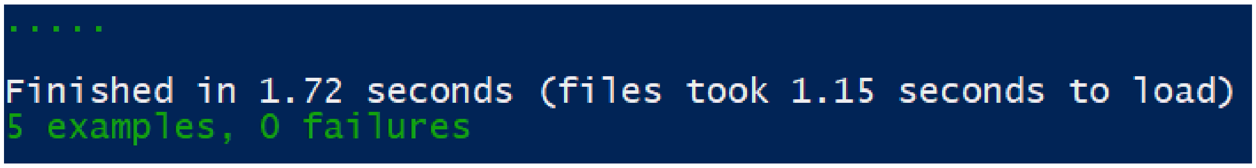
```
it 'creates the index file' do
      expect(chef_run).to
render_file('/var/www/html/index.html').with_content('<h1>Welc
ome Home!</h1>')
    end

    it 'starts and enables the httpd service' do
      expect(chef_run).to start_service('httpd')
      expect(chef_run).to enable_service('httpd')
    end
```



```
context 'When all attributes are default, on CentOS 7' do
  # for a complete list of available platforms and versions see:
  # https://github.com/chefspec/fauxhai/blob/main/PLATFORMS.md
  platform 'centos', '7'

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

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

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

  it 'starts and enables the httpd service' do
    expect(chef_run).to start_service('httpd')
    expect(chef_run).to enable_service('httpd')
  end
end
```

15. Execute your ChefSpec tests again

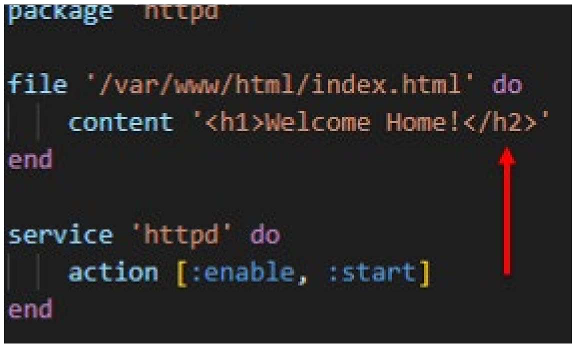
    Expected output:



```
.....

Finished in 1.72 seconds (files took 1.15 seconds to load)
5 examples, 0 failures
```

16. In order to better troubleshoot issues, purposely introduce an error in **recipes/default.rb**:

    Change the </h1> tag to </h2>

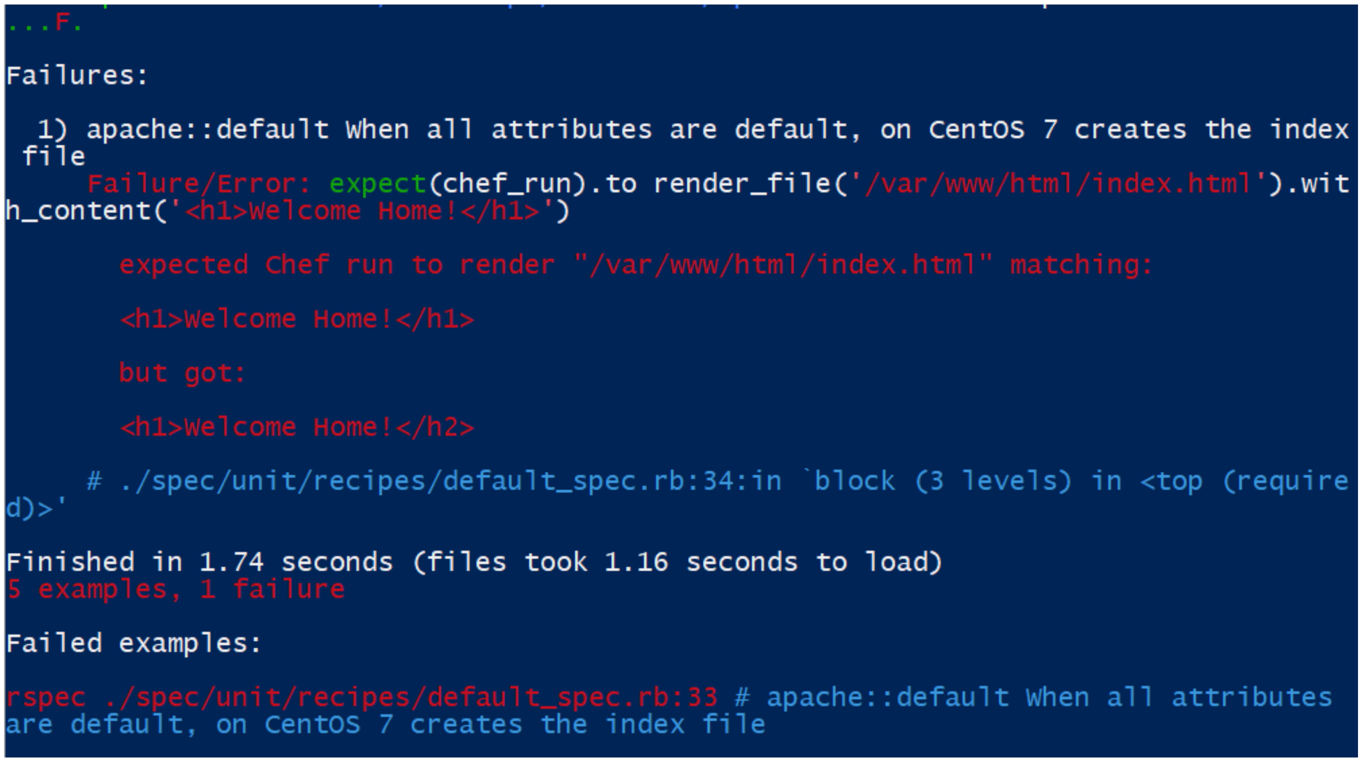


```
package 'httpd'

file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h2>'
end

service 'httpd' do
  action [:enable, :start]
end
```

17. To see the error, execute rspec:

    $ chef exec rspec

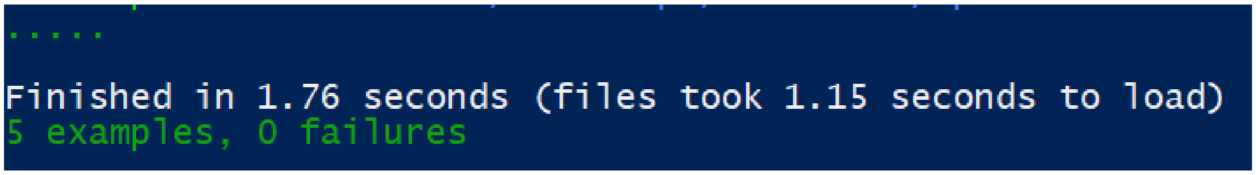
Expected output



```
...F.

Failures:

  1) apache::default When all attributes are default, on CentOS 7 creates the index
 file
     Failure/Error: expect(chef_run).to render_file('/var/www/html/index.html').wit
h_content('<h1>Welcome Home!</h1>')

       expected Chef run to render "/var/www/html/index.html" matching:

       <h1>Welcome Home!</h1>

       but got:

       <h1>Welcome Home!</h2>

     # ./spec/unit/recipes/default_spec.rb:34:in `block (3 levels) in <top (require
d)>'

Finished in 1.74 seconds (files took 1.16 seconds to load)
5 examples, 1 failure

Failed examples:

rspec ./spec/unit/recipes/default_spec.rb:33 # apache::default When all attributes
are default, on CentOS 7 creates the index file
```

18. Fix the error created in **recipes/default.rb**. Execute rspec again to ensure the errors were fixed properly

    Expected output with no errors:



```
.....

Finished in 1.76 seconds (files took 1.15 seconds to load)
5 examples, 0 failures
```

Notify your instructor that you are done with the lab

END OF LAB

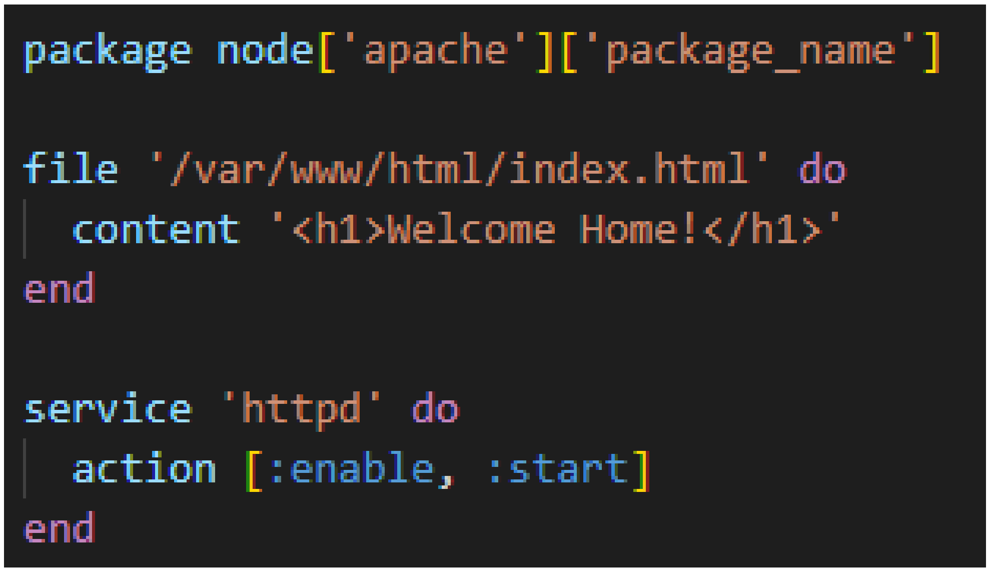
Lab 06, Objectives:

- Refactor resources to use attributes

Use a Cookbook to define node attributes

1. Update **package** resource in the default recipe, then ensure your ChefSpec test fails.

    a. The package name is replaced with a node attribute that has not been created yet.

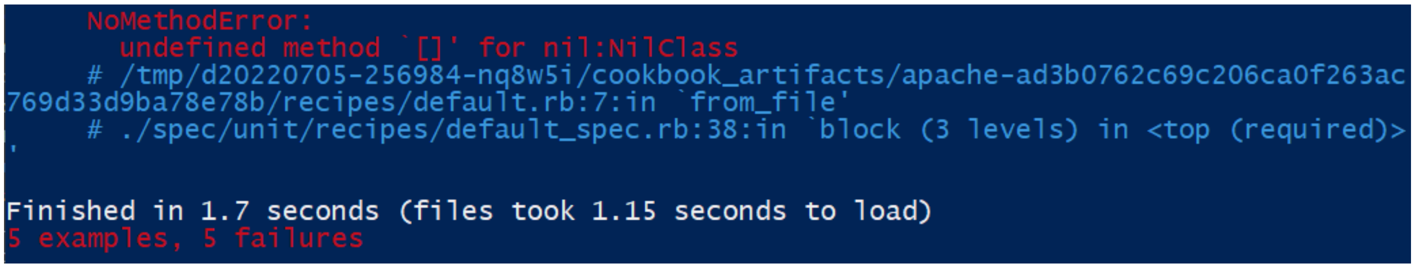


```
package node['apache']['package_name']

file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h1>'
end

service 'httpd' do
  action [:enable, :start]
end
```

    b. Execute your ChefSpec test, expecting it to fail:

    $ chef exec rspec

    c. The expected output is for 5 failed tests (all tests):



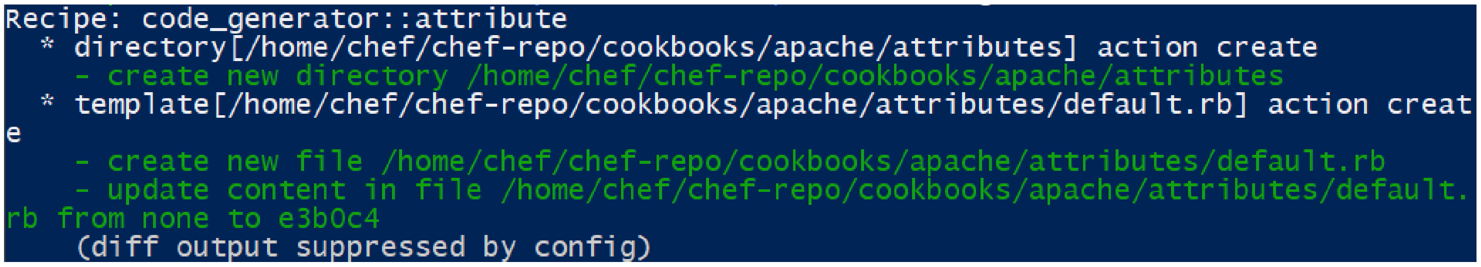
```
     NoMethodError:
       undefined method `[]' for nil:NilClass
     # /tmp/d20220705-256984-nq8w5i/cookbook_artifacts/apache-ad3b0762c69c206ca0f263ac769d33d9ba78e78b/recipes/default.rb:7:in `from_file'
     # ./spec/unit/recipes/default_spec.rb:38:in `block (3 levels) in <top (required)>'

Finished in 1.7 seconds (files took 1.15 seconds to load)
5 examples, 5 failures
```

Generating an attributes file

2. From the root of the apache cookbook directory, execute:

$ chef generate attribute default

Expected output:

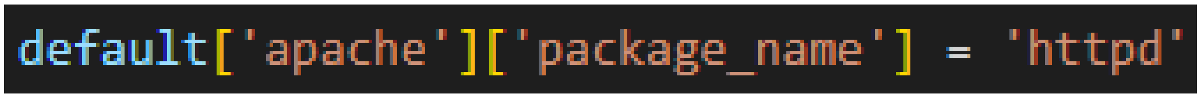


```
Recipe: code_generator::attribute
  * directory[/home/chef/chef-repo/cookbooks/apache/attributes] action create
    - create new directory /home/chef/chef-repo/cookbooks/apache/attributes
  * template[/home/chef/chef-repo/cookbooks/apache/attributes/default.rb] action creat
e
    - create new file /home/chef/chef-repo/cookbooks/apache/attributes/default.rb
    - update content in file /home/chef/chef-repo/cookbooks/apache/attributes/default.
rb from none to e3b0c4
    (diff output suppressed by config)
```

3. Define the node attributes in the newly created **apache/attributes/default.rb** file. Setting a node to 'default' gives this attribute the lowest level precedence

$ vi ~/chef-repo/cookbooks/apache/attributes/default.rb

Add the following node attribute:

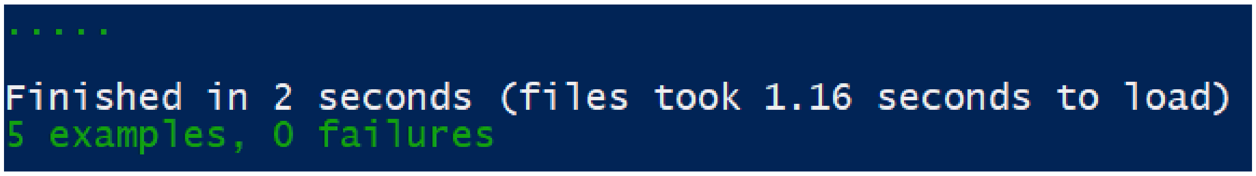


```
default['apache']['package_name'] = 'httpd'
```

Note: Creating this attribute should correct the error made from updating the package resource in the default recipe.

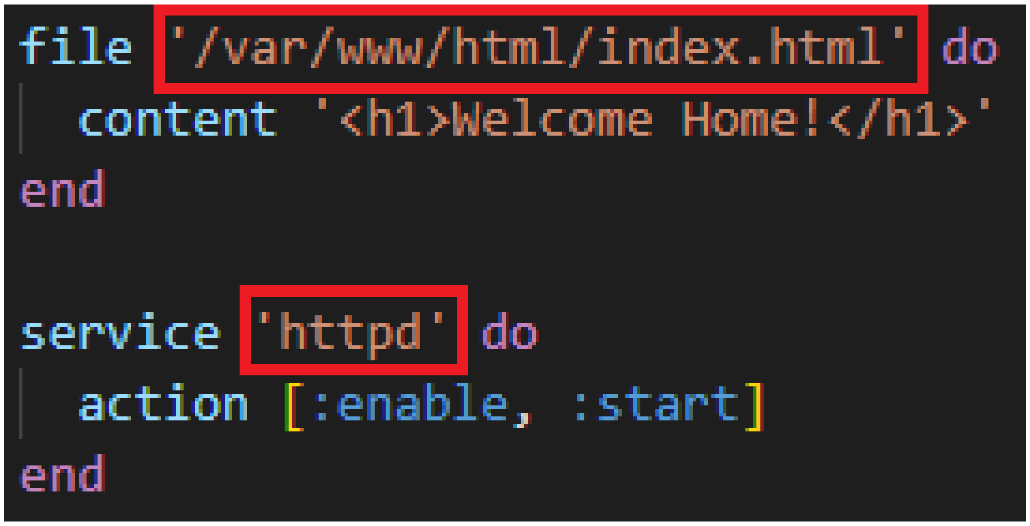
4. Run ChefSpec to see all tests pass:

$ chef exec rspec



```
.....

Finished in 2 seconds (files took 1.16 seconds to load)
5 examples, 0 failures
```

5. Update the **file** and **service** resource names in the default recipe to also use node attributes**. If you need help with this, see the solution at the bottom of the lab**



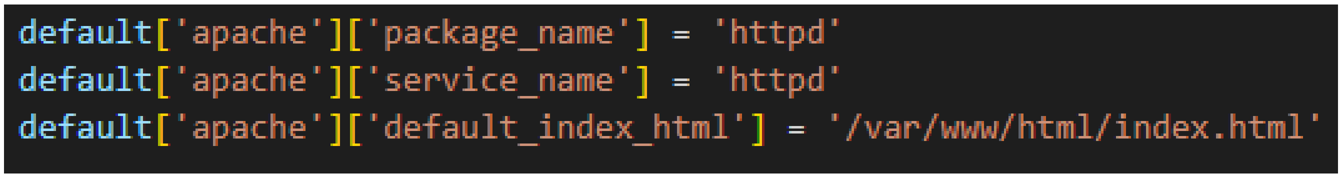
```
file '/var/www/html/index.html' do
  content '<h1>Welcome Home!</h1>'
end

service 'httpd' do
  action [:enable, :start]
end
```

HINT: Use the node attributes you'll define below to update the remaining resources.

6. Run ChefSpec to see the failures

   $ chef exec rspec

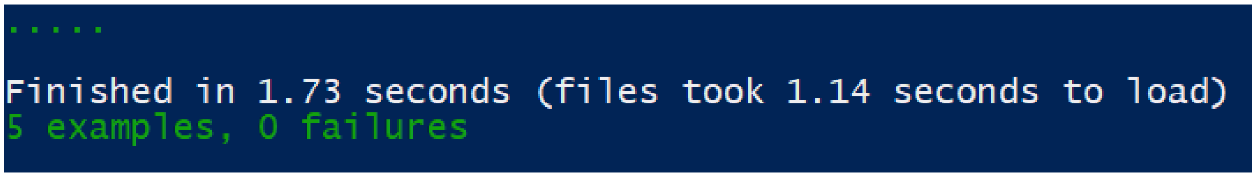
7. To get your tests to pass, update **attributes/default.rb** to reflect the addition of node attributes used in the default recipe:



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

8. Execute ChefSpec, your tests should pass now.

   $ chef exec rspec



```
.....

Finished in 1.73 seconds (files took 1.14 seconds to load)
5 examples, 0 failures
```

Notify your instructor that you are done with the lab

END OF LAB…

...to view the solution, keep on scrolling

. . . .

. . . .

. . . .

. . . .

. . . . . .

. . . . .

. . . .

. . .

. .

.

Keep scrolling...

. . . .

. . . .

. . . .

. . . .

. . . . . .

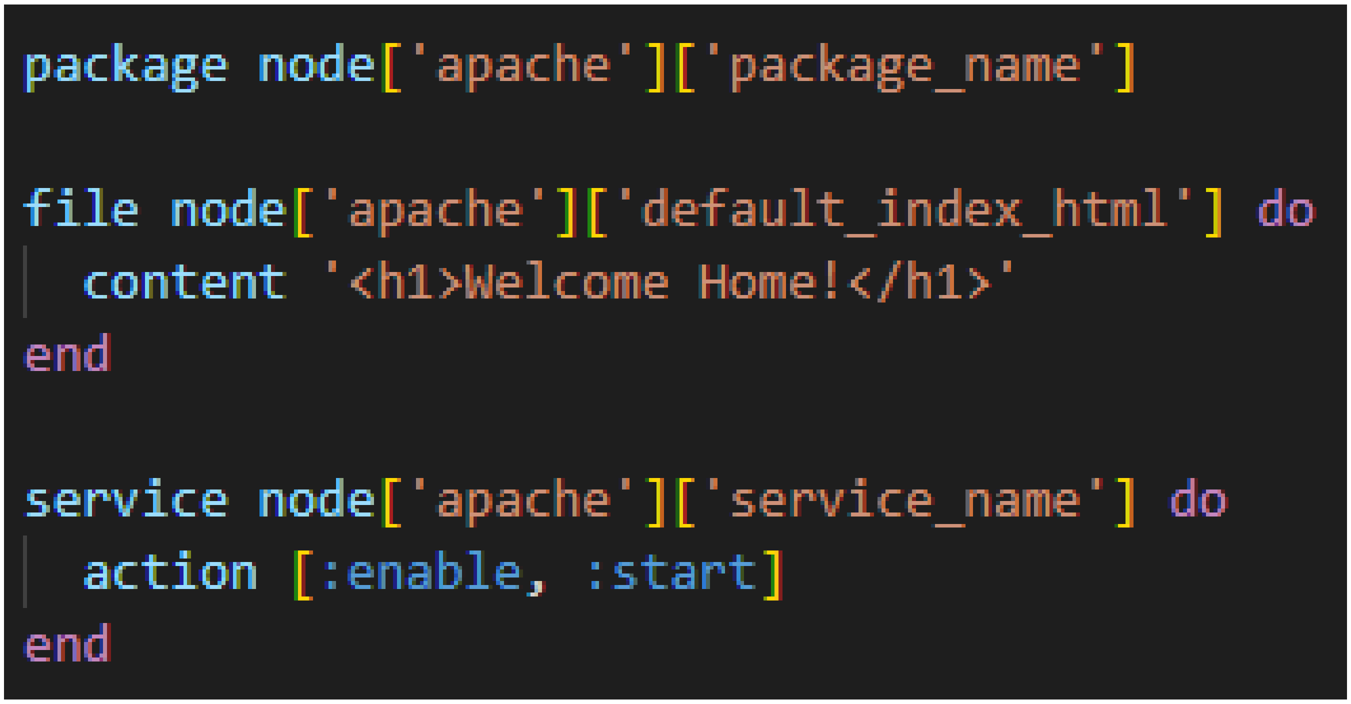
. . . . .

. . . .

. . .

. .

.

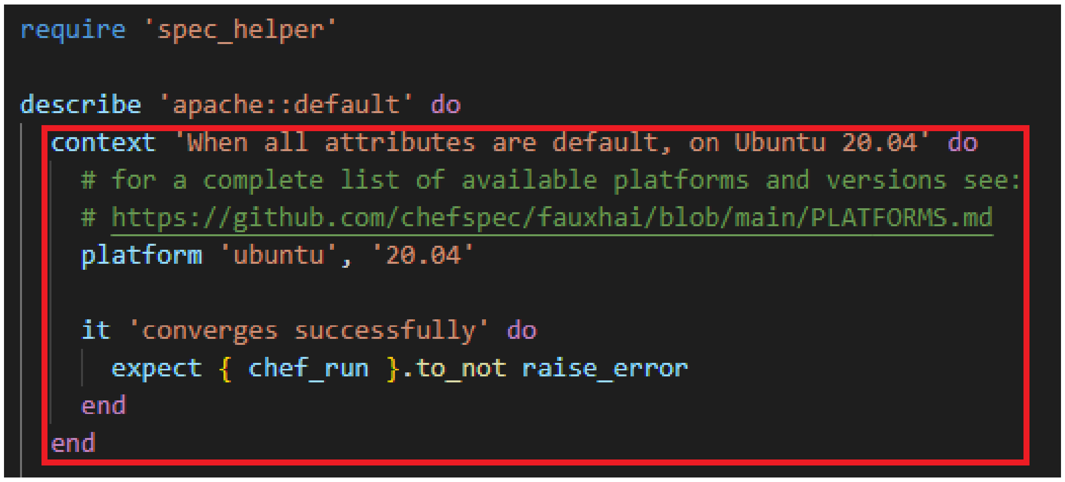


```ruby
package node['apache']['package_name']

file node['apache']['default_index_html'] do
  content '<h1>Welcome Home!</h1>'
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

Lab 07, Objectives:

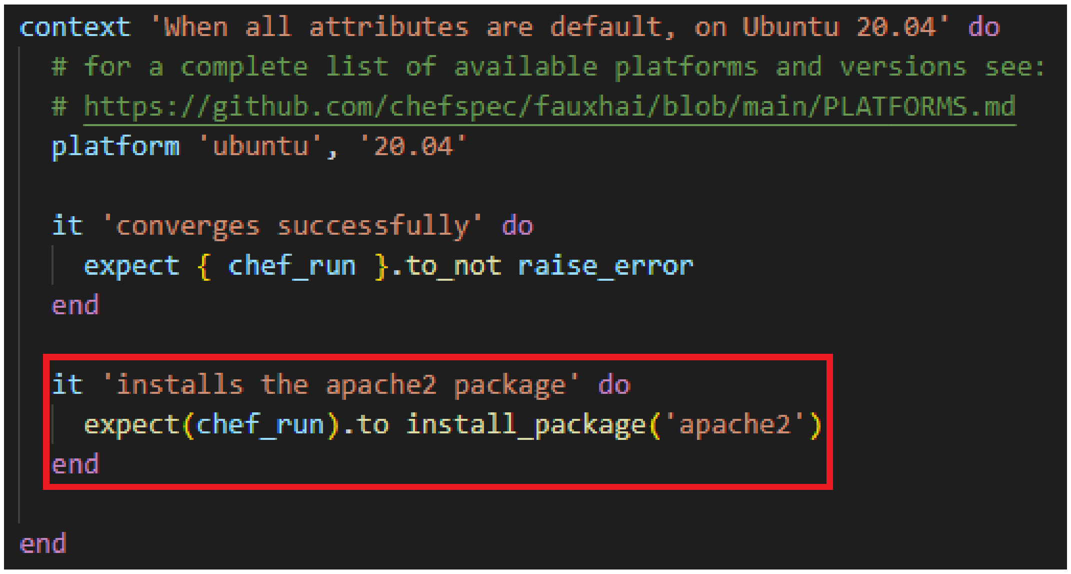
- Define expectations for multiple platforms
- Implement a multi-platform cookbook

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/main/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 the apache service correctly on ubuntu.



```
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/main/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

end
```

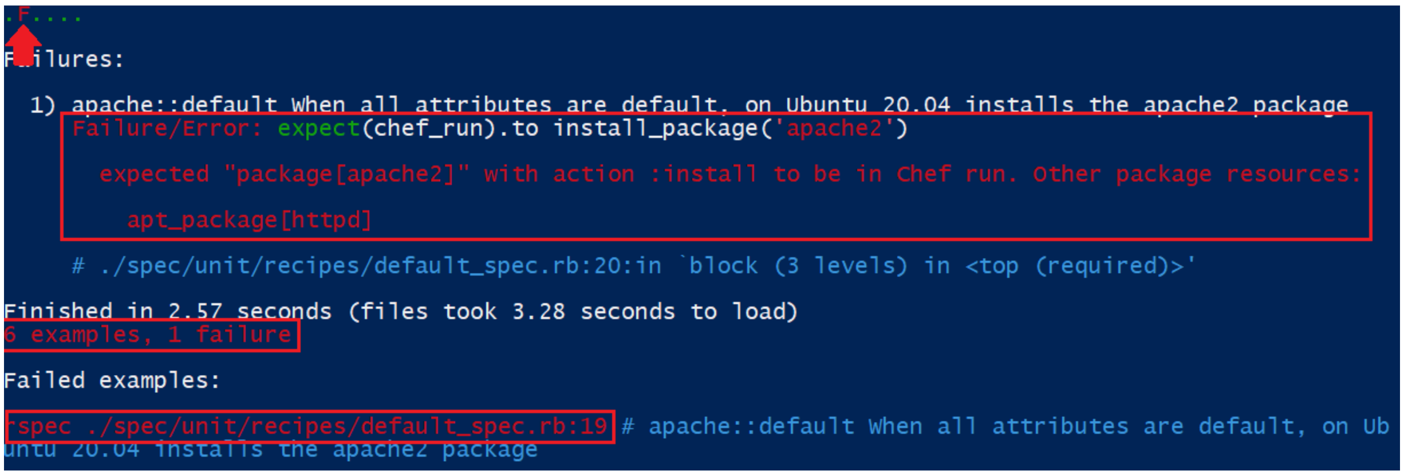
3. After including the new block, execute unit tests with rspec:

   $ chef exec rspec

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’.

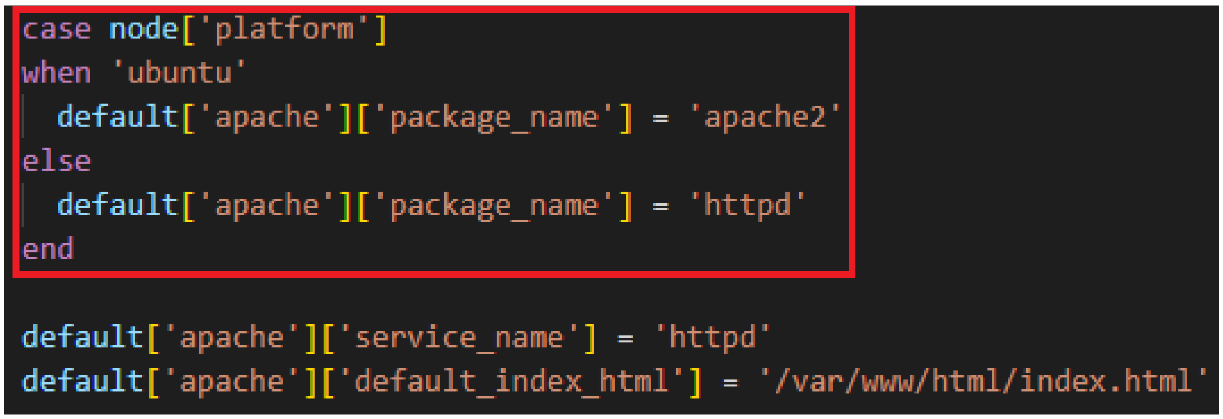
The error states that the apache2 installation on ubuntu failed, which means if this recipe executes on a platform operating on Ubuntu, this recipe will not work as expected.

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



```
.F....

Failures:

  1) apache::default When all attributes are default, on Ubuntu 20.04 installs the apache2 package
     Failure/Error: expect(chef_run).to install_package('apache2')

       expected "package[apache2]" with action :install to be in Chef run. Other package resources:

         apt_package[httpd]

     # ./spec/unit/recipes/default_spec.rb:20:in `block (3 levels) in <top (required)>'

Finished in 2.57 seconds (files took 3.28 seconds to load)
6 examples, 1 failure

Failed examples:

rspec ./spec/unit/recipes/default_spec.rb:19 # apache::default When all attributes are default, on Ubuntu 20.04 installs the apache2 package
```

4. Update the **attributes/default.rb** file to change the package name depending on whether it is running on an **Ubuntu** or **CentOS** machine. The following case statement will accomplish this:



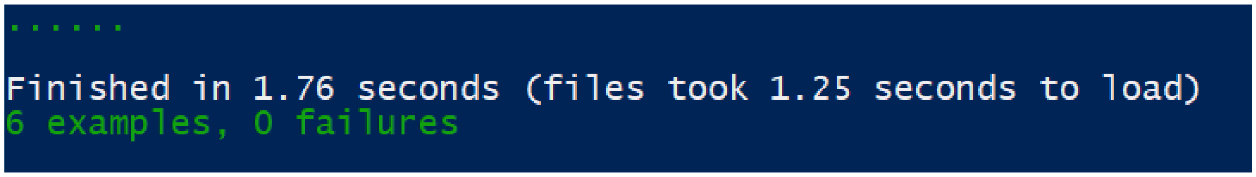
```
case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
else
  default['apache']['package_name'] = 'httpd'
end

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

Node attributes from Ohai will be used to identify the **platform**. The attribute value is **conditionally** based on the **platform**.

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. 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

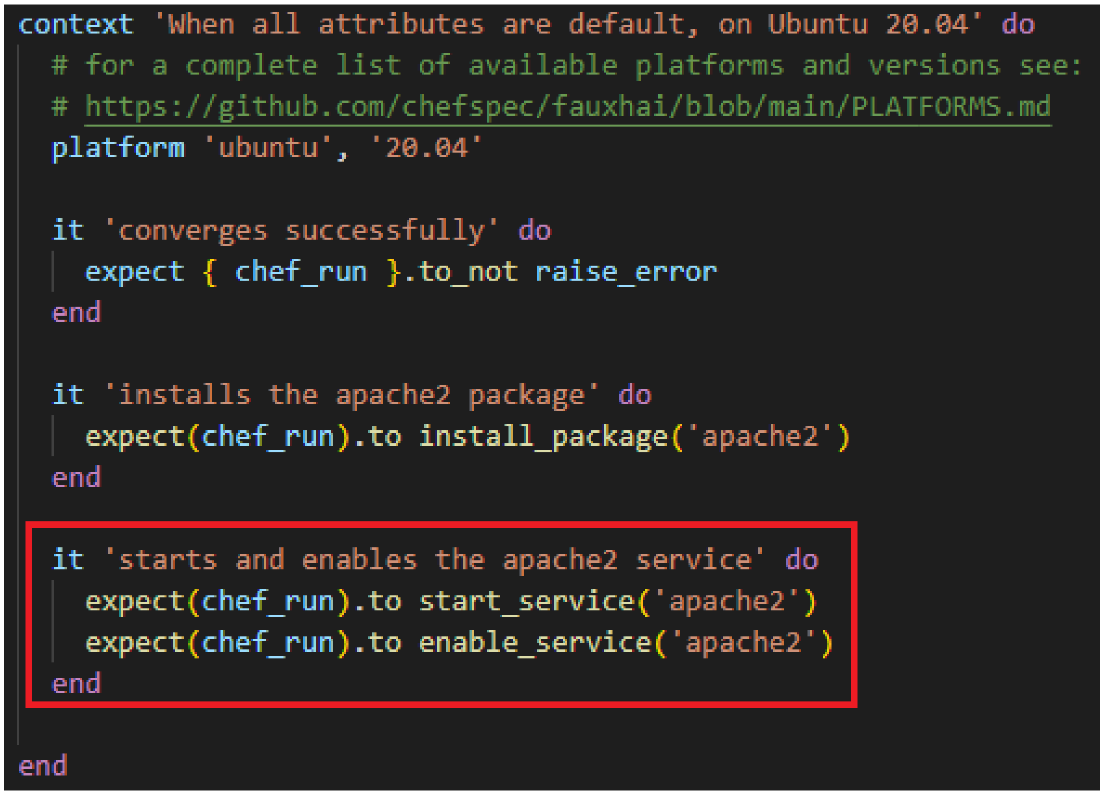
5. Execute the unit test with:

   $ chef exec rspec



```
......

Finished in 1.76 seconds (files took 1.25 seconds to load)
6 examples, 0 failures
```

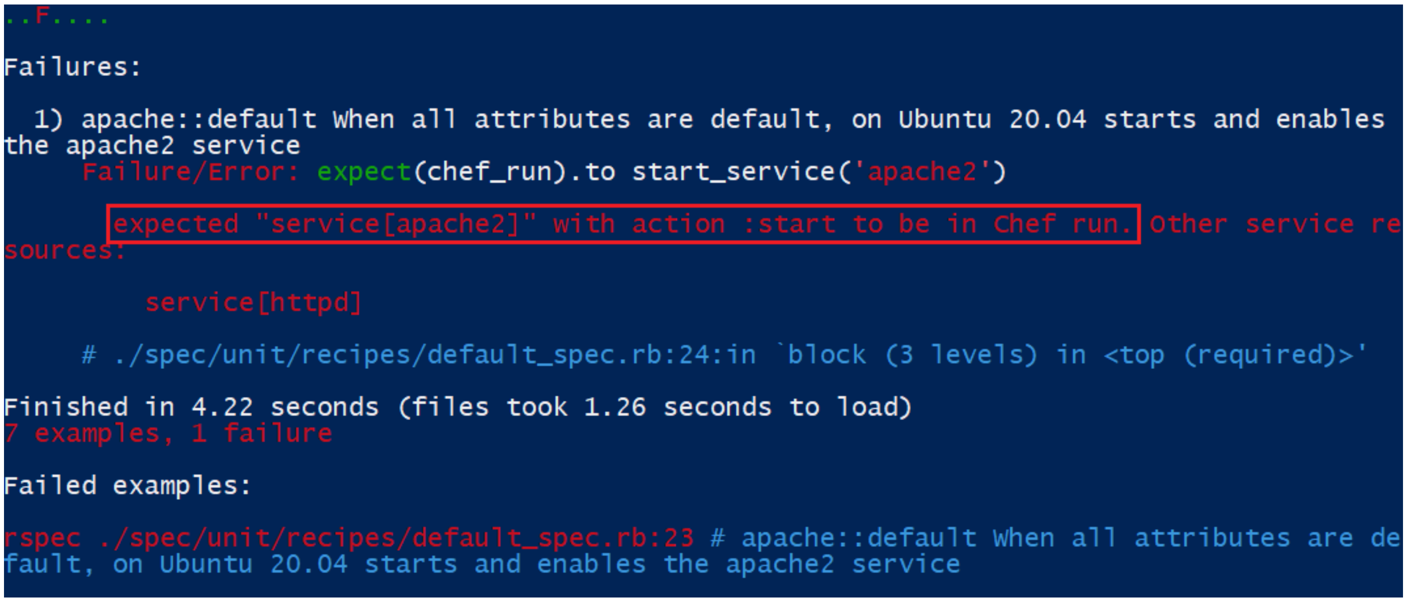
6. The ChefSpec test needs to be updated with a test to ensure that apache2 is started and enabled. The following code will test this condition:



```
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/main/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 and enables the apache2 service' do
    expect(chef_run).to start_service('apache2')
    expect(chef_run).to enable_service('apache2')
  end

end
```

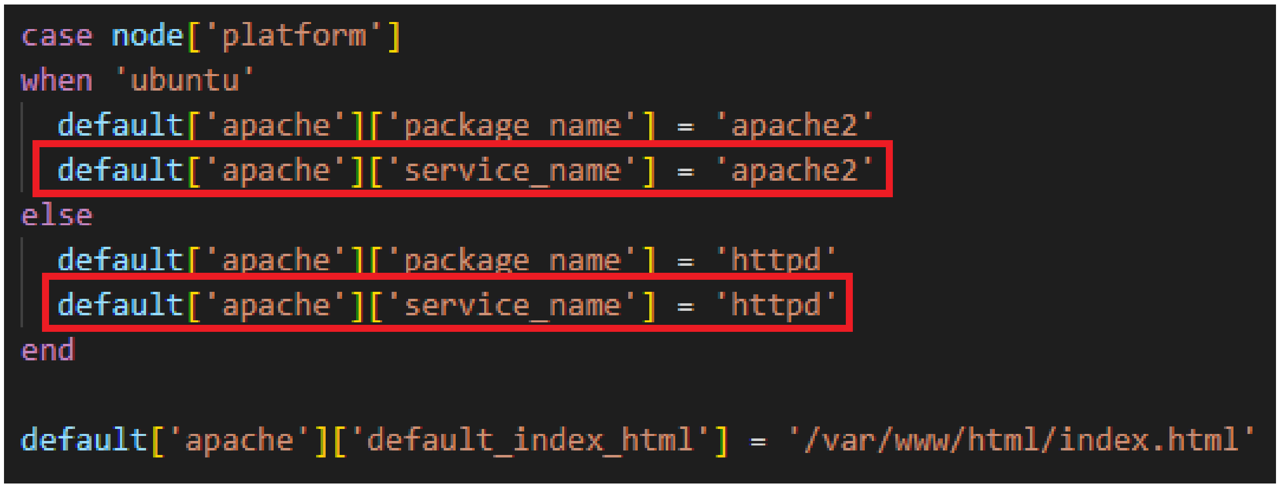
7. Execute the unit test:

$ chef exec rspec



```
..F....

Failures:

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

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

         service[httpd]

     # ./spec/unit/recipes/default_spec.rb:24:in `block (3 levels) in <top (required)>'

Finished in 4.22 seconds (files took 1.26 seconds to load)
7 examples, 1 failure

Failed examples:

rspec ./spec/unit/recipes/default_spec.rb:23 # apache::default When all attributes are de
fault, on Ubuntu 20.04 starts and enables the apache2 service
```

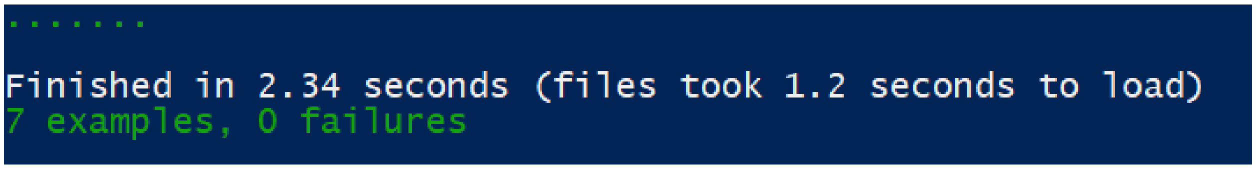
8. Update **attributes/default.rb** to support Ubuntu platform. Start by removing **default [‘apache’] [‘service_name’] = ‘httpd’**. Then define **default [‘apache’] [‘service_name’]** inside of the case statement with the correct value. It should look like this:



```
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'
```

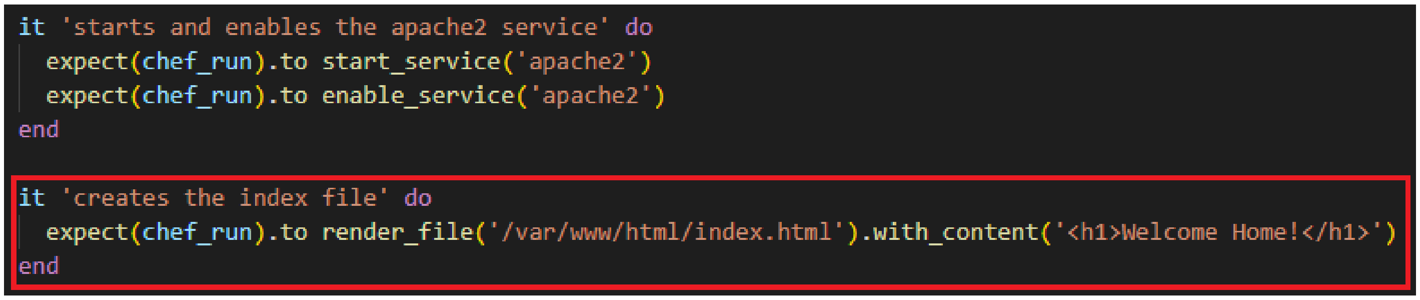
9. Execute the unit test

$ chef exec rspec



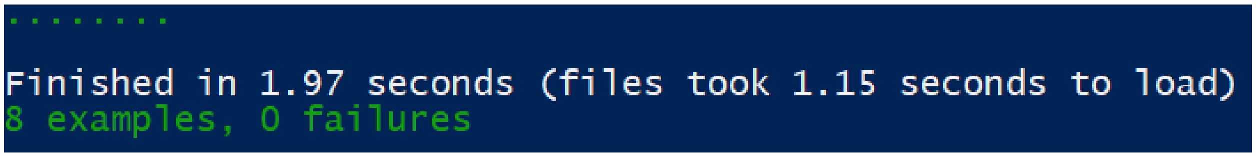
```
.......

Finished in 2.34 seconds (files took 1.2 seconds to load)
7 examples, 0 failures
```

10. Update the unit test to verify the index.html file exists by adding the following code:



```
it 'starts and enables the apache2 service' do
  expect(chef_run).to start_service('apache2')
  expect(chef_run).to enable_service('apache2')
end

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

11. Execute the unit test:

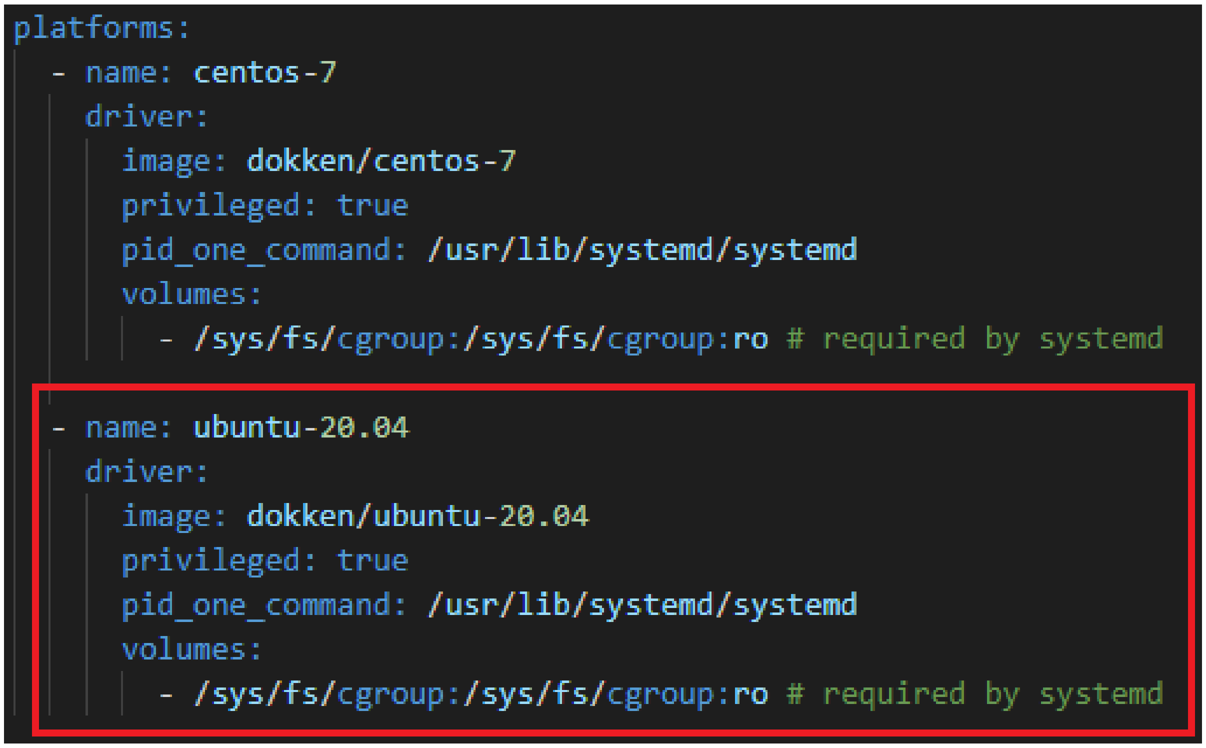


```
........

Finished in 1.97 seconds (files took 1.15 seconds to load)
8 examples, 0 failures
```

*Note: The attributes file **did not** need to be modified because the creation and content of the **index.html** file will be the same **irrespective** of the **platform**.*

12. Update Test Kitchen integration tests to execute on Ubuntu by adding the following to the **kitchen.yml** file:

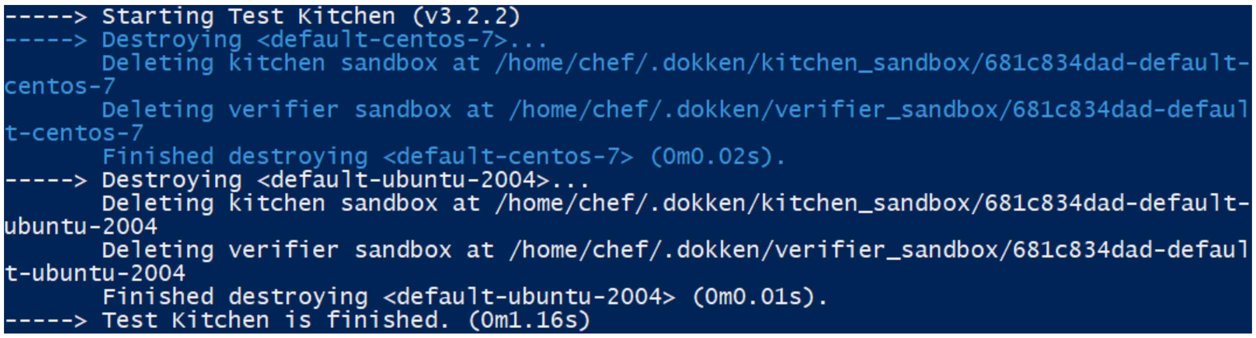
```
- name: ubuntu-20.04
  driver:
    image: dokken/ubuntu-20.04
    privileged: true
    pid_one_command: /usr/lib/systemd/systemd
    volumes:
      - /sys/fs/cgroup:/sys/fs/cgroup:ro # required by
systemd
```



```
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
```

13. Shut down any existing test machines:

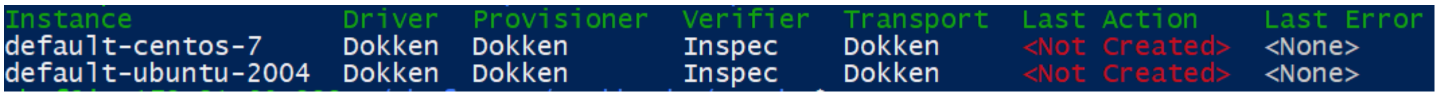
$ kitchen destroy



```
-----> Starting Test Kitchen (v3.2.2)
-----> Destroying <default-centos-7>...
       Deleting kitchen sandbox at /home/chef/.dokken/kitchen_sandbox/681c834dad-default-
centos-7
       Deleting verifier sandbox at /home/chef/.dokken/verifier_sandbox/681c834dad-defaul
t-centos-7
       Finished destroying <default-centos-7> (0m0.02s).
-----> Destroying <default-ubuntu-2004>...
       Deleting kitchen sandbox at /home/chef/.dokken/kitchen_sandbox/681c834dad-default-
ubuntu-2004
       Deleting verifier sandbox at /home/chef/.dokken/verifier_sandbox/681c834dad-defaul
t-ubuntu-2004
       Finished destroying <default-ubuntu-2004> (0m0.01s).
-----> Test Kitchen is finished. (0m1.16s)
```

14. List the existing kitchens to see the new environment. The output confirms that the kitchen was destroyed in the previous step by showing the ‘Last Action’ column as ‘Not Created’ for both kitchen instances:

$ kitchen list

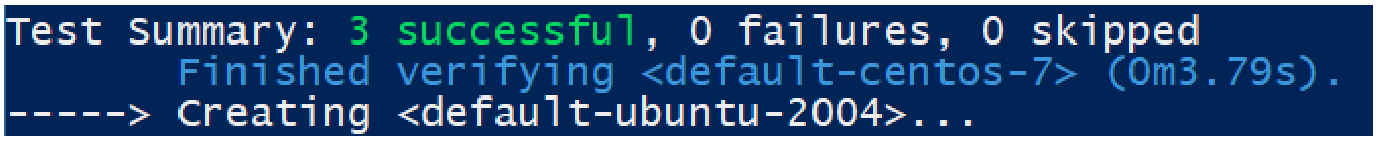


```
Instance             Driver  Provisioner  Verifier  Transport  Last Action    Last Error
default-centos-7     Dokken  Dokken       Inspec    Dokken     <Not Created>  <None>
default-ubuntu-2004  Dokken  Dokken       Inspec    Dokken     <Not Created>  <None>
```

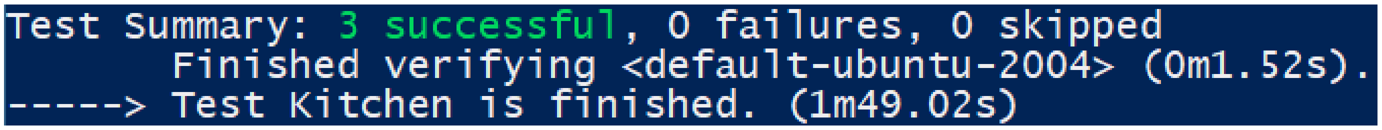
15. Run integration tests by executing the command:

$ kitchen verify

Note: because **kitchen destroy** was ran in a previous, **kitchen verify** will automatically run **kitchen create** and **kitchen converge** after checking if the kitchen exists, then it will run **kitchen verify**



```
Test Summary: 3 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.79s).
-----> Creating <default-ubuntu-2004>...
```



```
Test Summary: 3 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.52s).
-----> Test Kitchen is finished. (1m49.02s)
```

Notify your instructor that you are done with the lab

END OF LAB

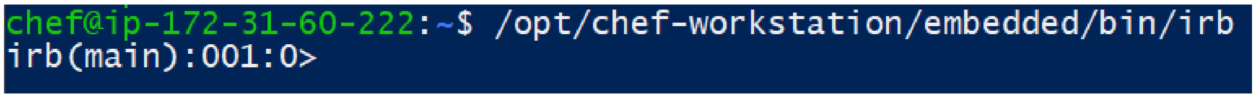
Lab 08, Objectives:

- Work within an interactive ruby shell (IRB)
- Learn the basics of Ruby

This lab is completed inside of an interactive ruby shell. To enter this shell, execute the following command:

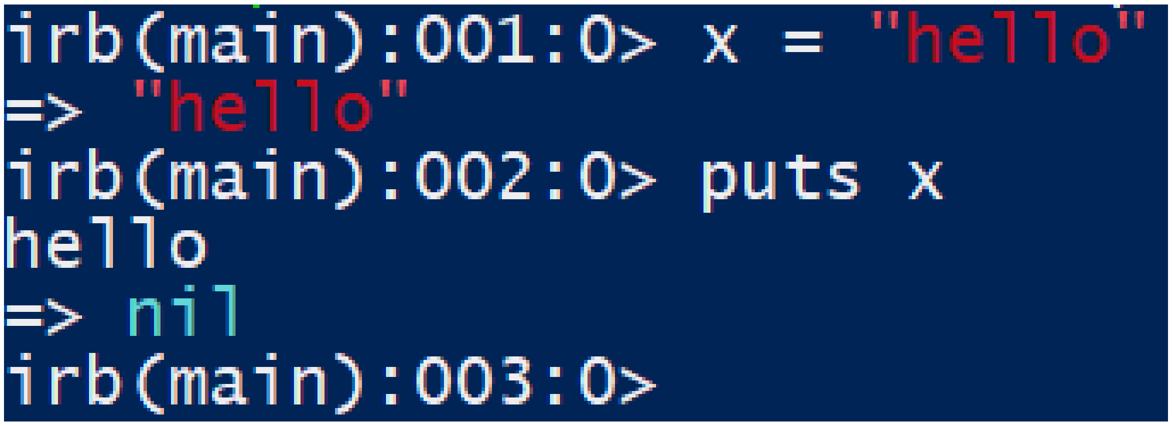
$ /opt/chef-workstation/embedded/bin/irb

Expected output:



```
chef@ip-172-31-60-222:~$ /opt/chef-workstation/embedded/bin/irb
irb(main):001:0>
```

1. Execute the commands shown below:

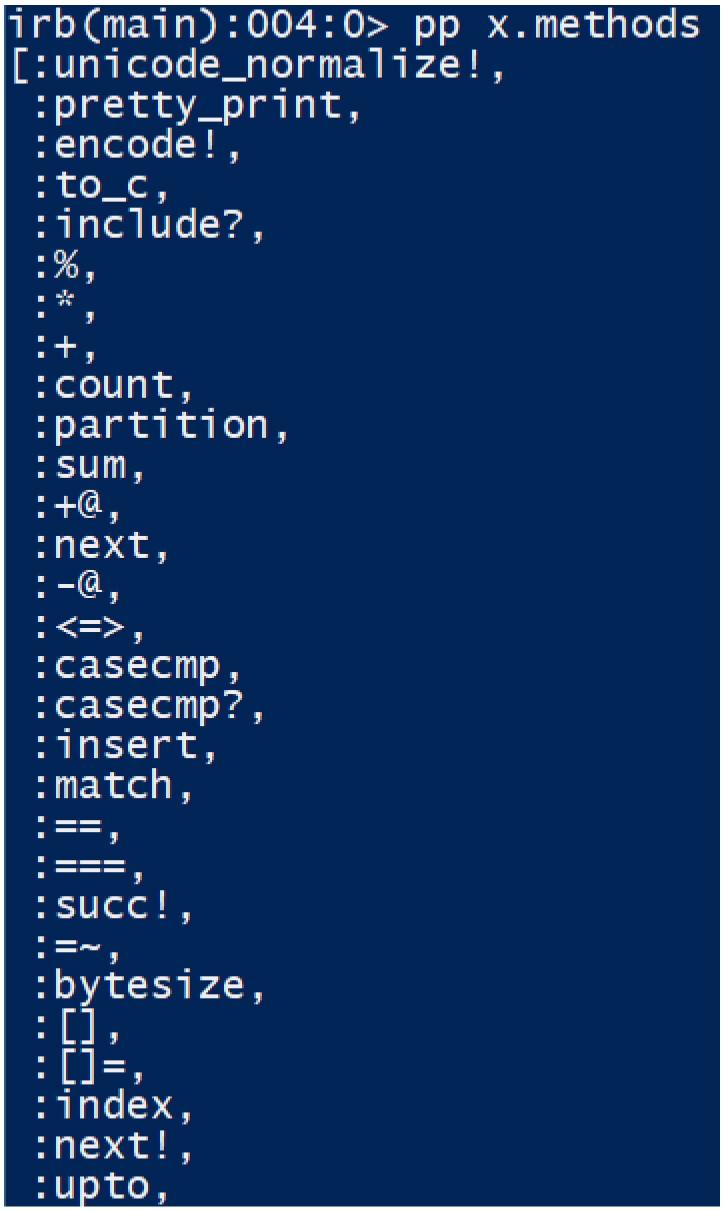


```
irb(main):001:0> x = "hello"
=> "hello"
irb(main):002:0> puts x
hello
=> nil
irb(main):003:0>
```

Variable assignment:

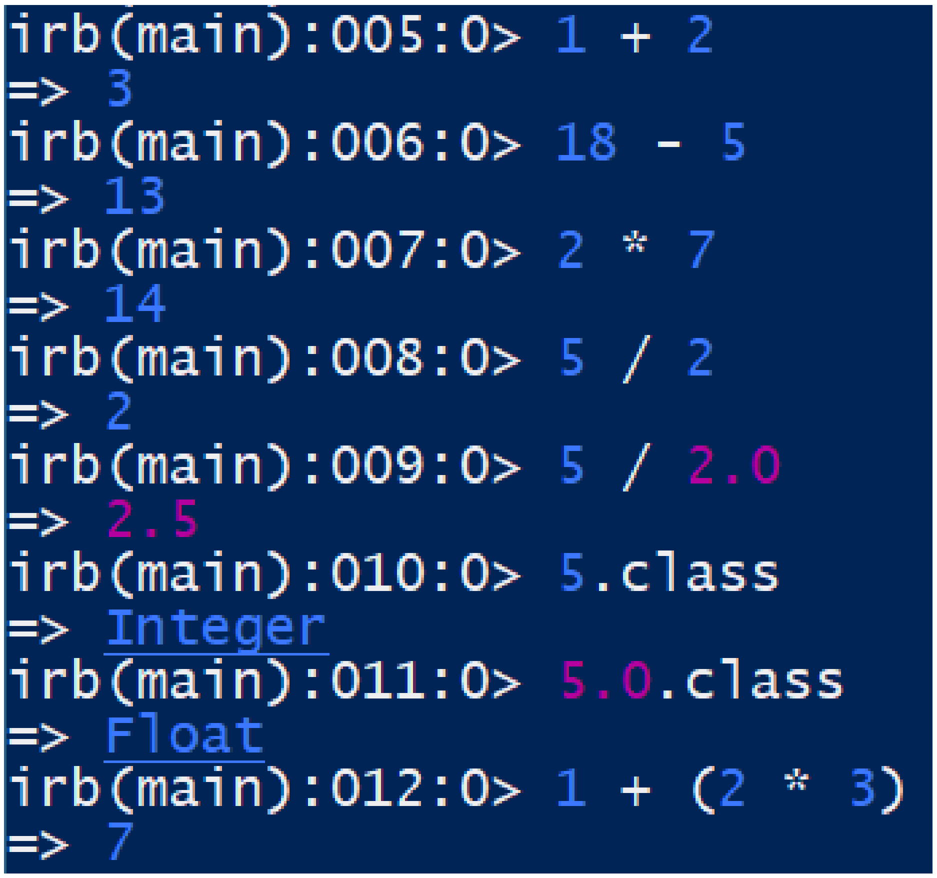
- Variables are assigned with an equal sign ("=”).
- The REPL prints the return value of the last statement, with a => in front.
- "**Puts**" is the ruby equivalent of "print" or "echo" in other languages - it automatically includes a newline.
- "**pp**" (pretty print) also prints objects in a more visually organized way

2. List Methods by executing: **pp x.methods**



```
irb(main):004:0> pp x.methods
[:unicode_normalize!,
 :pretty_print,
 :encode!,
 :to_c,
 :include?,
 :%,
 :*,
 :+,
 :count,
 :partition,
 :sum,
 :+@,
 :next,
 :-@,
 :<=>,
 :casecmp,
 :casecmp?,
 :insert,
 :match,
 :==,
 :===,
 :succ!,
 :=~,
 :bytesize,
 :[],
 :[]=,
 :index,
 :next!,
 :upto,
```

3. Execute the following commands demonstrating arithmetic.

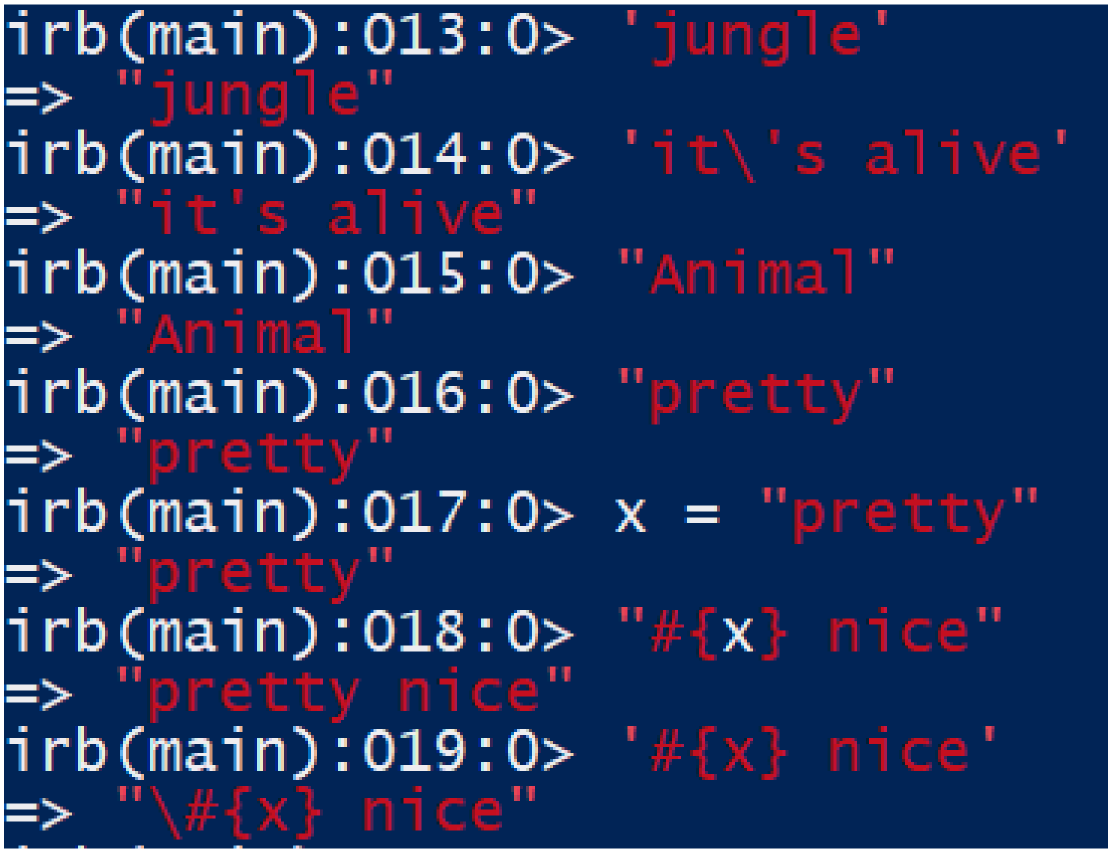


```
irb(main):005:0> 1 + 2
=> 3
irb(main):006:0> 18 - 5
=> 13
irb(main):007:0> 2 * 7
=> 14
irb(main):008:0> 5 / 2
=> 2
irb(main):009:0> 5 / 2.0
=> 2.5
irb(main):010:0> 5.class
=> Integer
irb(main):011:0> 5.0.class
=> Float
irb(main):012:0> 1 + (2 * 3)
=> 7
```

Key Points:

- Numbers are unquoted.
- Addition with quotes are treated as a string and are concatenated:
  "1" + "2" = "12"
- Division using whole numbers means no decimal points:
  5 / 2 returns => 2.
- Division on floating point numbers means decimal points.
- Ruby has dots to call methods and everything in ruby is an object including a number. We can find out what class an object is by calling "**class**".
- Expressions can be grouped with parenthesis like:
  1 + (2 * 3).

4. Execute the following commands demonstrating strings:

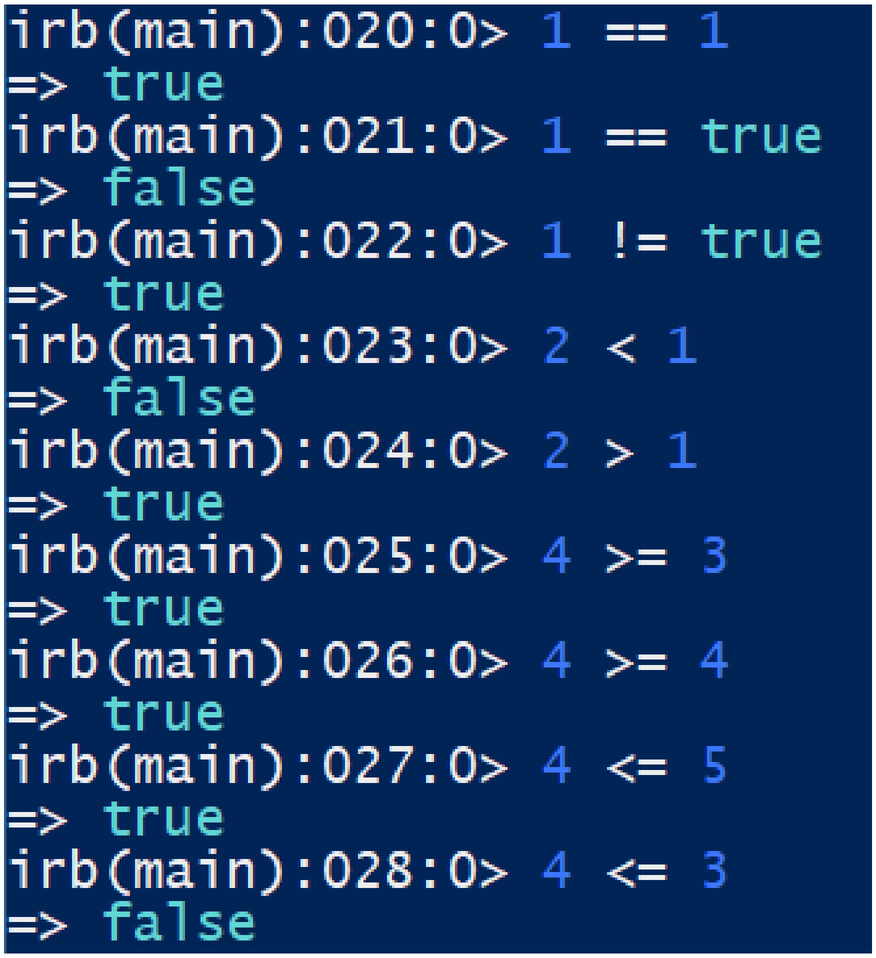


```
irb(main):013:0> 'jungle'
=> "jungle"
irb(main):014:0> 'it\'s alive'
=> "it's alive"
irb(main):015:0> "Animal"
=> "Animal"
irb(main):016:0> "pretty"
=> "pretty"
irb(main):017:0> x = "pretty"
=> "pretty"
irb(main):018:0> "#{x} nice"
=> "pretty nice"
irb(main):019:0> '#{x} nice'
=> "\#{x} nice"
```

Key points:

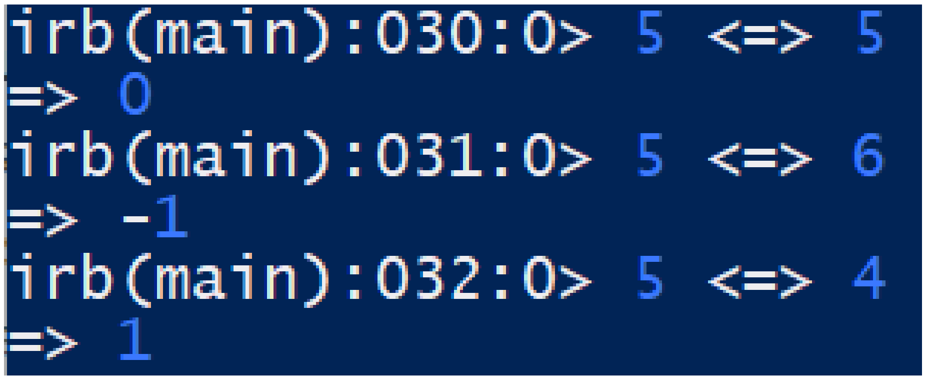
- Strings in ruby can use single quotes or double quotes
- The ruby shell prints results in double quotes to explicitly state that the return value is a string.
- A forward slash escapes from the quote delimiter.
- The ruby shell gives a double quoted version, so there is no escape character.
- Interpolation: the value of an expression can use a variable with #{}
- Strings with single quotes can’t use interpolated variables.

5. Execute the following commands demonstrating equality tests:



```
irb(main):020:0> 1 == 1
=> true
irb(main):021:0> 1 == true
=> false
irb(main):022:0> 1 != true
=> true
irb(main):023:0> 2 < 1
=> false
irb(main):024:0> 2 > 1
=> true
irb(main):025:0> 4 >= 3
=> true
irb(main):026:0> 4 >= 4
=> true
irb(main):027:0> 4 <= 5
=> true
irb(main):028:0> 4 <= 3
=> false
```

6. Ruby has an operator called the combined comparison operator. It is commonly referred to as the spaceship operator.

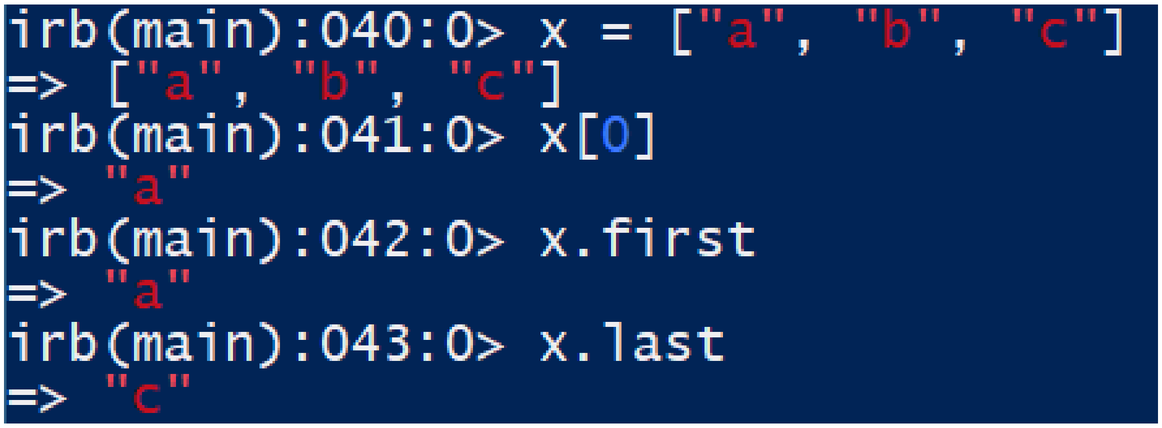


```
irb(main):030:0> 5 <=> 5
=> 0
irb(main):031:0> 5 <=> 6
=> -1
irb(main):032:0> 5 <=> 4
=> 1
```

Key points:

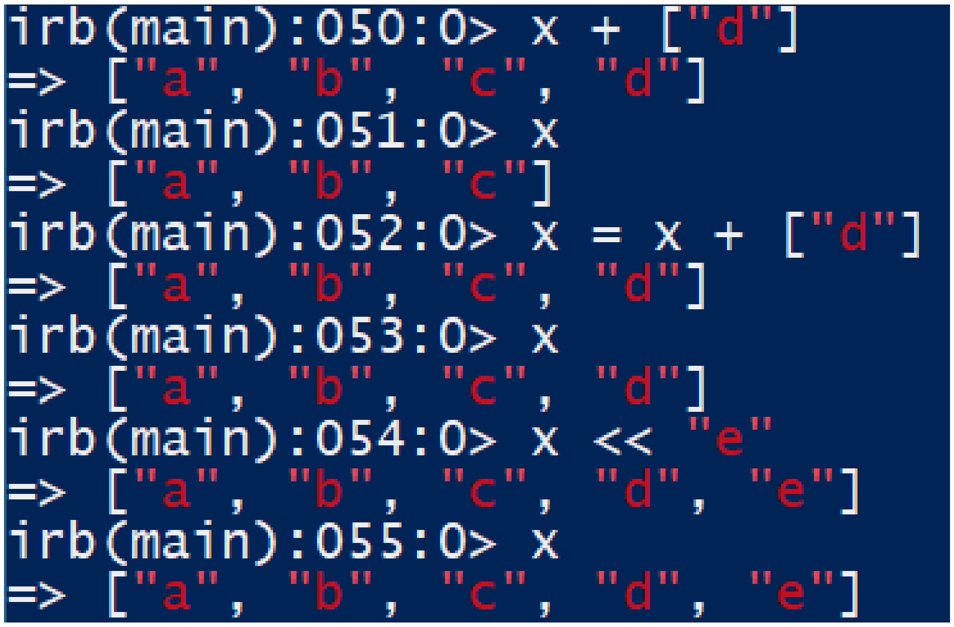
- When greater than, -1 will be returned
- When equal, 0 will be returned
- When less than, 1 will be returned
- This is very useful for sorting

7. Arrays have methods just like everything else, first and last are two methods that apply specifically to arrays.



```
irb(main):040:0> x = ["a", "b", "c"]
=> ["a", "b", "c"]
irb(main):041:0> x[0]
=> "a"
irb(main):042:0> x.first
=> "a"
irb(main):043:0> x.last
=> "c"
```

8. Execute these array manipulation techniques:

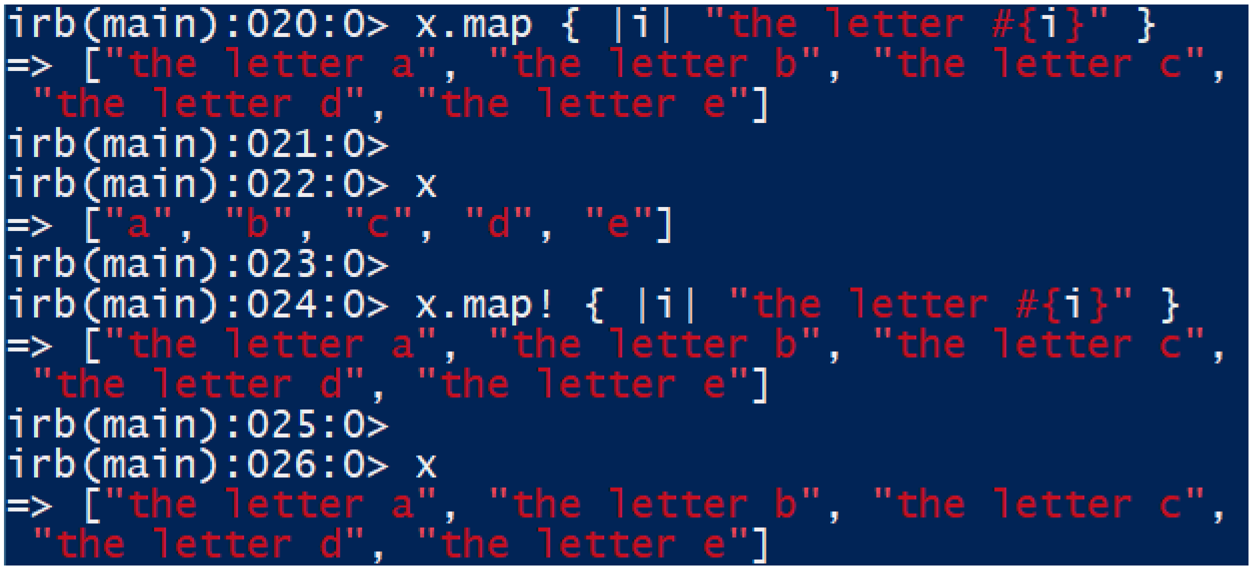


```
irb(main):050:0> x + ["d"]
=> ["a", "b", "c", "d"]
irb(main):051:0> x
=> ["a", "b", "c"]
irb(main):052:0> x = x + ["d"]
=> ["a", "b", "c", "d"]
irb(main):053:0> x
=> ["a", "b", "c", "d"]
irb(main):054:0> x << "e"
=> ["a", "b", "c", "d", "e"]
irb(main):055:0> x
=> ["a", "b", "c", "d", "e"]
```

Key points:

- Adding the element didn't mutate the original array. This is treated as string concatenation.
- In order to destroy the original value, a new value needs to be assigned to 'x'.
- The "<<" operator will add an item to an array!
- there is no "prepend operator", but there are functions that allow you to prepend.

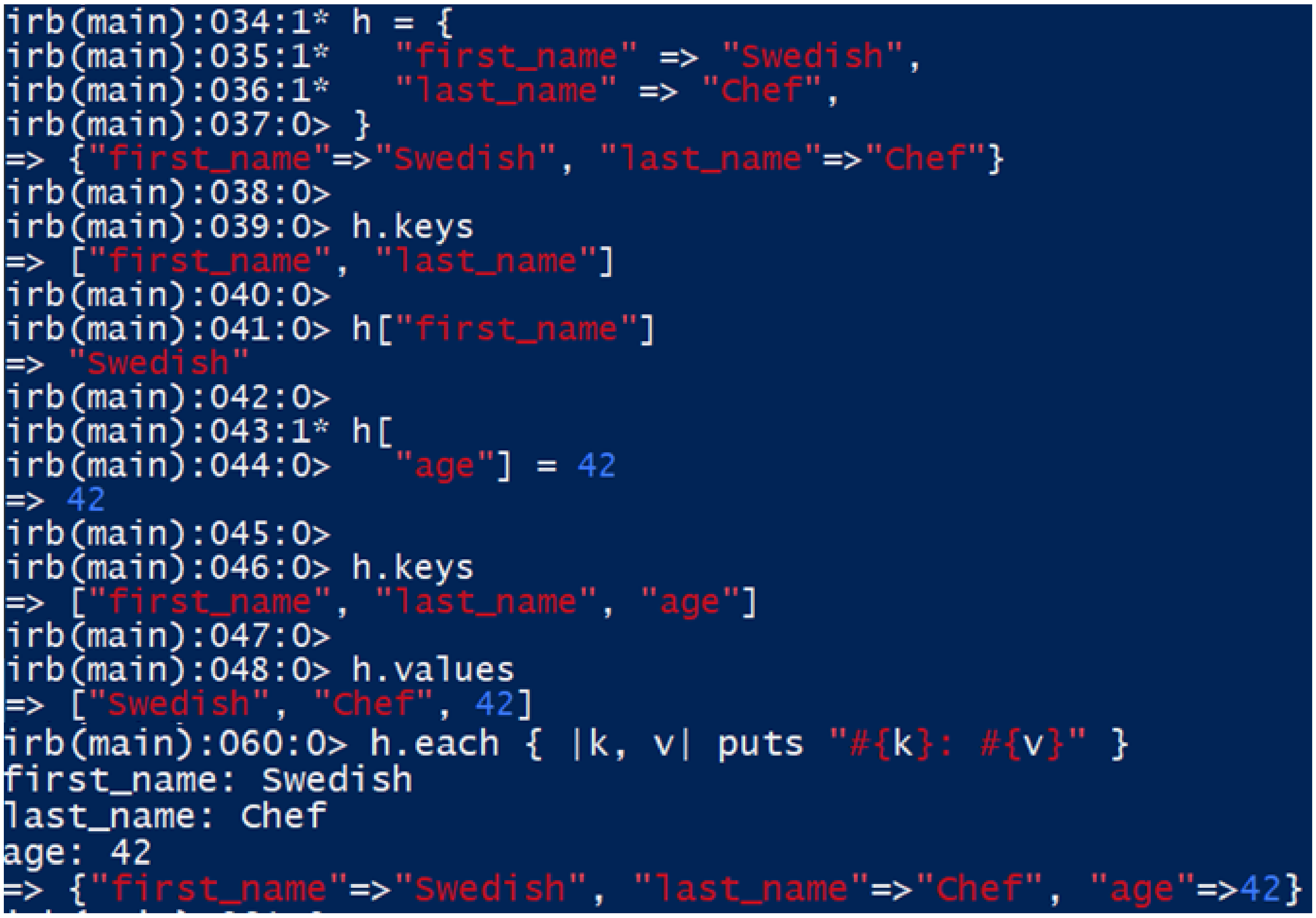
9. The following is one way of iterating through an array:


```
irb(main):020:0> x.map { |i| "the letter #{i}" }
=> ["the letter a", "the letter b", "the letter c",
 "the letter d", "the letter e"]
irb(main):021:0>
irb(main):022:0> x
=> ["a", "b", "c", "d", "e"]
irb(main):023:0>
irb(main):024:0> x.map! { |i| "the letter #{i}" }
=> ["the letter a", "the letter b", "the letter c",
 "the letter d", "the letter e"]
irb(main):025:0>
irb(main):026:0> x
=> ["the letter a", "the letter b", "the letter c",
 "the letter d", "the letter e"]
```

Key points:

- “**map**” does not change the original value of x
- “**map!**” does change the original value of x.

10. Execute the following commands that introduce hashes:

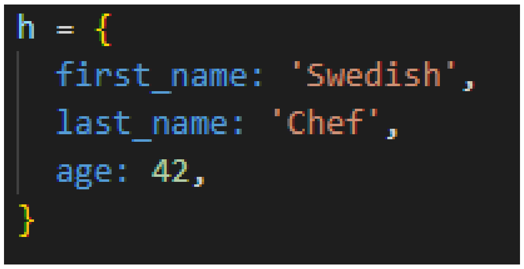


```
irb(main):034:1* h = {
irb(main):035:1*   "first_name" => "Swedish",
irb(main):036:1*   "last_name" => "Chef",
irb(main):037:0> }
=> {"first_name"=>"Swedish", "last_name"=>"Chef"}
irb(main):038:0>
irb(main):039:0> h.keys
=> ["first_name", "last_name"]
irb(main):040:0>
irb(main):041:0> h["first_name"]
=> "Swedish"
irb(main):042:0>
irb(main):043:1* h[
irb(main):044:0>   "age"] = 42
=> 42
irb(main):045:0>
irb(main):046:0> h.keys
=> ["first_name", "last_name", "age"]
irb(main):047:0>
irb(main):048:0> h.values
=> ["Swedish", "Chef", 42]
irb(main):060:0> h.each { |k, v| puts "#{k}: #{v}" }
first_name: Swedish
last_name: Chef
age: 42
=> {"first_name"=>"Swedish", "last_name"=>"Chef", "age"=>42}
```

Key points:

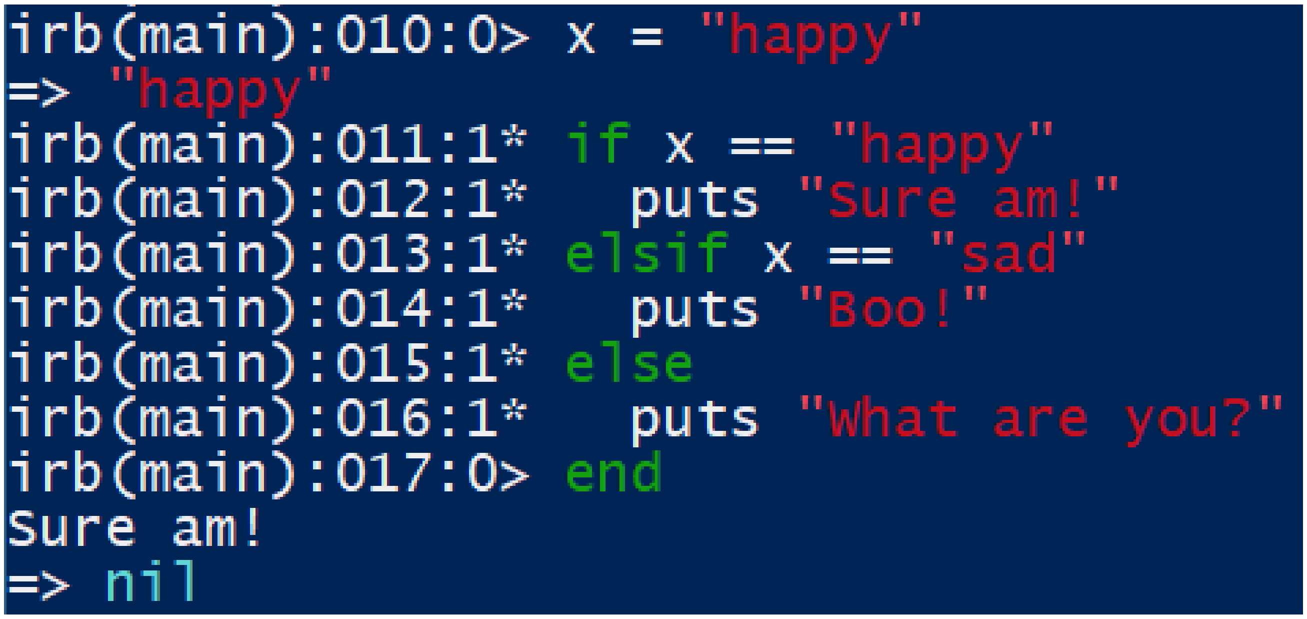
- A hash is a key value pair like a dict in python.
- Items can be added to the hash by putting the key in “square brackets” “[]”, and using assignment on that key name.
- All of the keys in the hash are returned with the **keys** method
- All of the key values in the hash are returned with the **values** method
- A hash can be iterated over with the **each** method

- The follow is also valid syntax for creating a ruby hash:



```
h = {
  first_name: 'Swedish',
  last_name: 'Chef',
  age: 42,
}
```

11. Execute the following commands that introduce the if/elsif/else statement:

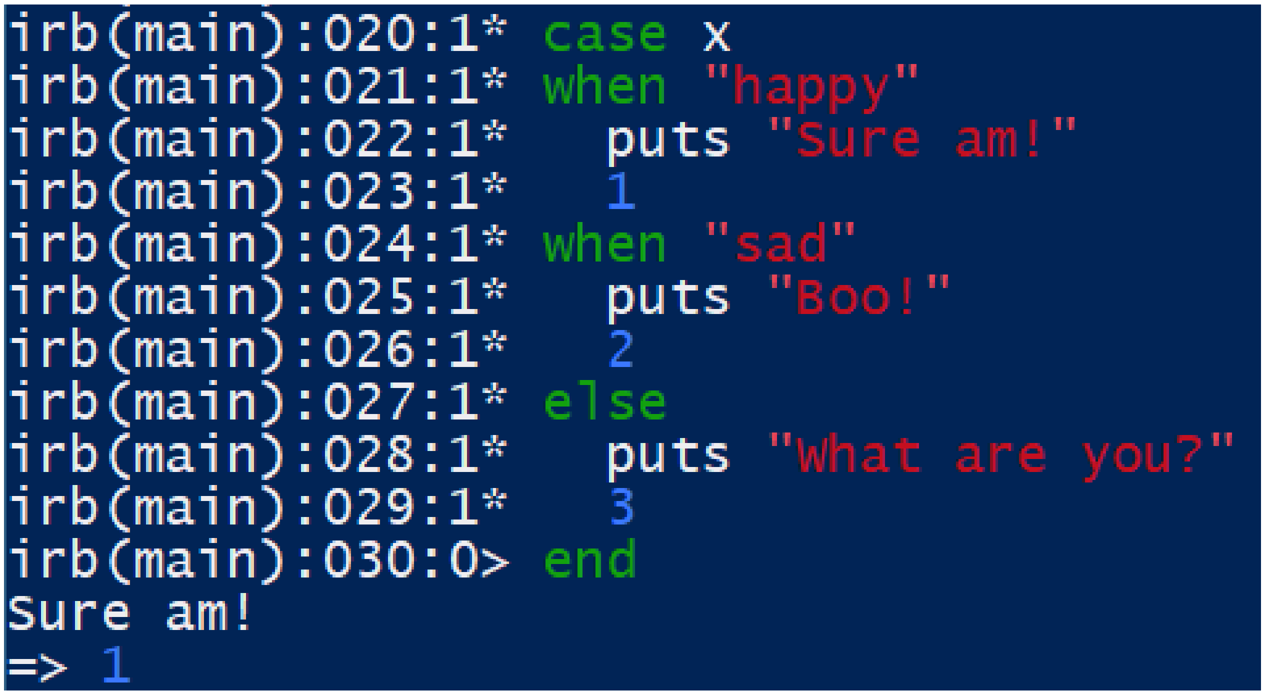


```
irb(main):010:0> x = "happy"
=> "happy"
irb(main):011:1* if x == "happy"
irb(main):012:1*   puts "Sure am!"
irb(main):013:1* elsif x == "sad"
irb(main):014:1*   puts "Boo!"
irb(main):015:1* else
irb(main):016:1*   puts "What are you?"
irb(main):017:0> end
Sure am!
=> nil
```

Key points:

- If/elsif/else is a commonly used conditional.
- What is uncommon - this construct has a return value of the return value of the last expression in the leg that matches.
- It is best to use the if/elsif/else statement when 3 or fewer options exist

12. Execute the following commands that introduce the case statement:

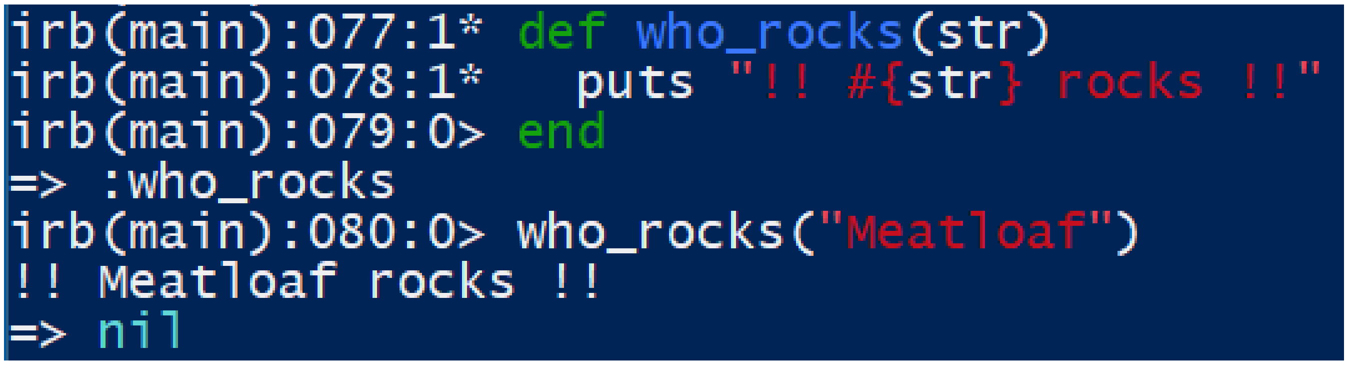


```
irb(main):020:1* case x
irb(main):021:1* when "happy"
irb(main):022:1*   puts "Sure am!"
irb(main):023:1*   1
irb(main):024:1* when "sad"
irb(main):025:1*   puts "Boo!"
irb(main):026:1*   2
irb(main):027:1* else
irb(main):028:1*   puts "What are you?"
irb(main):029:1*   3
irb(main):030:0> end
Sure am!
=> 1
```

Key points:

- Case is a shorter way to evaluate more than two conditionals on the same variable.
- Return values can be added as shown.
- It is best to use a case statement when more than 3 options exist

13. Execute the following commands to create a new method:

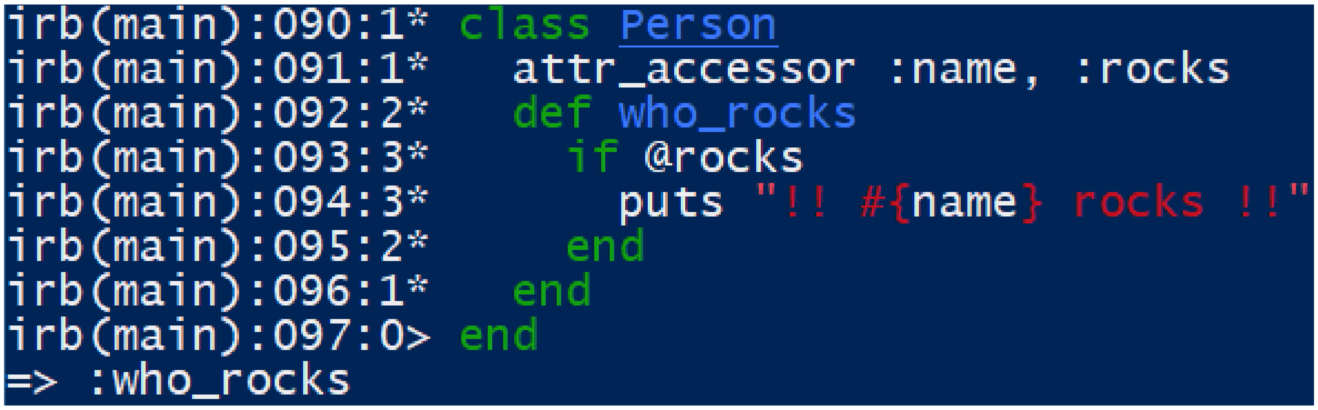


```
irb(main):077:1* def who_rocks(str)
irb(main):078:1*   puts "!! #{str} rocks !!"
irb(main):079:0> end
=> :who_rocks
irb(main):080:0> who_rocks("Meatloaf")
!! Meatloaf rocks !!
=> nil
```

Key points:

- Methods are used to bundle one or more repeatable statements into a single unit
- "**def**" defines a new method with a string input.
- Method names should begin with a lowercase letter.

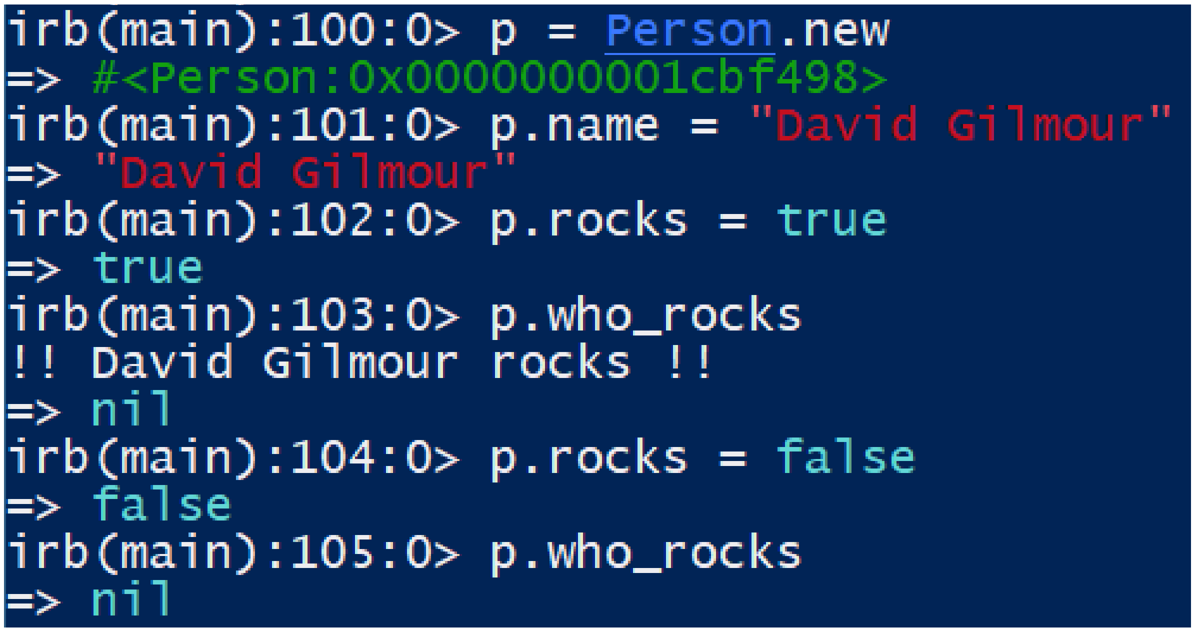
14. Execute the following commands to create a new class:



```
irb(main):090:1* class Person
irb(main):091:1*   attr_accessor :name, :rocks
irb(main):092:2*   def who_rocks
irb(main):093:3*     if @rocks
irb(main):094:3*       puts "!! #{name} rocks !!"
irb(main):095:2*     end
irb(main):096:1*   end
irb(main):097:0> end
=> :who_rocks
```

Key points:

- Ruby is object oriented, everything has a class.
- A class is a collection of properties with methods attached.
- attr_accessor creates two "getter/setter" (aka read/write) methods for a person. There also exists attr_reader & attr_writer
- The metal method is updated to use the @ variable. These are instance variables, they are different based on the object we create.

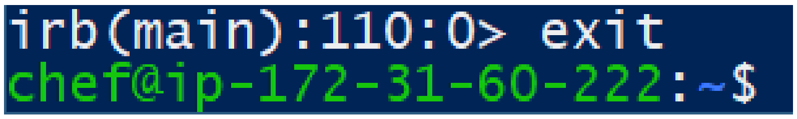


```
irb(main):100:0> p = Person.new
=> #<Person:0x000000001cbf498>
irb(main):101:0> p.name = "David Gilmour"
=> "David Gilmour"
irb(main):102:0> p.rocks = true
=> true
irb(main):103:0> p.who_rocks
!! David Gilmour rocks !!
=> nil
irb(main):104:0> p.rocks = false
=> false
irb(main):105:0> p.who_rocks
=> nil
```

Key points:

- Create a new instance of a class with "**new**" method.
- **p.name** and **p.rocks** set the values for the new object
- When **rocks** is true, who_rocks executes
- When **rocks** is false, who_rocks does not run

15. Exit IRB



```
irb(main):110:0> exit
chef@ip-172-31-60-222:~$
```

Notify your instructor that you are done with the lab

END OF LAB

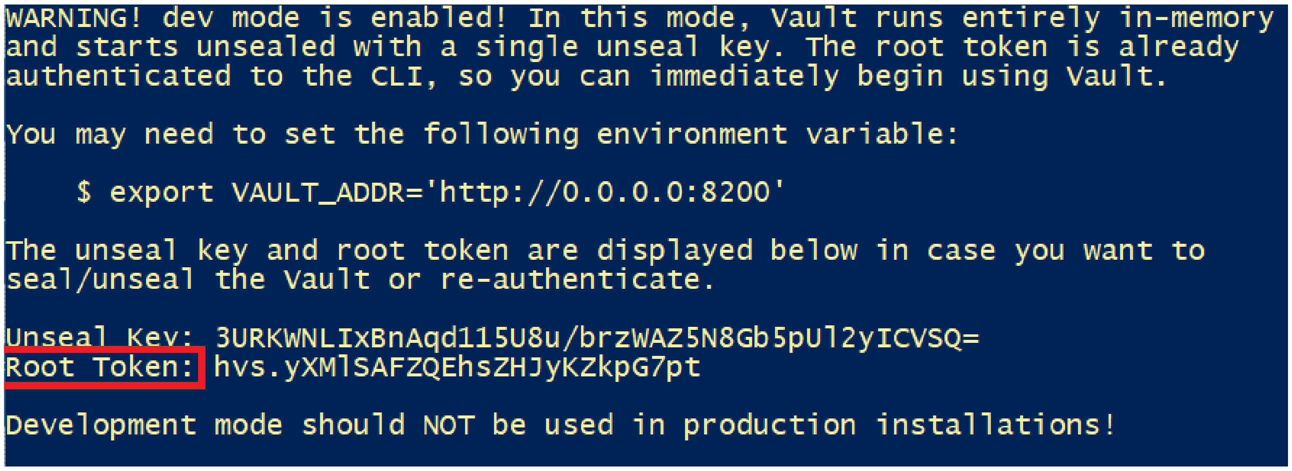
Lab 09, Objectives:

- Setup Hashicorp Vault
- Store a password
- Add ‘vault’ Ruby Gem
- Access password stored in vault

1. In order to create a vault for testing, execute the following command:

   $ vault server -dev -dev-listen-address=”0.0.0.0:8200”

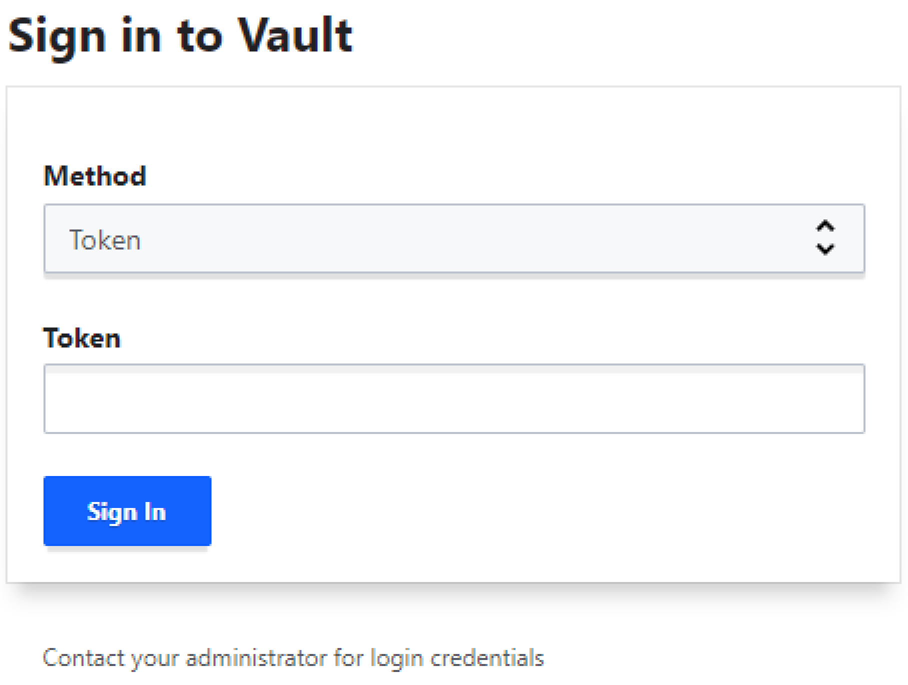
   Expected output:



```
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variable:

    $ export VAULT_ADDR='http://0.0.0.0:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: 3URKWNLIxBnAqd115U8u/brzWAZ5N8Gb5pUl2yICVSQ=
Root Token: hvs.yXMlSAFZQEhsZHJyKZkpG7pt

Development mode should NOT be used in production installations!
```

   Note: **do not end** the vault server or **close** the terminal window to ensure the server continues to run for the next steps.

2. Open a web browser and navigate to http://IP-ADDRESS:8200. A sign in window to the vault should be displaying:


Sign in to Vault

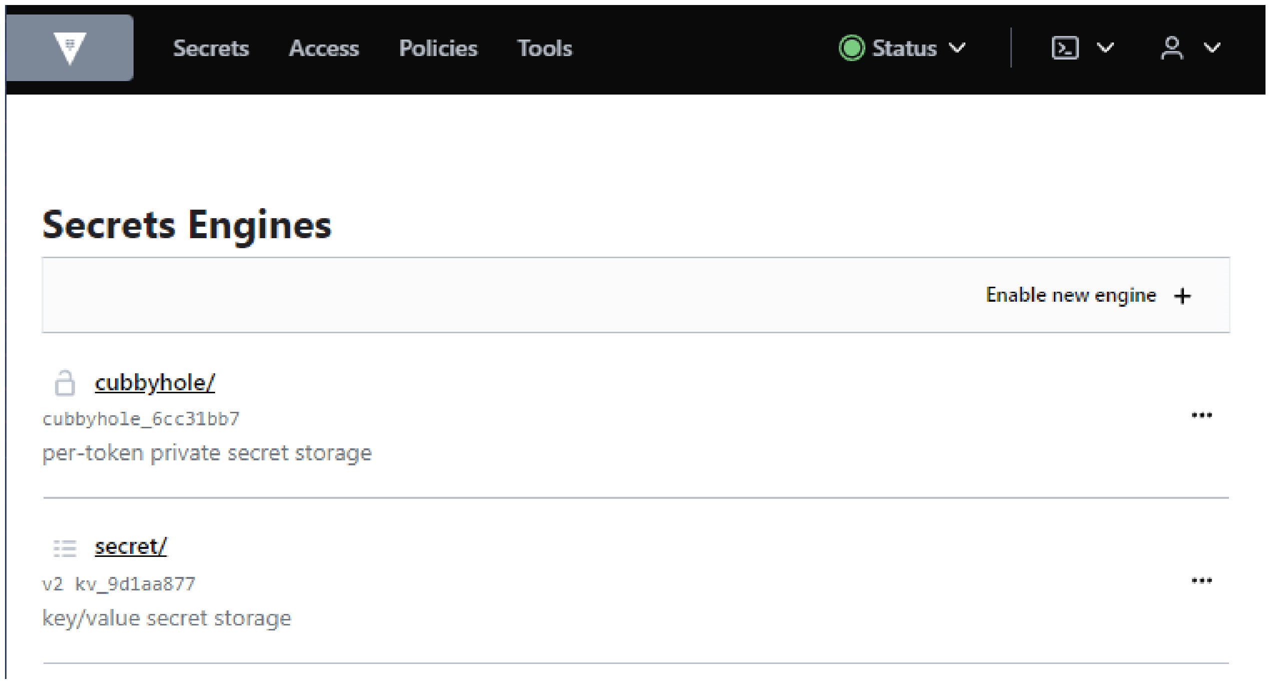
Method

Token

Token

Contact your administrator for login credentials

3. Use the **Root Token** to sign in. It is still visible in the terminal window running the vault server. With a successful sign in, the following page will display:



Secrets Engines

cubbyhole/
cubbyhole_6cc31bb7
per-token private secret storage

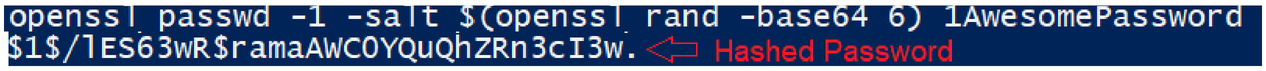
secret/
v2 kv_9d1aa877
key/value secret storage

4. Open a new terminal window (so as not to shut down your Vault server) and ssh into your lab machine

   Note: If you do not open a new terminal window, you will stop your vault server

5. We need a hashed password to store in Vault. Create one by executing the following command:

   $ openssl passwd -1 -salt $(openssl rand -base64 6) 1AwesomePassword



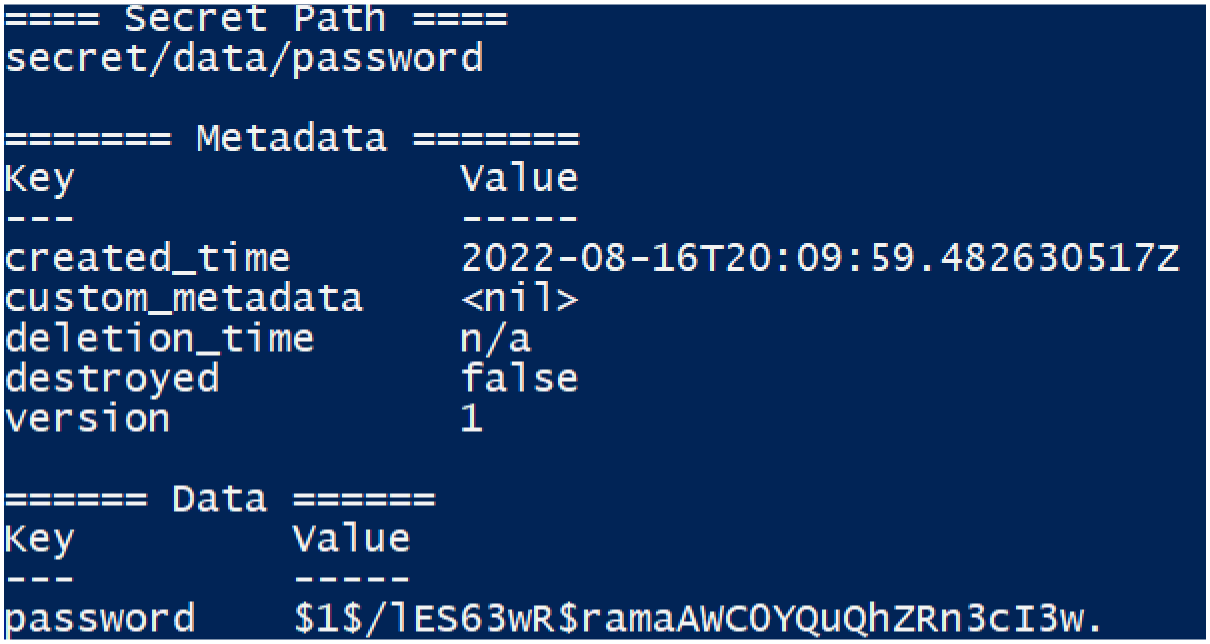
```
openssl passwd -1 -salt $(openssl rand -base64 6) 1AwesomePassword
$1$/lES63wR$ramaAWCOYQuQhZRn3cI3w.
```

6. Use the following vault command to save a hashed password to the vault:

   $ vault kv put secret/password password='HASHED_PASSWORD'

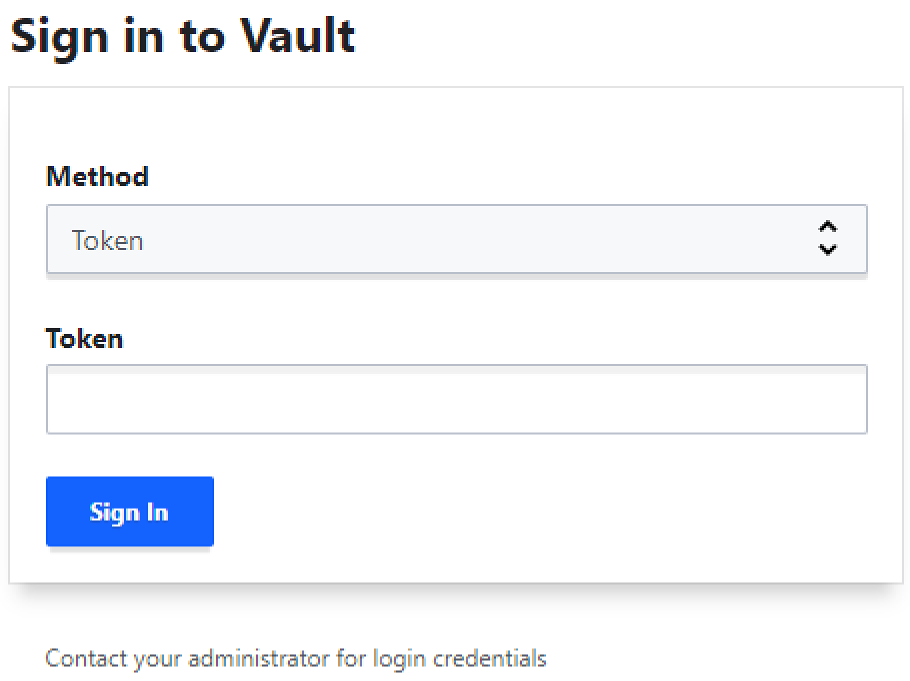
7. Verify the hashed password was stored correctly using the following command:

   $ vault kv get secret/password

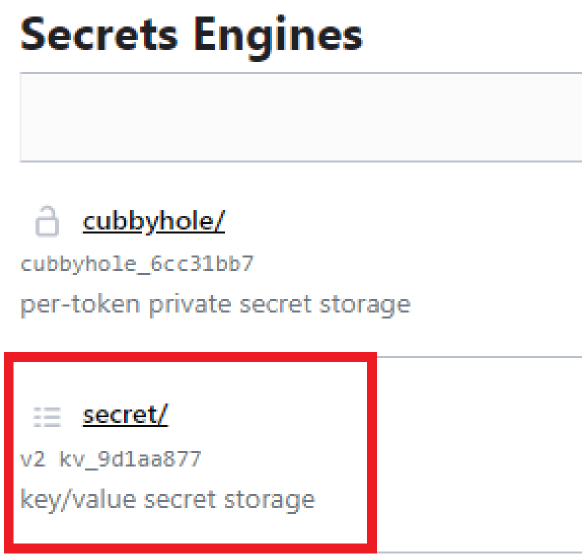


```
==== Secret Path ====
secret/data/password

======= Metadata =======
Key                Value
---                -----
created_time       2022-08-16T20:09:59.482630517Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

====== Data ======
Key         Value
---         -----
password    $1$/lES63wR$ramaAWCOYQuQhZRn3cI3w.
```

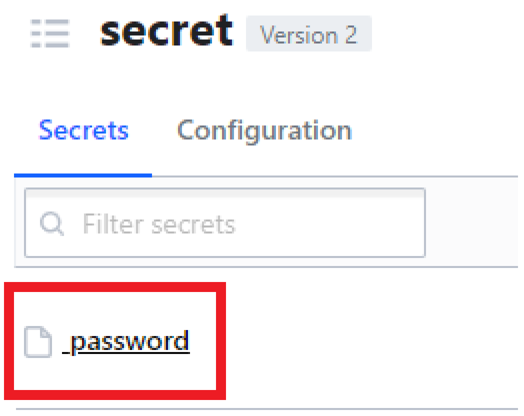
8. Verify the hashed password was stored correctly using web interface. Open a browser window to http://IP-ADDRESS:8200. Use the **Root Token** to sign in



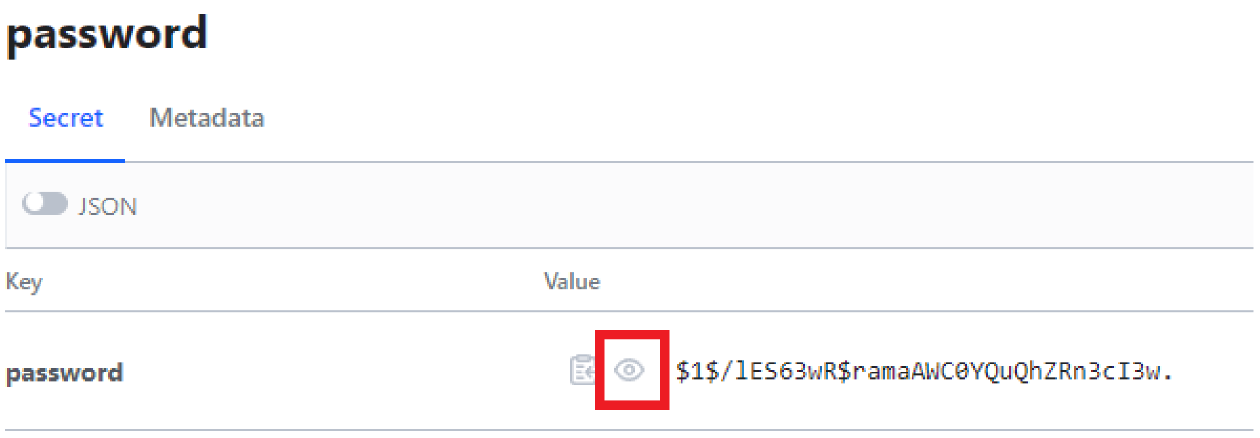
9. Click on secret/:



10. Click on password



11. Now click on eye button to view the hashed password:



Key	Value
password	$1$/lES63wR$ramaAWC0YQuQhZRn3cI3w.

12. Add the following code to the end of your default recipe:

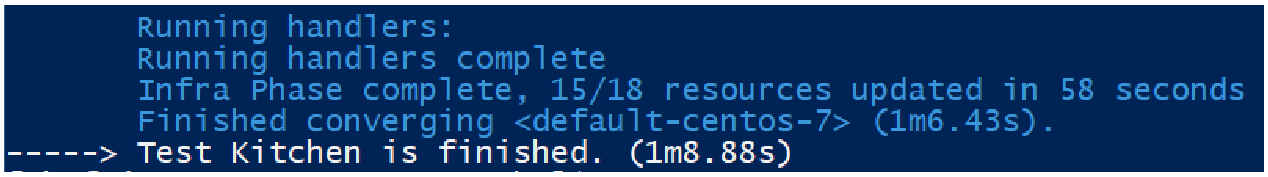
```
chef_gem 'vault' do
  compile_time true
end

require 'vault'

Vault.configure do |config|
  # The address of the Vault server
  config.address = 'http://<VAULT_SERVER_IPADDRESS>:8200'
  # The token to authenticate with Vault
  config.token = '<VAULT_ROOT_TOKEN>'
end

creds = Vault.kv('secret').read('password')

user 'WebAdmin' do
  comment 'Web Admin'
  uid 2000
  home '/home/WebAdmin'
  shell '/bin/bash'
  manage_home true
  password creds.data[:password]
end
```

This block of code contains:

- **Chef_gem:** A Chef resource to download a Ruby gem.
- **Require vault:** This recipe requires vault dependency.
- **Vault.configure:** Configures the vault to access secrets.
- **Vault.kv("secret").read("password")** is used to read credentials.
- **Creds.data[:password]** is used to filter a hashed password in a vault server.

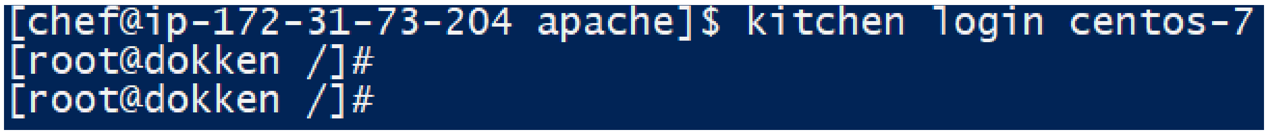
NOTE: It's not good practice to use the vault server root token in a production server.

13. Execute **kitchen converge** to apply the updated configuration to a new virtual machine



```
       Running handlers:
       Running handlers complete
       Infra Phase complete, 15/18 resources updated in 58 seconds
       Finished converging <default-centos-7> (1m6.43s).
-----> Test Kitchen is finished. (1m8.88s)
```

14. Execute **kitchen login centos-7** to login to the centos virtual machine



```
[chef@ip-172-31-73-204 apache]$ kitchen login centos-7
[root@dokken /]#
[root@dokken /]#
```

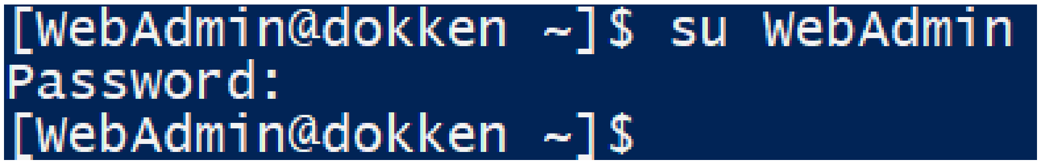
15. You will be logged in as root, change to the new WebAdmin user with the following command:

    # su WebAdmin

16. Test the password for WebAdmin with the command below. First use the correct password, then use a bad password

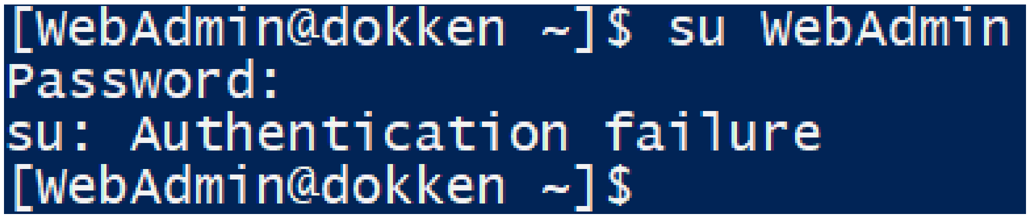
    $ su Webadmin

    Using the correct password, '*1AwesomePassword*'



```
[WebAdmin@dokken ~]$ su WebAdmin
Password:
[WebAdmin@dokken ~]$
```

    Using an incorrect password:



```
[WebAdmin@dokken ~]$ su WebAdmin
Password:
su: Authentication failure
[WebAdmin@dokken ~]$
```

17. Execute the **exit** command twice to logout of the WebAdmin account and the centos-7 virtual machine.

Notify your instructor that you are done with the lab

END OF LAB

Lab 10, Objectives:

- Create a custom resource in the apache cookbook

Custom resources help clarify recipes by hiding some implementation details which makes the code more concise.

A **VHOST** (virtual host) is an Apache configuration that allows multiple websites to run on a single web server. By using vhosts, one web server could serve hundreds or even thousands of different websites, based on each vhost's unique port number (eg. 8080 instead of the default 80).

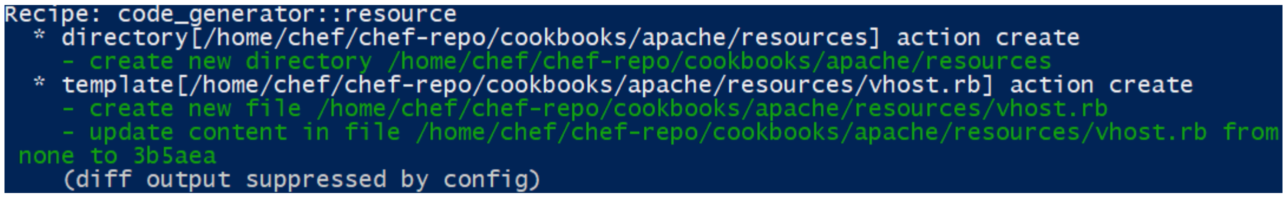
Each request to a predefined, unique port number serves up its own **index.html** page, which is stored in a directory defined by the vhost configuration file
(e.g. **/srv/apache/admins/index.html**)

Creating a new **VHOST** requires these steps**:**

- Create a directory to house the new **index.html** page for this specific vhost configuration
- Add a configuration file defining:
  - The file locations from which the Apache service will serve files
  - The port number on which Apache will listen for requests
- Create the **vhost specific index.html** file
  - A custom resource is able to be reused and allows for multiple vhosts to be created quickly on the web server.
  - The custom resources are similar to those defined in the user cookbook's default recipe.

1. Starting in **~/chef-repo/cookbooks/apache**, generate a new custom resource named **vhost** by executing:

   $ chef generate resource vhost



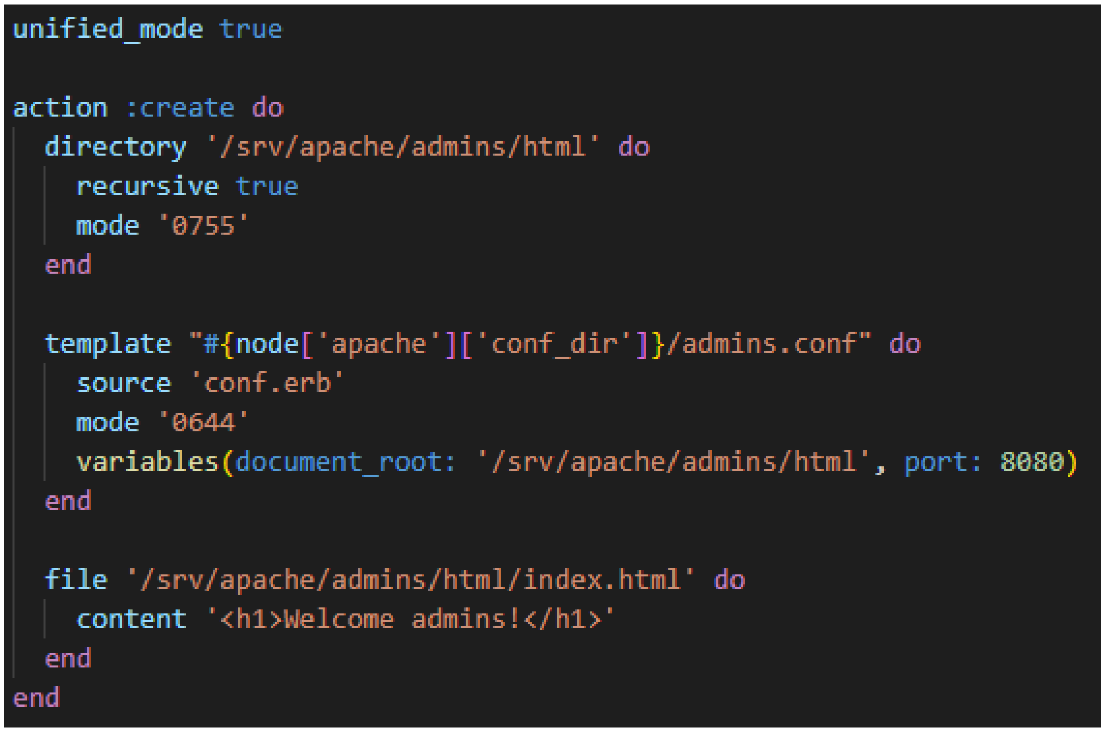
```
Recipe: code_generator::resource
  * directory[/home/chef/chef-repo/cookbooks/apache/resources] action create
    - create new directory /home/chef/chef-repo/cookbooks/apache/resources
  * template[/home/chef/chef-repo/cookbooks/apache/resources/vhost.rb] action create
    - create new file /home/chef/chef-repo/cookbooks/apache/resources/vhost.rb
    - update content in file /home/chef/chef-repo/cookbooks/apache/resources/vhost.rb from
 none to 3b5aea
    (diff output suppressed by config)
```

   The first implementation for the custom resource will create an admin site exactly as was done in the default recipe for the apache cookbook

   These values are hard-coded to the admin site for now, this will be improved in a later module.

2. Update **~/chef-repo/cookbooks/apache/resources/vhost.rb** as follows:

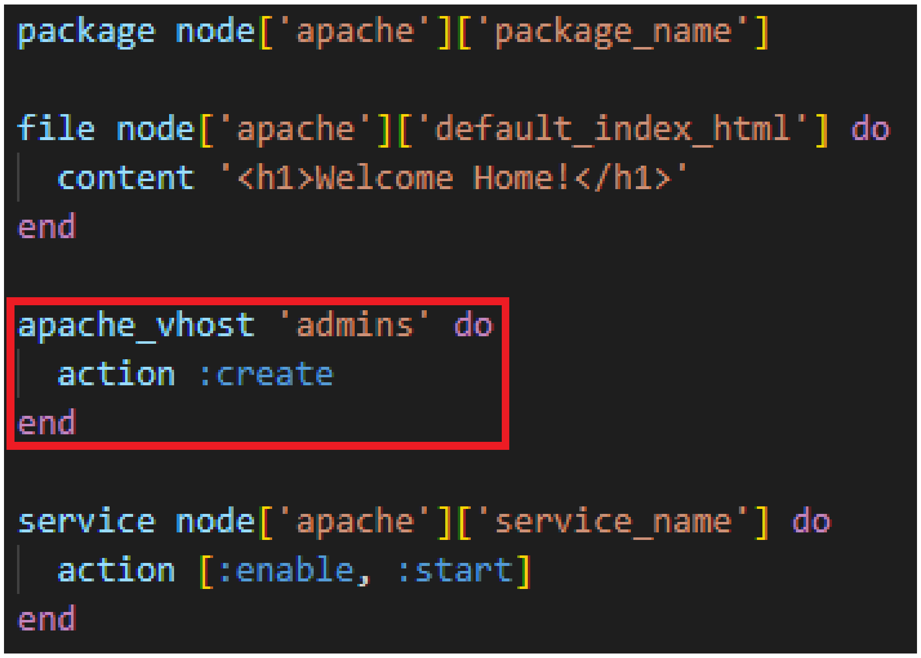
```
unified_mode true


action :create do
  directory '/srv/apache/admins/html' do
    recursive true
    mode '0755'
  end

  template "#{node['apache']['conf_dir']}/admins.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: '/srv/apache/admins/html',
port: 8080)
  end

  file '/srv/apache/admins/html/index.html' do
    content '<h1>Welcome admins!</h1>'
  end
end
```



```
unified_mode true

action :create do
  directory '/srv/apache/admins/html' do
    recursive true
    mode '0755'
  end

  template "#{node['apache']['conf_dir']}/admins.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: '/srv/apache/admins/html', port: 8080)
  end

  file '/srv/apache/admins/html/index.html' do
    content '<h1>Welcome admins!</h1>'
  end
end
```

3. Add the **apache_vhost** resource below to **recipes/default.rb** after the **file** resource

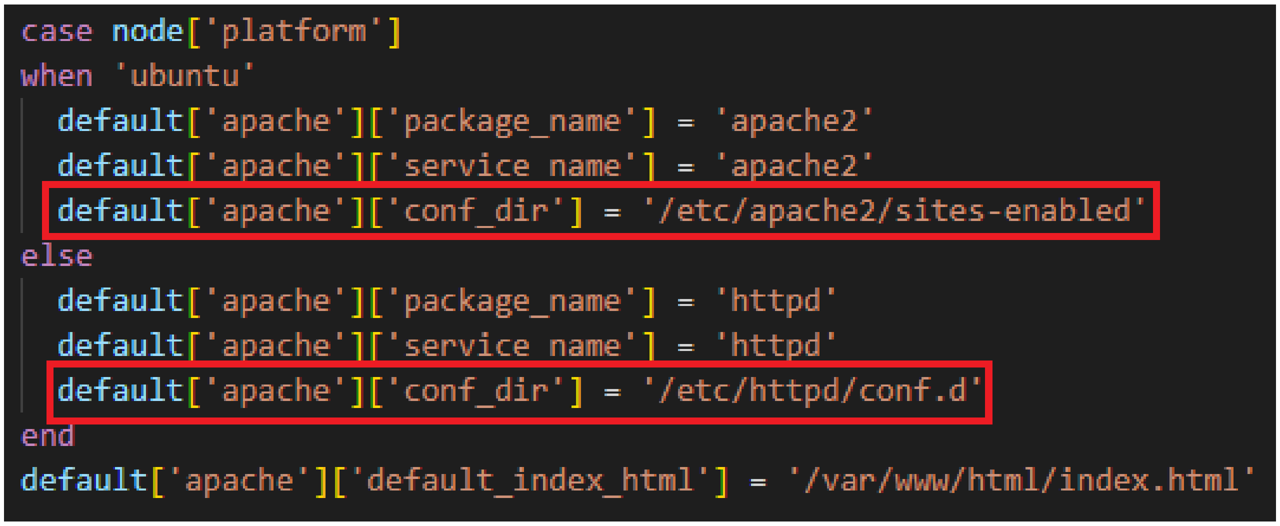
```
apache_vhost 'admins' do
  action :create
end
```



```
package node['apache']['package_name']

file node['apache']['default_index_html'] do
  content '<h1>Welcome Home!</h1>'
end

apache_vhost 'admins' do
  action :create
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

4. Add the following two lines to the case statement in your **attributes/default.rb** file:

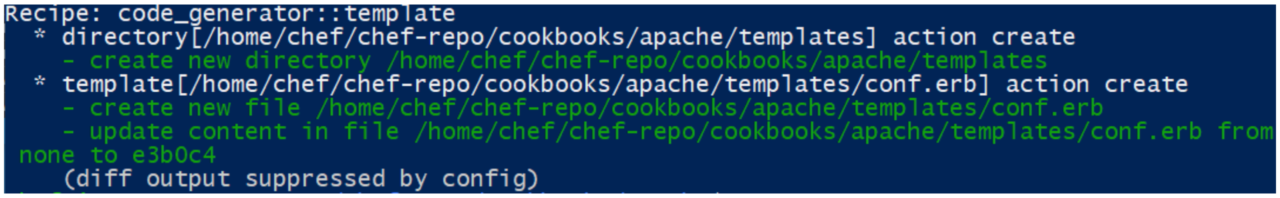
```
default['apache']['conf_dir'] = '/etc/apache2/sites-
enabled'

default['apache']['conf_dir'] = '/etc/httpd/conf.d'
```



```
case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
  default['apache']['conf_dir'] = '/etc/apache2/sites-enabled'
else
  default['apache']['package_name'] = 'httpd'
  default['apache']['service_name'] = 'httpd'
  default['apache']['conf_dir'] = '/etc/httpd/conf.d'
end
default['apache']['default_index_html'] = '/var/www/html/index.html'
```

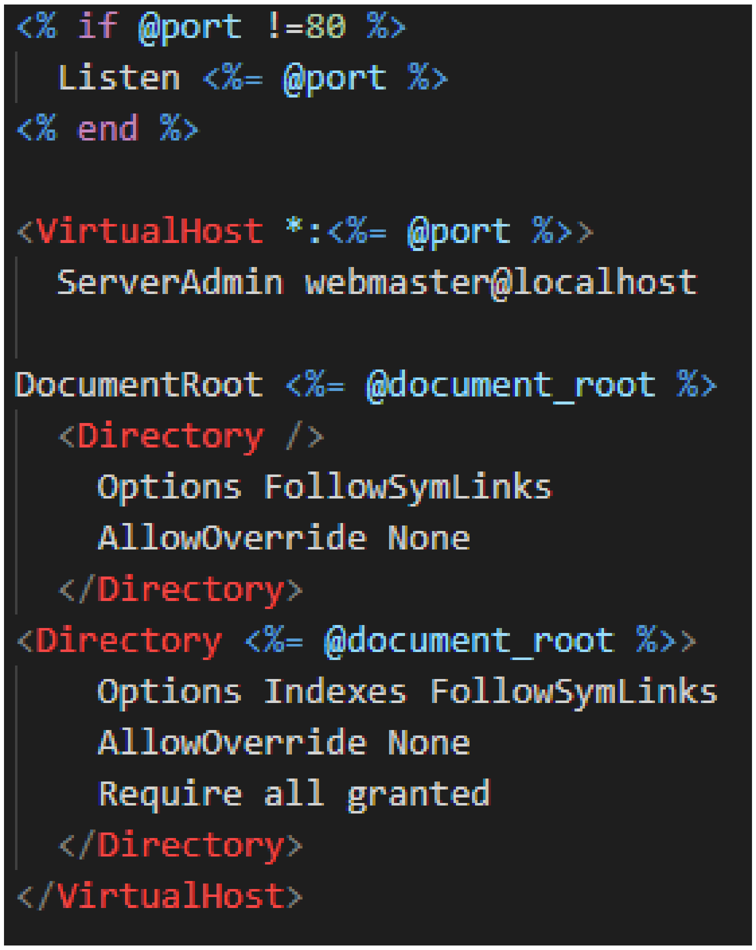
5. From **~/chef-repo/cookbooks/apache**, generate a new template named **conf** by executing:

$ chef generate template conf



```
Recipe: code_generator::template
  * directory[/home/chef/chef-repo/cookbooks/apache/templates] action create
    - create new directory /home/chef/chef-repo/cookbooks/apache/templates
  * template[/home/chef/chef-repo/cookbooks/apache/templates/conf.erb] action create
    - create new file /home/chef/chef-repo/cookbooks/apache/templates/conf.erb
    - update content in file /home/chef/chef-repo/cookbooks/apache/templates/conf.erb from
 none to e3b0c4
    (diff output suppressed by config)
```

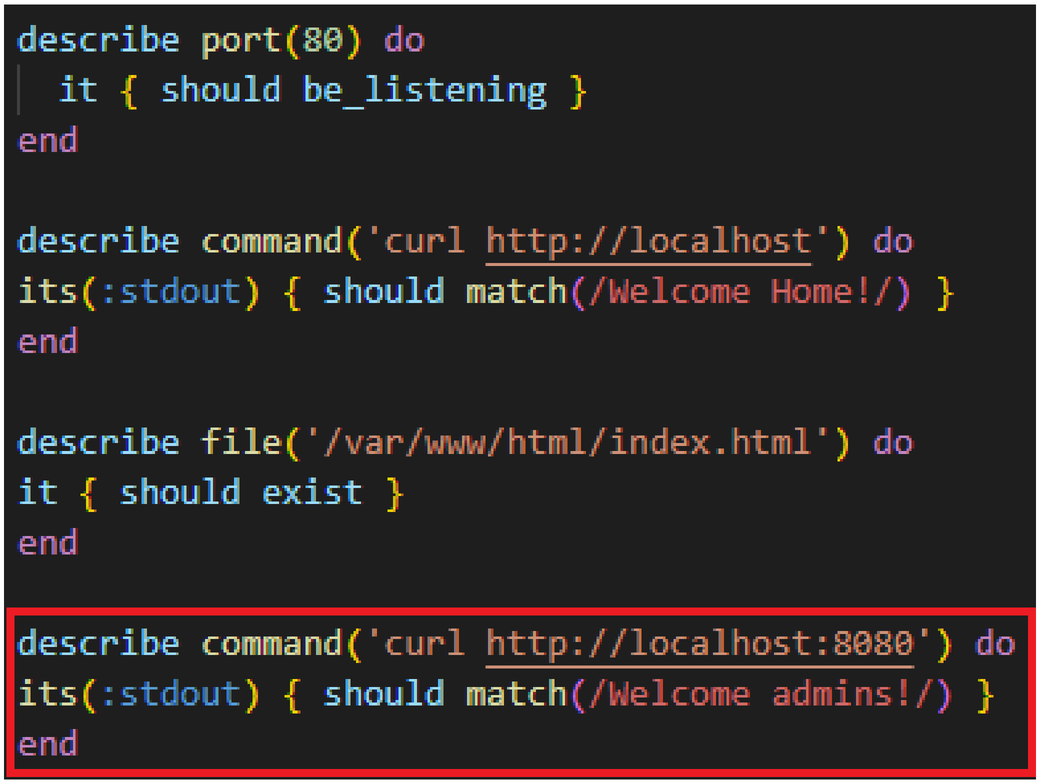
6. Update **templates/conf.erb** as follows:



```
<% if @port !=80 %>
  Listen <%= @port %>
<% end %>

<VirtualHost *:<%= @port %>>
  ServerAdmin webmaster@localhost

DocumentRoot <%= @document_root %>
  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
<Directory <%= @document_root %>>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
  </Directory>
</VirtualHost>
```

```
<% if @port !=80 %>
  Listen <%= @port %>
<% end %>

<VirtualHost *:<%= @port %>>
  ServerAdmin webmaster@localhost

DocumentRoot <%= @document_root %>
  <Directory />
    Options FollowSymLinks
    AllowOverride None
  </Directory>
<Directory <%= @document_root %>>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
  </Directory>
</VirtualHost>
```

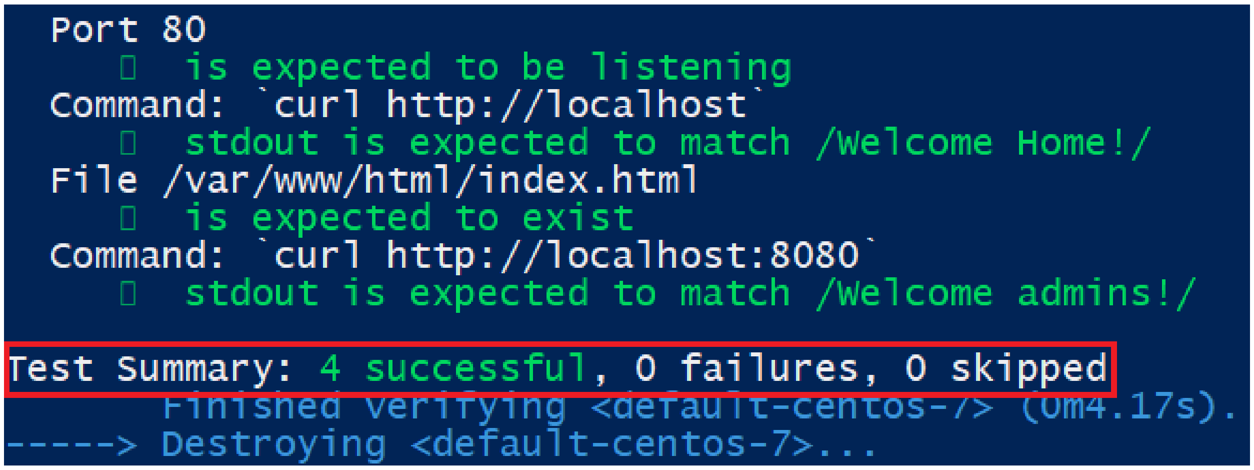
7. Add a new test to **test/integration/default/default_test.rb,** to test the following:



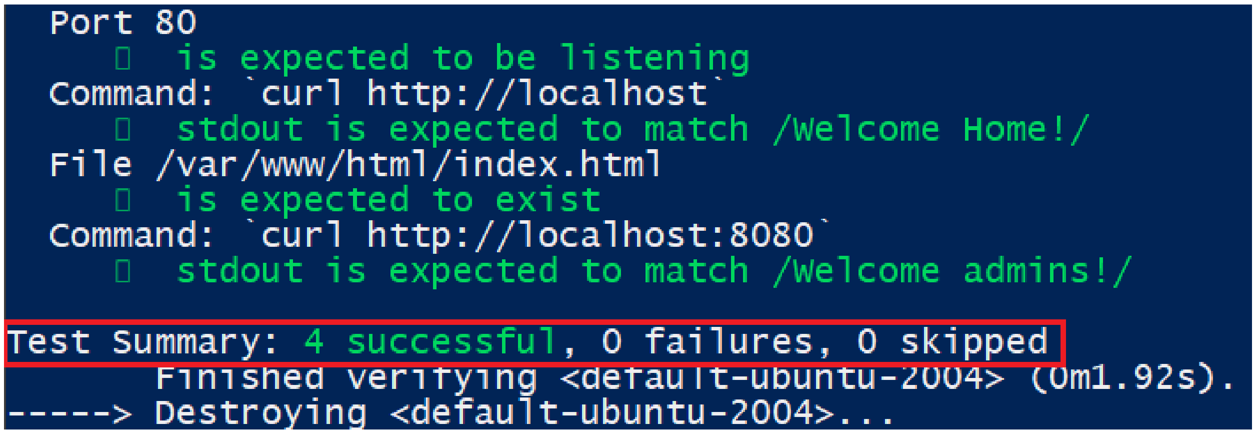
```
describe port(80) do
  it { should be_listening }
end

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

describe file('/var/www/html/index.html') do
it { should exist }
end

describe command('curl http://localhost:8080') do
its(:stdout) { should match(/Welcome admins!/) }
end
```

8. Execute **kitchen test** and verify that all tests are passing.

$ kitchen test



```
  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
  Command: `curl http://localhost:8080`
     ✓  stdout is expected to match /Welcome admins!/

Test Summary: 4 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m4.17s).
-----> Destroying <default-centos-7>...
```



```
  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
  Command: `curl http://localhost:8080`
     stdout is expected to match /Welcome admins!/

Test Summary: 4 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.92s).
-----> Destroying <default-ubuntu-2004>...
```

Notify your instructor that you are done with the lab

END OF LAB

Lab 11, Objectives:

- Using properties to make a custom resource more flexible within a recipe

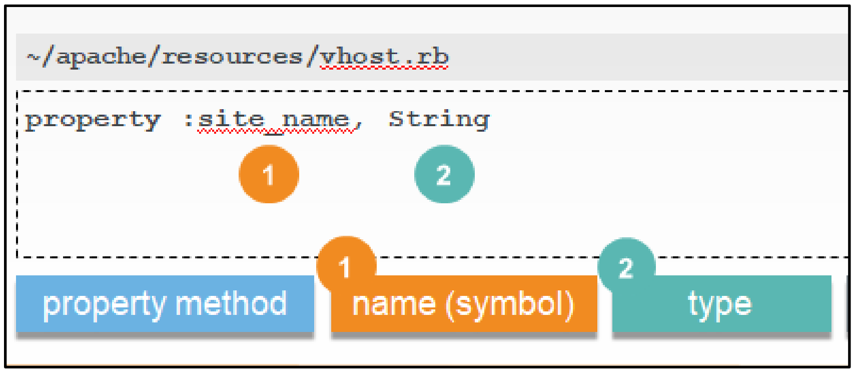
  Removing hard-coded values specific to the admin site makes for a dynamically updated site. This can be done through properties defined in the custom resource

  The vhost name, 'admins', is hardcoded throughout 'resources/vhost.rb' and 'recipe/default.rb'. This makes the custom resource inflexible and difficult to change.

  Properties:

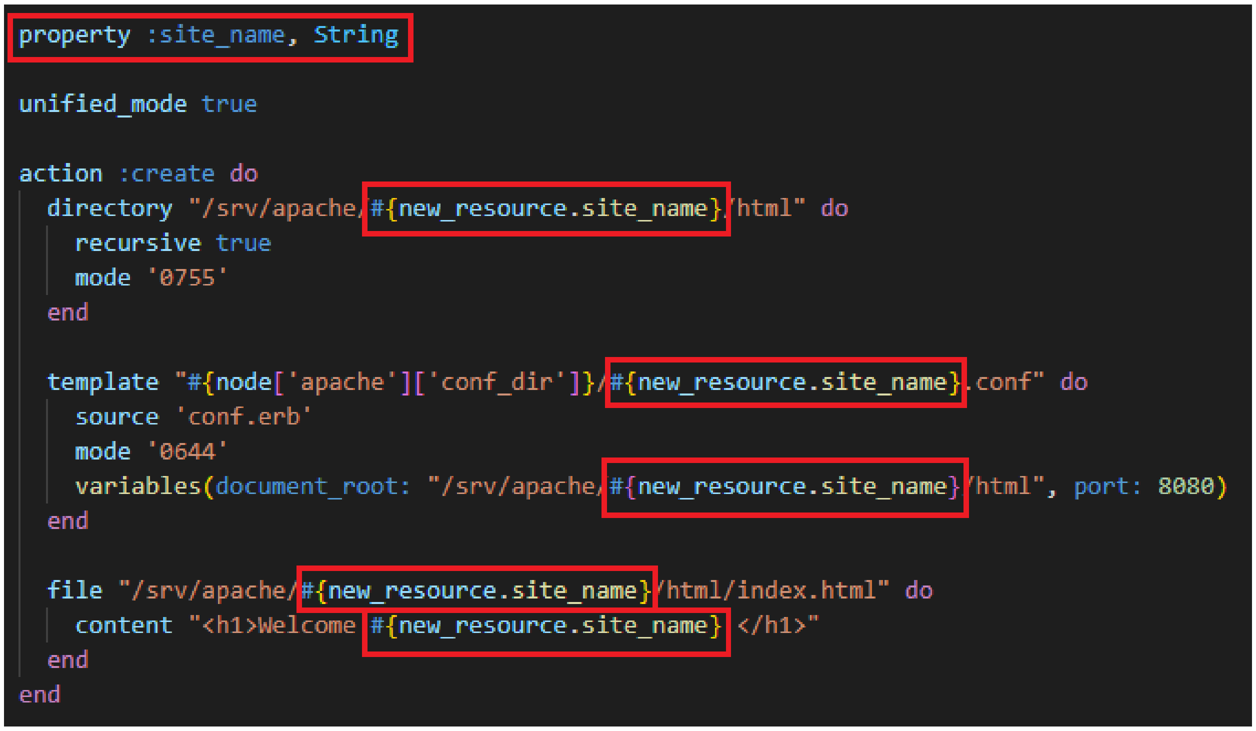
    - Properties are defined in the same file as the resource actions
    - Generally, they are defined at the top of the file to make them more visible
    - A property is defined by specifying a method named property with two required parameters and a third set of optional parameters
    - The name of the property is defined as a Ruby Symbol
    - The type of the property is a Ruby class name. This type enforces what kind of values are supported by this property; typically, it is a String for text and an Integer for numbers
    - The optional parameters are defined as a Hash

  Properties are defined with the following syntax:



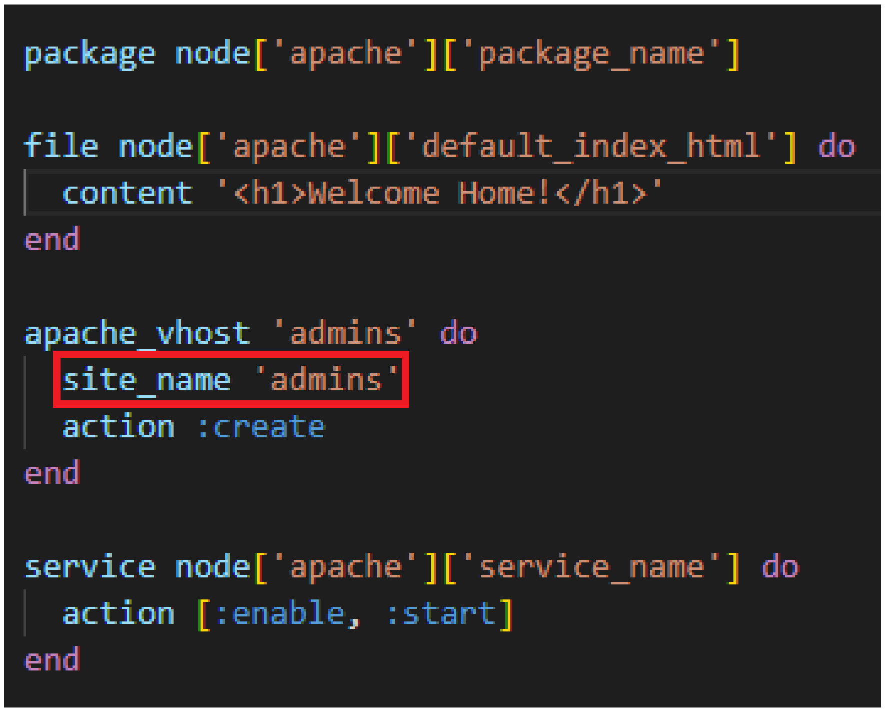
```
~/apache/resources/vhost.rb

property :site_name, String
```

1. Update the **apache/resources/vhost.rb** file as shown. Don't forget to change single quotes to double quotes where needed.



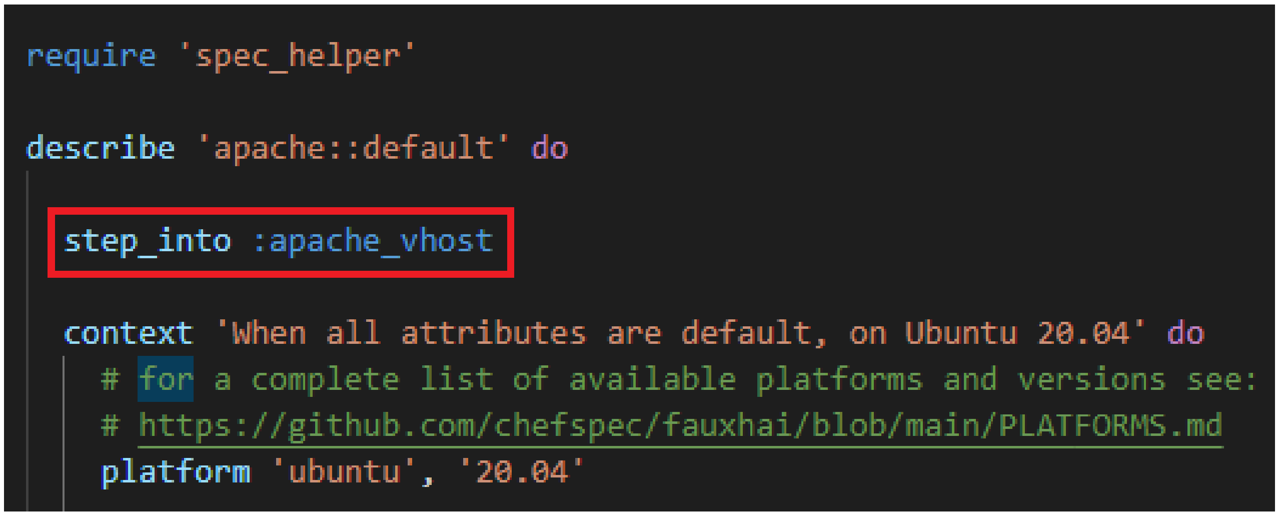
```
property :site_name, String

unified_mode true

action :create do
  directory "/srv/apache/#{new_resource.site_name}/html" do
    recursive true
    mode '0755'
  end

  template "#{node['apache']['conf_dir']}/#{new_resource.site_name}.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/apache/#{new_resource.site_name}/html", port: 8080)
  end

  file "/srv/apache/#{new_resource.site_name}/html/index.html" do
    content "<h1>Welcome #{new_resource.site_name} </h1>"
  end
end
```

2. Update the **apache_vhost** resource in **apache/recipes/default.rb** to use the newly established **site_name** property:



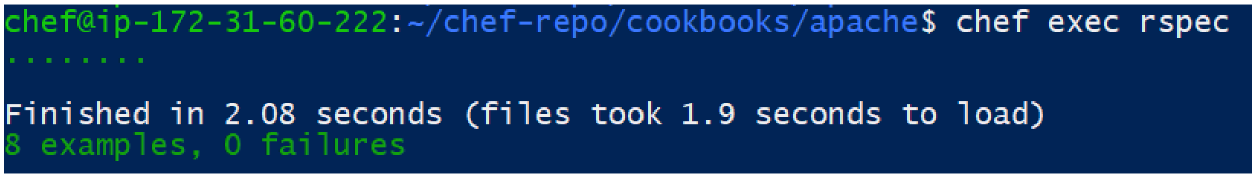
```
package node['apache']['package_name']

file node['apache']['default_index_html'] do
  content '<h1>Welcome Home!</h1>'
end

apache_vhost 'admins' do
  site_name 'admins'
  action :create
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

3. Add the **step_into** method to **spec/unit/recipes/default_spec.rb**:



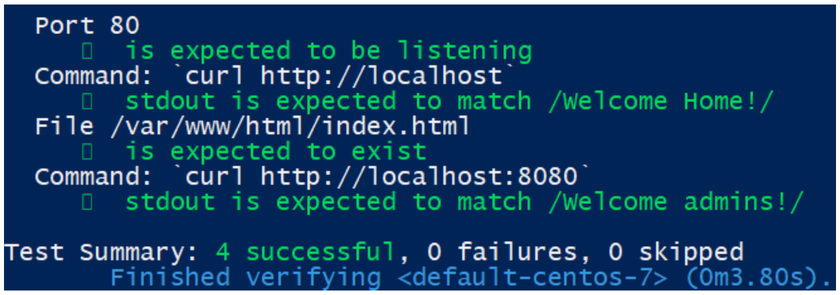
```
require 'spec_helper'

describe 'apache::default' do

  step_into :apache_vhost

  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/main/PLATFORMS.md
    platform 'ubuntu', '20.04'
```

4. Execute **chef exec rspec** to verify the unit tests still pass:

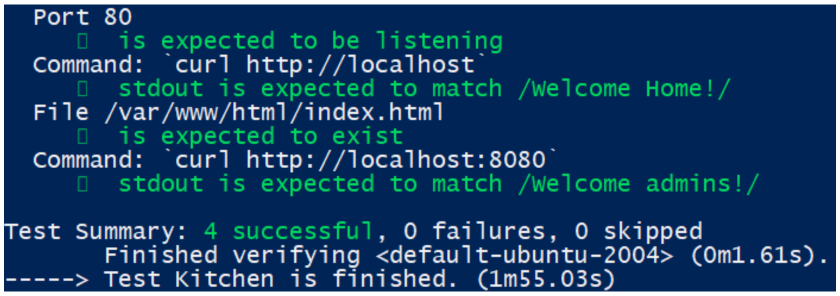


```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
........

Finished in 2.08 seconds (files took 1.9 seconds to load)
8 examples, 0 failures
```

5. Execute **kitchen verify** to ensure that your code still works. If the tests for both platforms pass, then the site_name property has been added successfully.

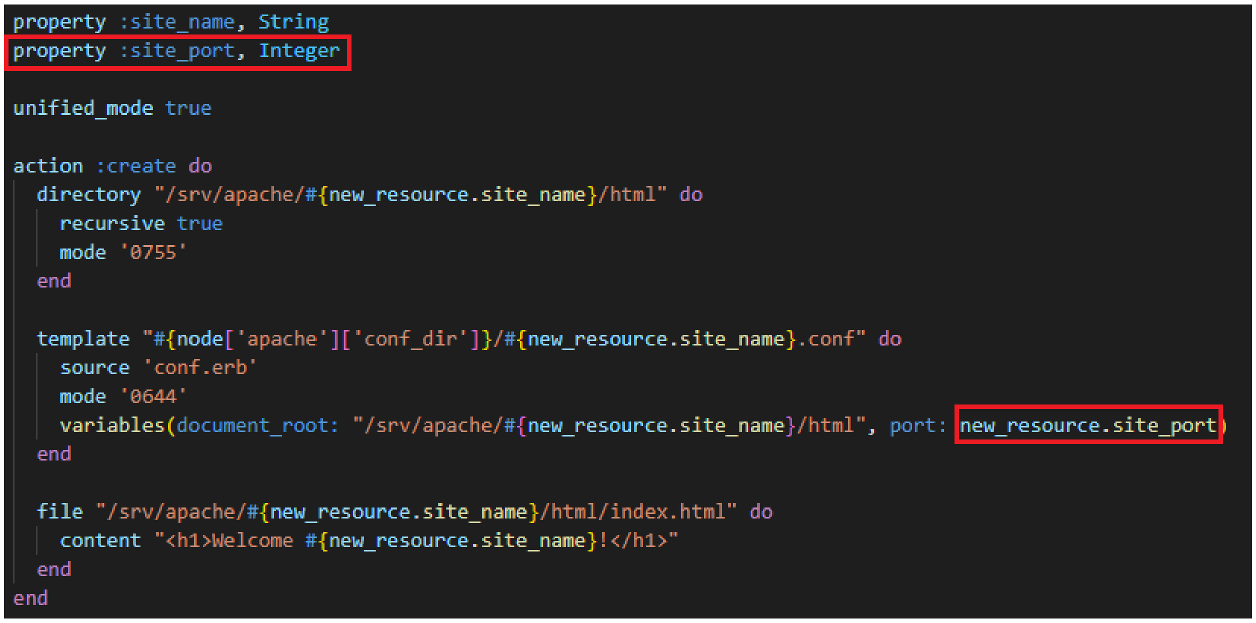


```
  Port 80
     is expected to be listening
  Command: `curl http://localhost`
     stdout is expected to match /Welcome Home!/
  File /var/www/html/index.html
     is expected to exist
  Command: `curl http://localhost:8080`
     stdout is expected to match /Welcome admins!/

Test Summary: 4 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.80s).
```



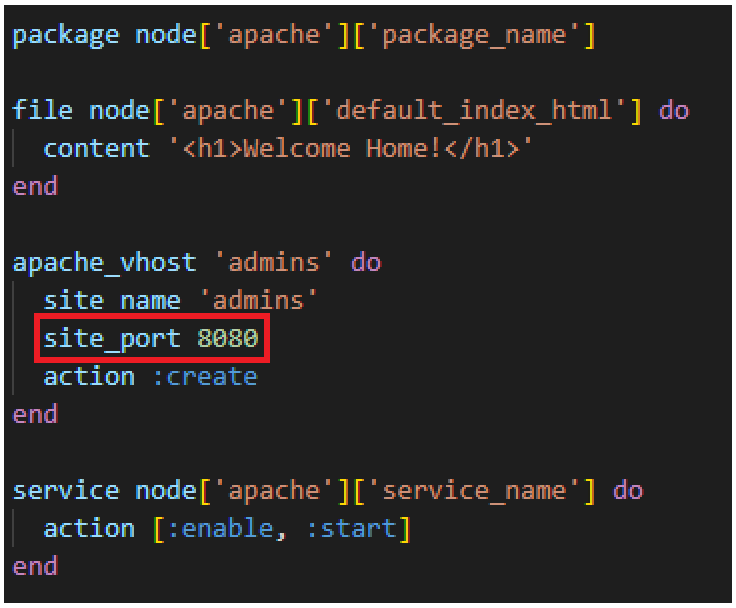
```
  Port 80
     is expected to be listening
  Command: `curl http://localhost`
     stdout is expected to match /Welcome Home!/
  File /var/www/html/index.html
     is expected to exist
  Command: `curl http://localhost:8080`
     stdout is expected to match /Welcome admins!/

Test Summary: 4 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.61s).
-----> Test Kitchen is finished. (1m55.03s)
```

6. Create another property named **site_port** in **apache/resources/vhost.rb** and use the new property in the **template** resource:



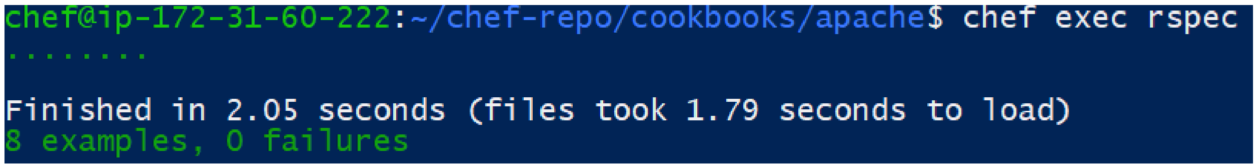
```ruby
property :site_name, String
property :site_port, Integer

unified_mode true

action :create do
  directory "/srv/apache/#{new_resource.site_name}/html" do
    recursive true
    mode '0755'
  end

  template "#{node['apache']['conf_dir']}/#{new_resource.site_name}.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/apache/#{new_resource.site_name}/html", port: new_resource.site_port)
  end

  file "/srv/apache/#{new_resource.site_name}/html/index.html" do
    content "<h1>Welcome #{new_resource.site_name}!</h1>"
  end
end
```

7. Update the **apache_vhost** resource in the default recipe to use the **site_port** property:



```ruby
package node['apache']['package_name']

file node['apache']['default_index_html'] do
  content '<h1>Welcome Home!</h1>'
end

apache_vhost 'admins' do
  site_name 'admins'
  site_port 8080
  action :create
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

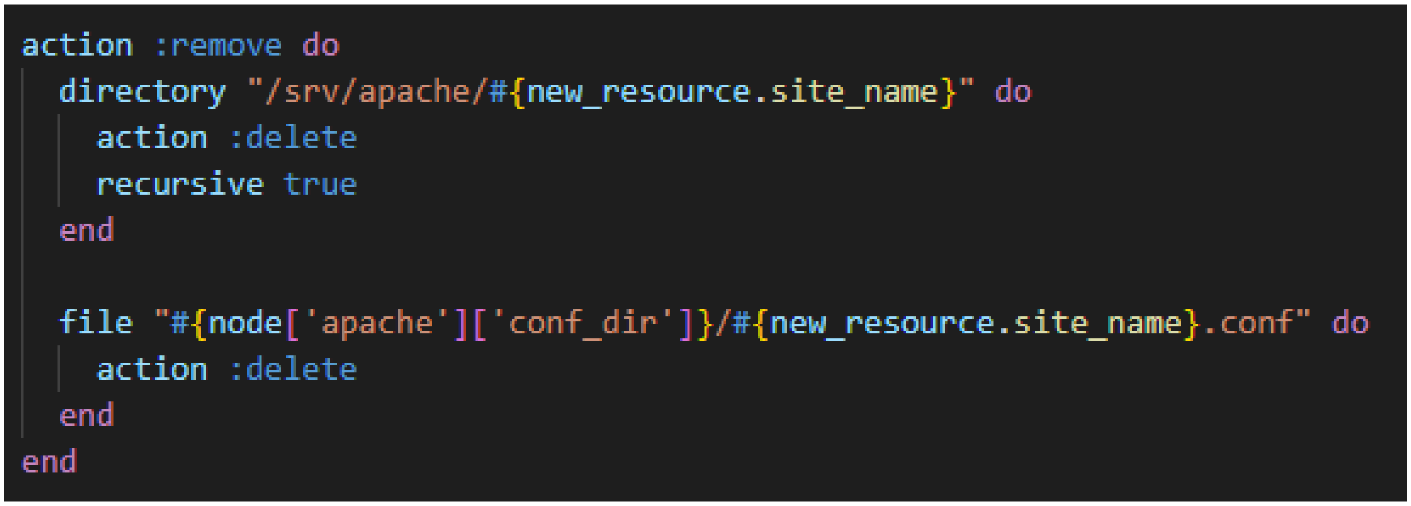
8. Execute **chef exec rspec** to verify the changes are working as expected.



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
........

Finished in 2.05 seconds (files took 1.79 seconds to load)
8 examples, 0 failures
```

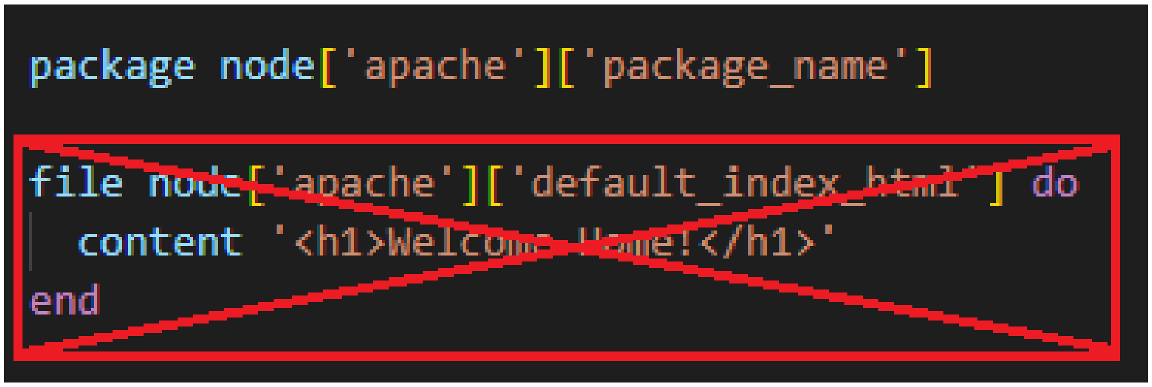
9. Create a new action, **remove**, in the **apache/resources/vhost.rb** file.

```
action :remove do
  directory "/srv/apache/#{new_resource.site_name}" do
    action :delete
    recursive true
  end

  file
"#{node['apache']['conf_dir']}/#{new_resource.site_name}.co
nf" do
    action :delete
  end
end
```



```
action :remove do
  directory "/srv/apache/#{new_resource.site_name}" do
    action :delete
    recursive true
  end

  file "#{node['apache']['conf_dir']}/#{new_resource.site_name}.conf" do
    action :delete
  end
end
```

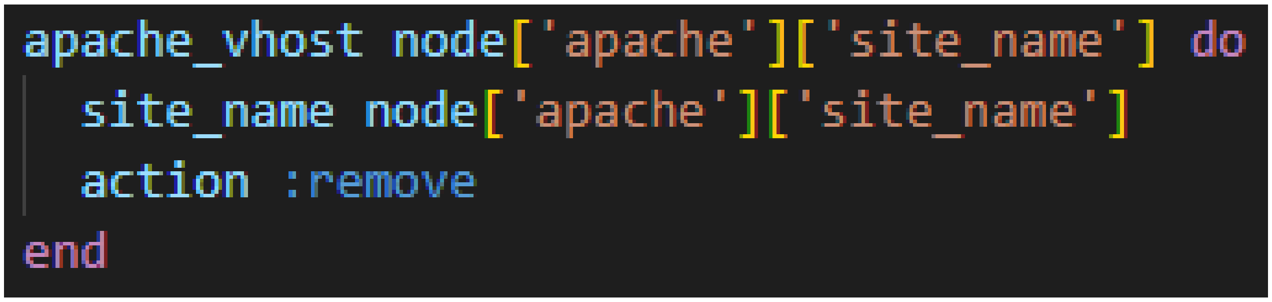
10. Remove the **file** resource from **apache/recipes/default.rb**:



```
package node['apache']['package_name']

file node['apache']['default_index_html'] do
  content '<h1>Welcome Home!</h1>'
end
```

11. Update **apache/recipes/default.rb** to include a new **apache_vhost** resource that uses the **remove** action.

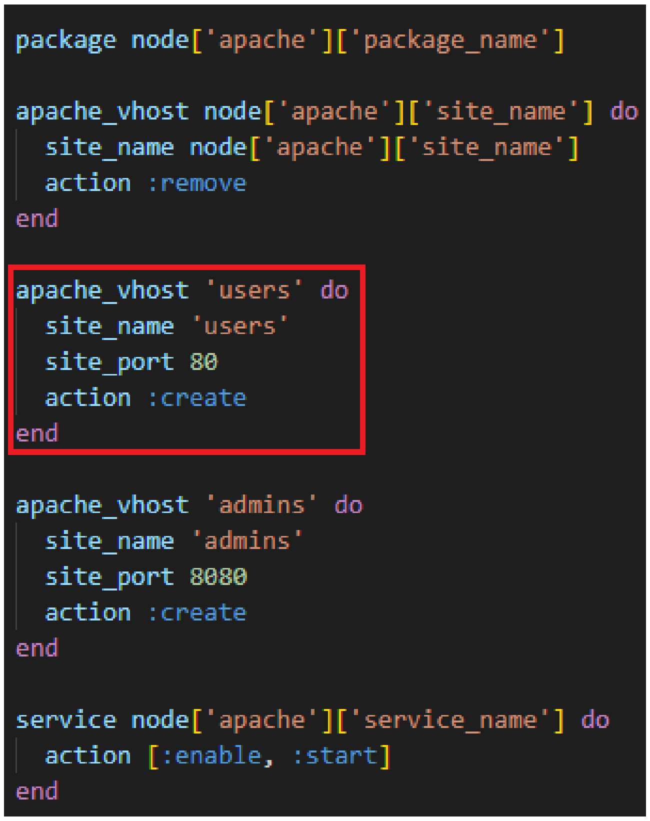
```
apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
end
```



```
apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
end
```

12. Update **apache/recipes/default.rb** to include a new **apache_vhost** resource that creates a new site for users.

```
apache_vhost 'users' do
  site_name 'users'
  site_port 80
  action :create
end
```

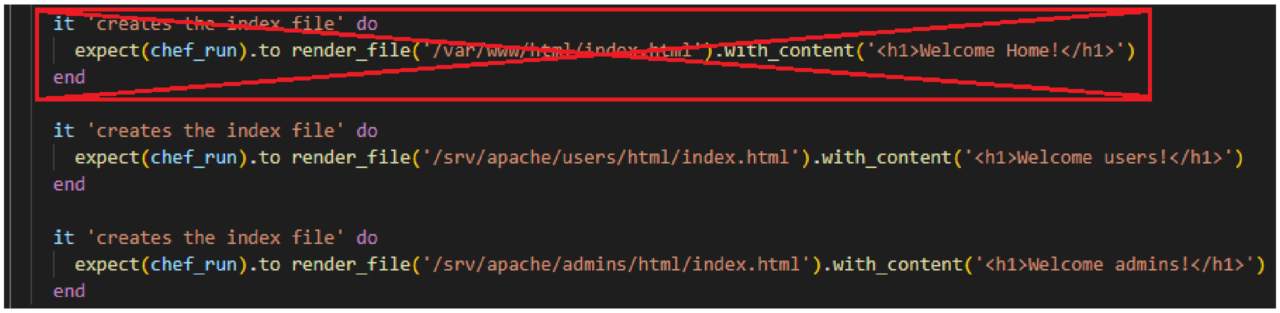


```
package node['apache']['package_name']

apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
end

apache_vhost 'users' do
  site_name 'users'
  site_port 80
  action :create
end

apache_vhost 'admins' do
  site_name 'admins'
  site_port 8080
  action :create
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

13. Create two new tests in **spec/unit/recipes/default_spec.rb**. The first will test for an html file that says “Welcome admins!” when rendering ‘/srv/apache/admins/html/index.html’. The second will test for an html file that says “Welcome users!” when rendering ‘/srv/apache/users/html/index.html’

...Answer at the bottom of the lab…

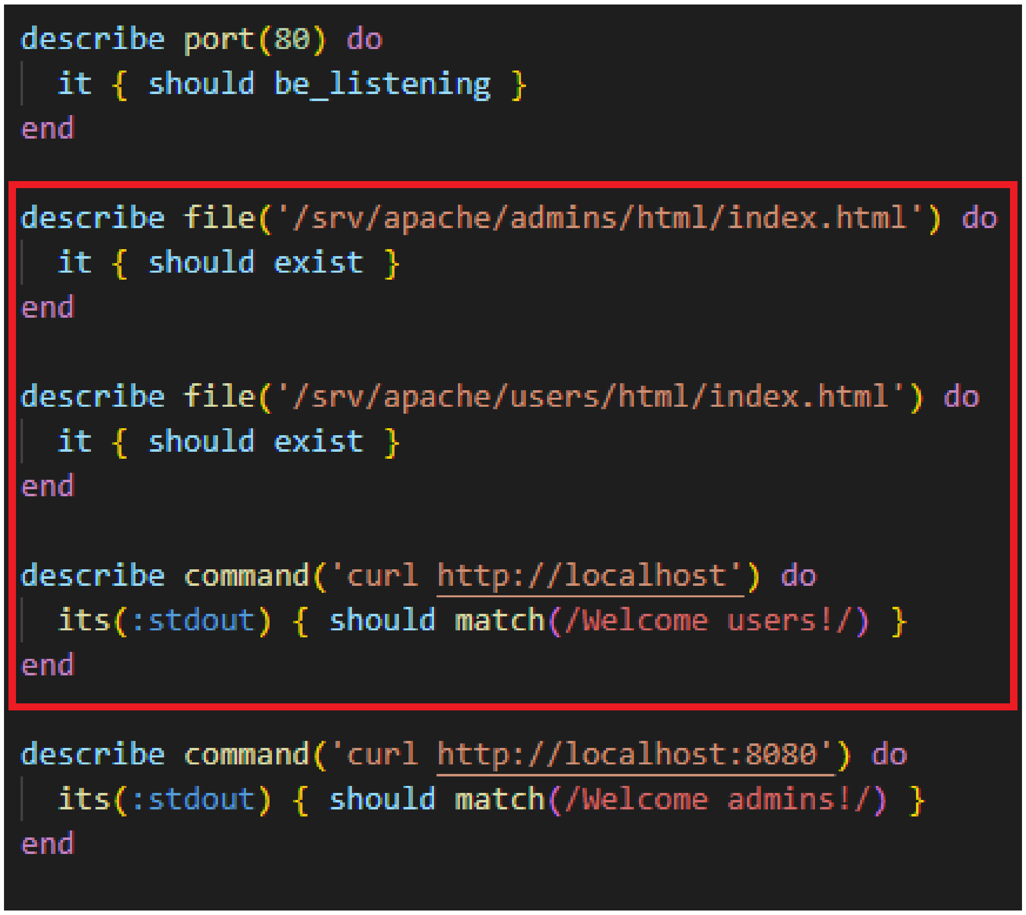
14. Remove the unit test looking for an html file that says “Welcome Home” when rendering ‘/var/www/html/index.html’.



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

it 'creates the index file' do
  expect(chef_run).to render_file('/srv/apache/users/html/index.html').with_content('<h1>Welcome users!</h1>')
end

it 'creates the index file' do
  expect(chef_run).to render_file('/srv/apache/admins/html/index.html').with_content('<h1>Welcome admins!</h1>')
end
```

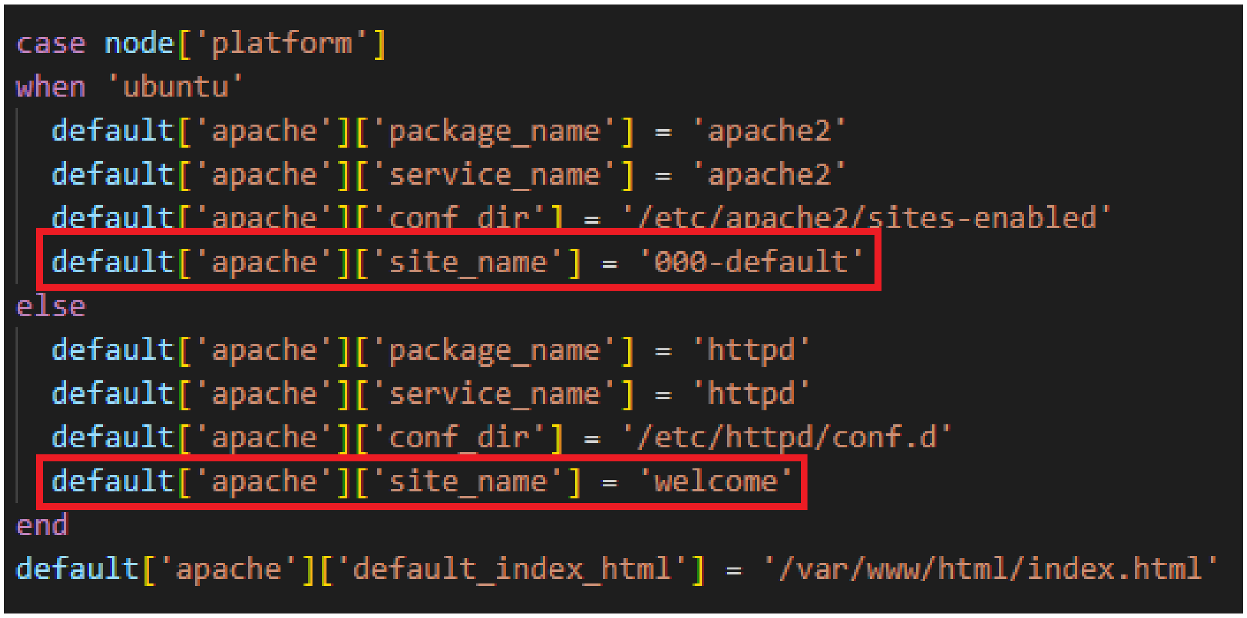
15. Make two updates to **test/integration/default/default_test.rb**. First remove the test for the file **/var/www/html/index.html**. Then add the tests shown below:

```ruby
describe file('/srv/apache/admins/html/index.html') do
  it { should exist }
end

describe file('/srv/apache/users/html/index.html') do
  it { should exist }
end

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



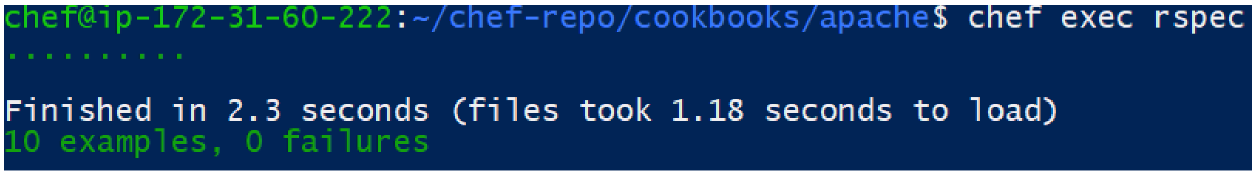
```
describe port(80) do
  it { should be_listening }
end

describe file('/srv/apache/admins/html/index.html') do
  it { should exist }
end

describe file('/srv/apache/users/html/index.html') do
  it { should exist }
end

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

describe command('curl http://localhost:8080') do
  its(:stdout) { should match(/Welcome admins!/) }
end
```

16. Update the **attributes/default.rb** file to include two additional attributes:

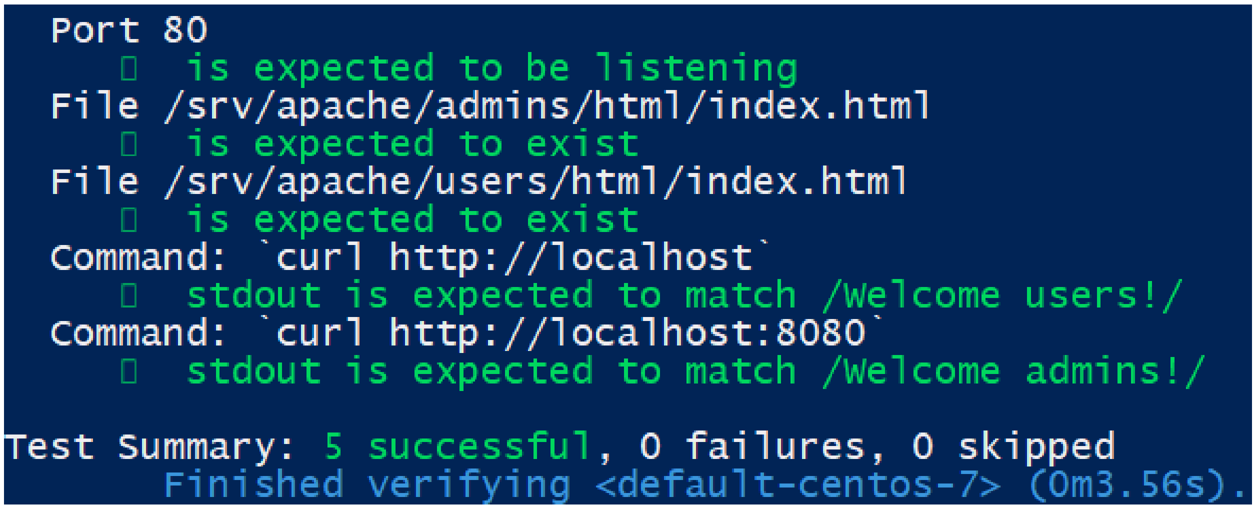


```
case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
  default['apache']['conf_dir'] = '/etc/apache2/sites-enabled'
  default['apache']['site_name'] = '000-default'
else
  default['apache']['package_name'] = 'httpd'
  default['apache']['service_name'] = 'httpd'
  default['apache']['conf_dir'] = '/etc/httpd/conf.d'
  default['apache']['site_name'] = 'welcome'
end
default['apache']['default_index_html'] = '/var/www/html/index.html'
```

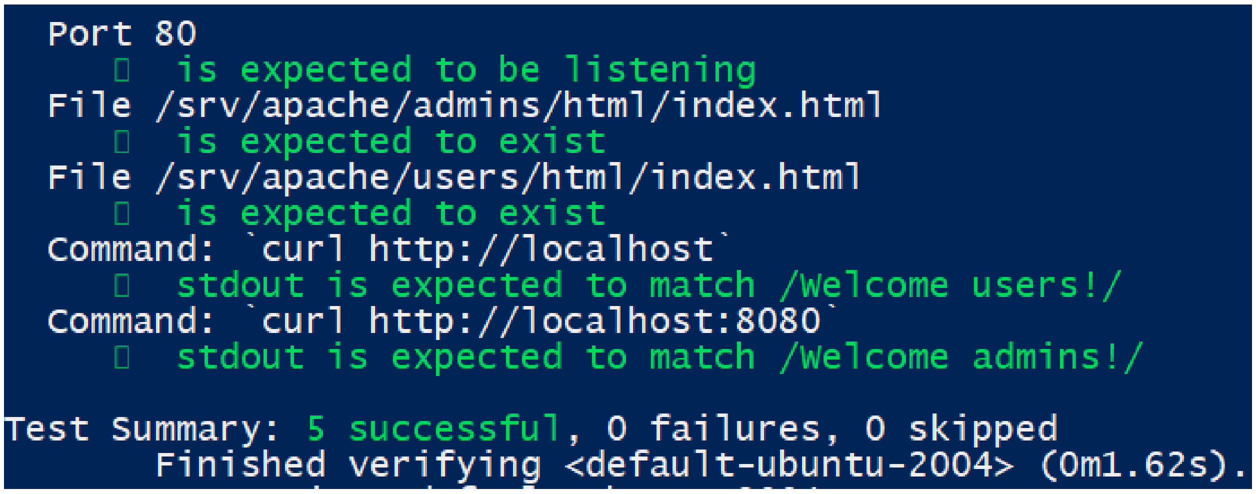
17. Execute **chef exec rspec** and **kitchen test** to verify everything is working as expected.



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..........

Finished in 2.3 seconds (files took 1.18 seconds to load)
10 examples, 0 failures
```



```
  Port 80
       is expected to be listening
  File /srv/apache/admins/html/index.html
       is expected to exist
  File /srv/apache/users/html/index.html
       is expected to exist
  Command: `curl http://localhost`
       stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
       stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.56s).
```



```
  Port 80
       is expected to be listening
  File /srv/apache/admins/html/index.html
       is expected to exist
  File /srv/apache/users/html/index.html
       is expected to exist
  Command: `curl http://localhost`
       stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
       stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.62s).
```

Notify your instructor that you are done with the lab

END OF LAB...

...to view the solution, keep on scrolling

. . . .

. . . .

. . . .

. . . .

. . . .

. . . . . .

. . . . .

. . . .

. . .

. .

.

Keep scrolling...

. . . .

. . . .

. . . .

. . . .

. . . .

. . . . . .

. . . . .

. . . .

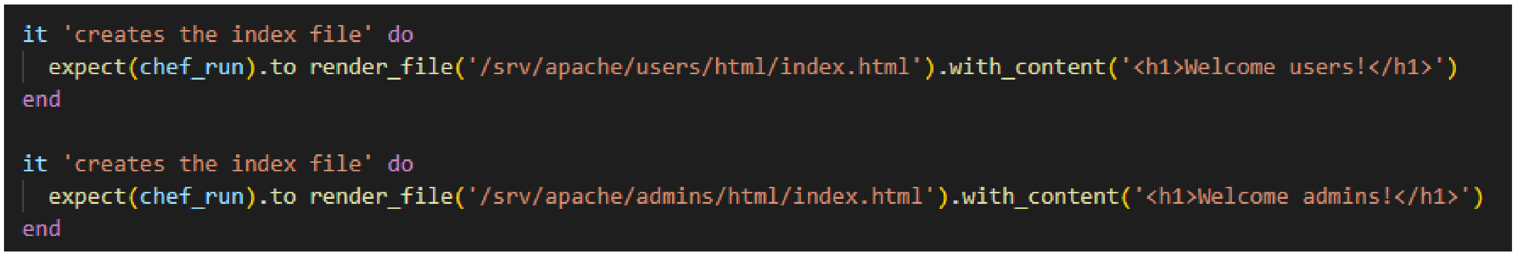
. . .

. .

.

Answer to Step 13. This should be placed in each platform section:

```
it 'creates the index file' do
  expect(chef_run).to
render_file('/srv/apache/users/html/index.html').with_content('<
h1>Welcome users!</h1>')
end

it 'creates the index file' do
  expect(chef_run).to
render_file('/srv/apache/admins/html/index.html').with_content('
<h1>Welcome admins!</h1>')
end
```



```
it 'creates the index file' do
  expect(chef_run).to render_file('/srv/apache/users/html/index.html').with_content('<h1>Welcome users!</h1>')
end

it 'creates the index file' do
  expect(chef_run).to render_file('/srv/apache/admins/html/index.html').with_content('<h1>Welcome admins!</h1>')
end
```

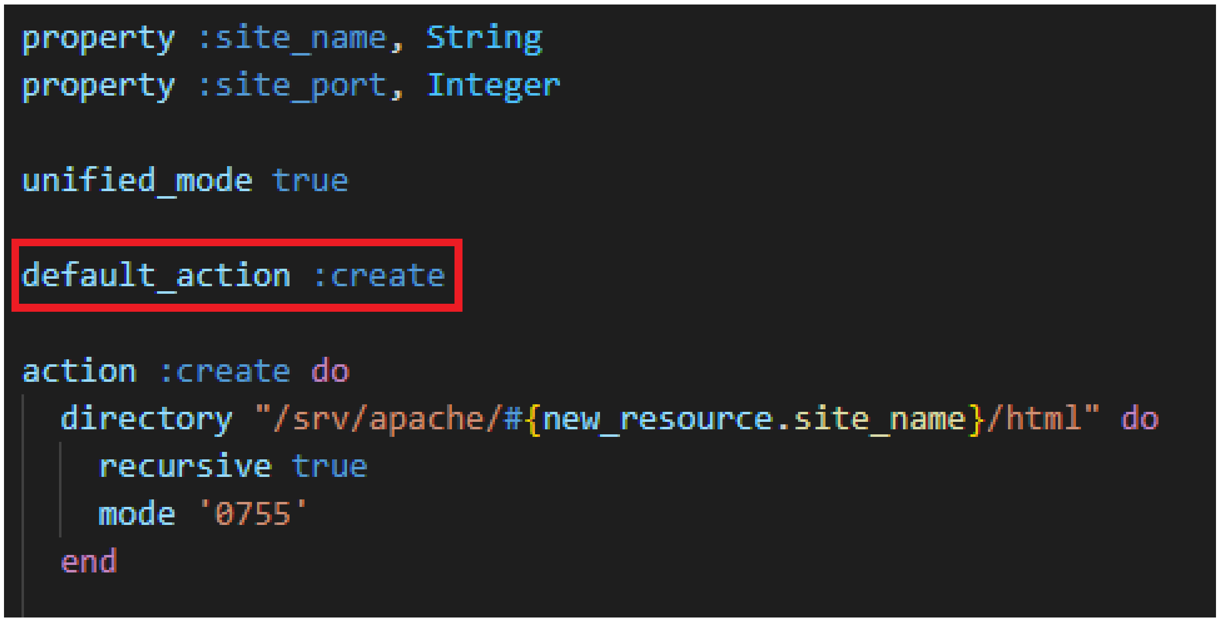
Lab 12, Objectives:

- Define a default action for the custom resource
- Set the resource name as the site_name property
- Set the default value of the site_port property
- Move the resource notifications to the recipe

Setting Default Actions:

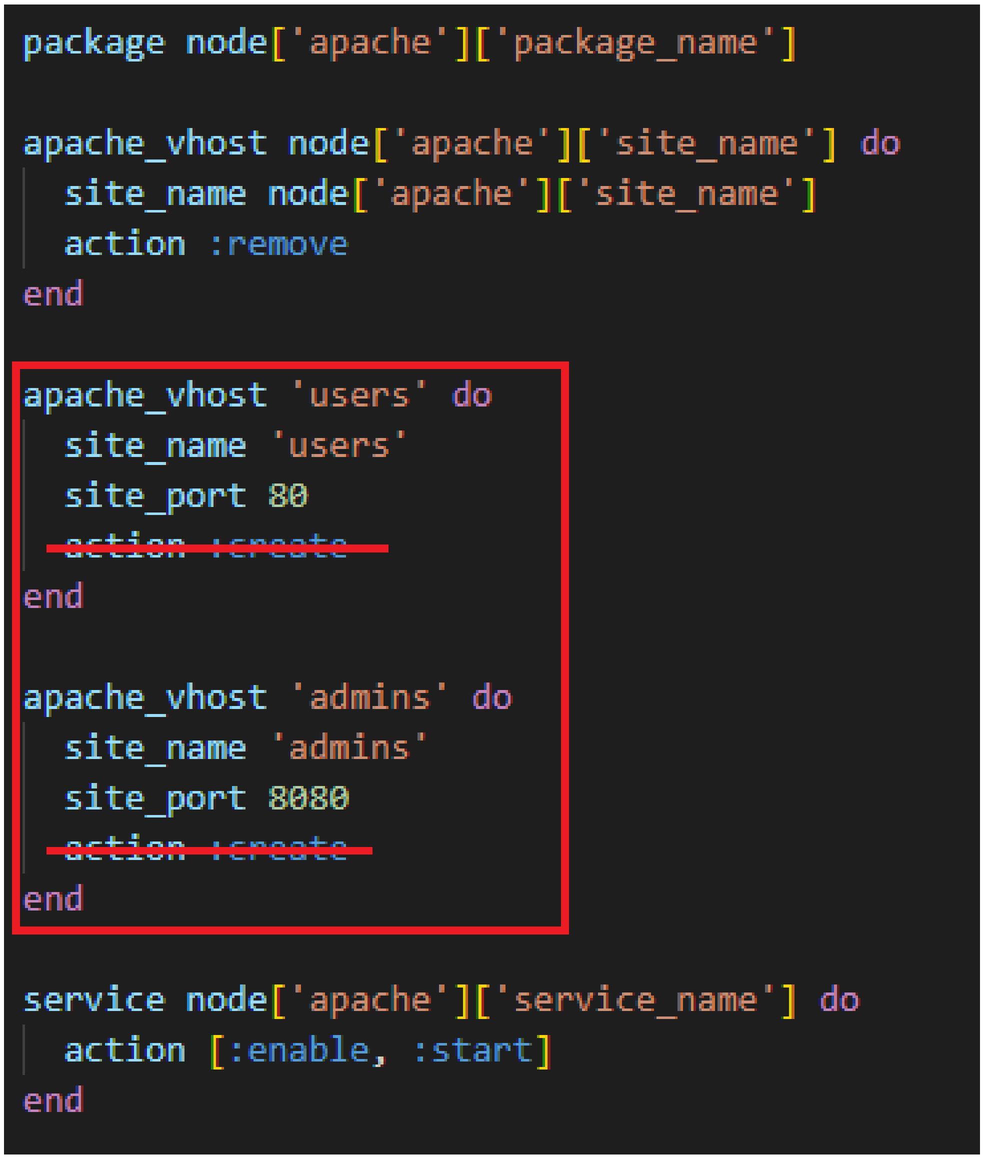
- Resources have a default action. The default action should be the action that is most expected. For most resources it is an additive/restorative action that moves the system into the desired state.
- The first action listed within the custom resource definition is, by default, the default action. **But the default action can be explicitly defined within the resource definition**, and can be any action listed in the custom resource.

1. Update **resources/vhost.rb** to the following:



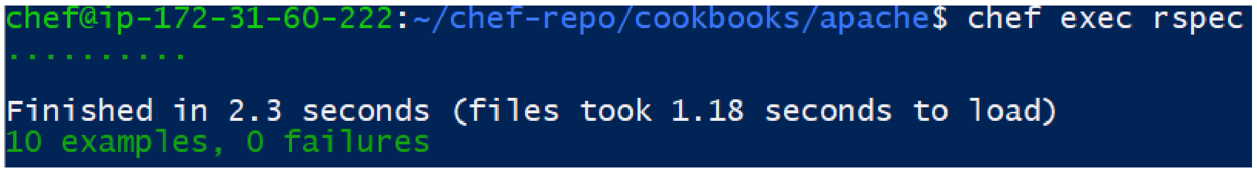
```
property :site_name, String
property :site_port, Integer

unified_mode true

default_action :create

action :create do
  directory "/srv/apache/#{new_resource.site_name}/html" do
    recursive true
    mode '0755'
  end
```

2. Remove the action property from the ‘users’ and ‘admins’ **apache_vhost** resources in the default recipe.

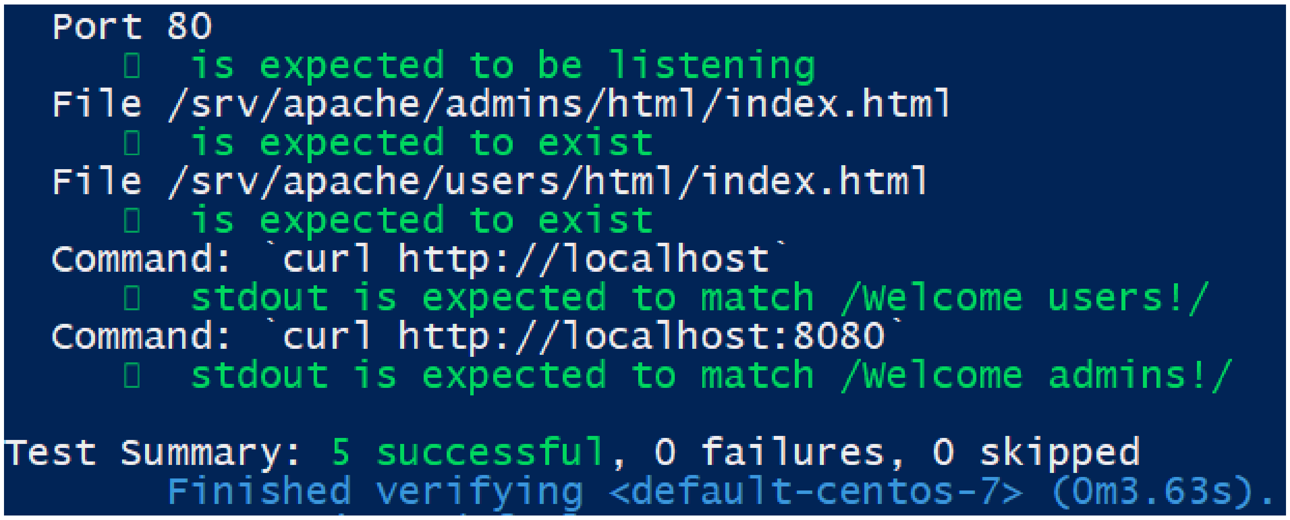


```
package node['apache']['package_name']

apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
end

apache_vhost 'users' do
  site_name 'users'
  site_port 80
  ~~action :create~~
end

apache_vhost 'admins' do
  site_name 'admins'
  site_port 8080
  ~~action :create~~
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

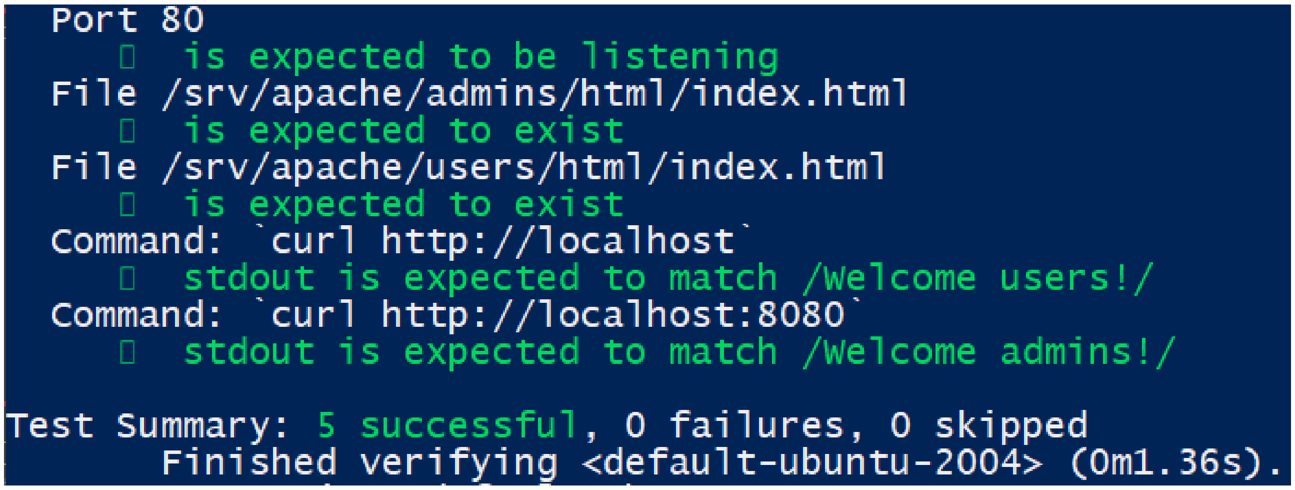
3. Use **ChefSpec** and **Test Kitchen** to verify your changes are working as expected



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..........

Finished in 2.3 seconds (files took 1.18 seconds to load)
10 examples, 0 failures
```



```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.63s).
```

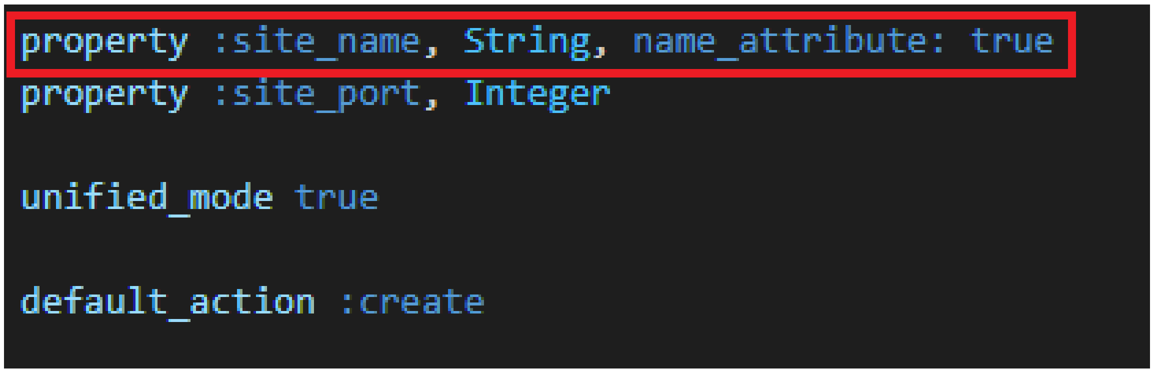


```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.36s).
```

Using the **name_attribute** option:

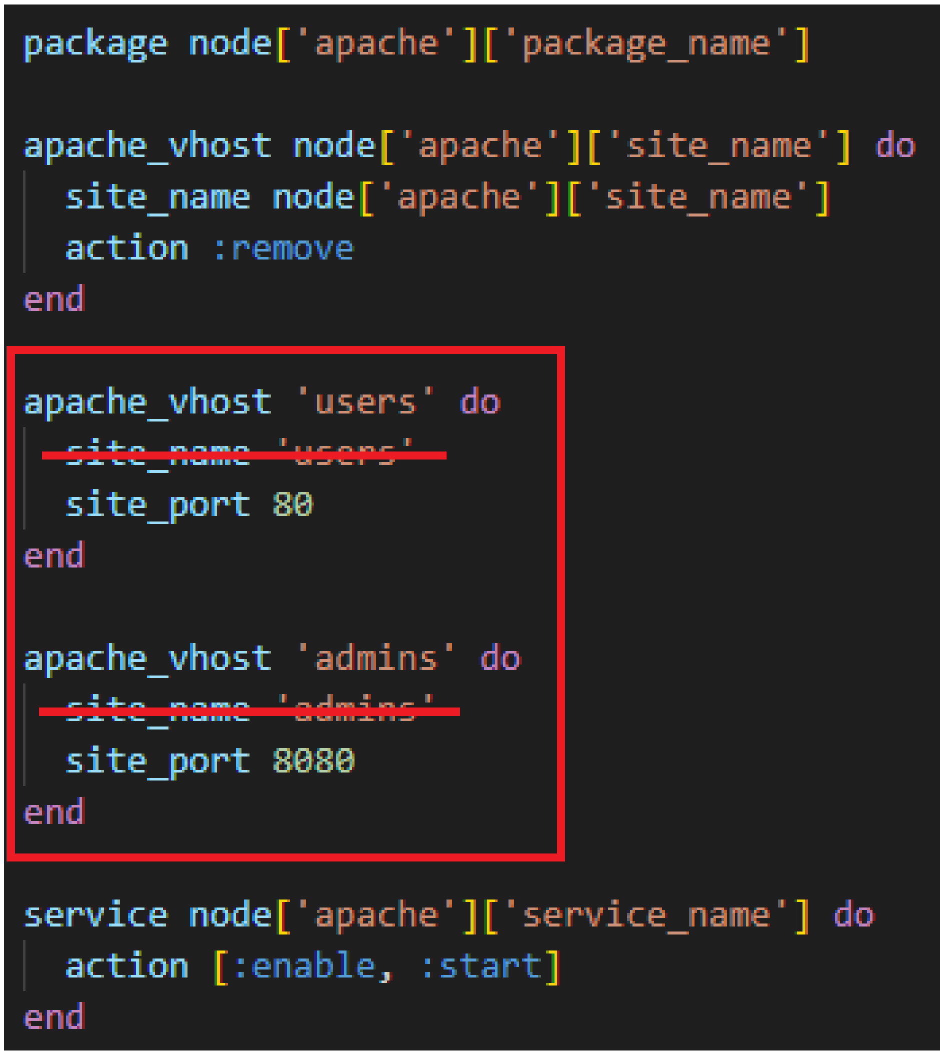
- Single properties (i.e. not an array or hash) may have the **'name_attribute'** option **set to true**
- This property is automatically populated from the name of the resource
- Using the name of the resource removes the need to specify that property within the resource on the recipe page

4. Update the **site_name** property to include the **name_attribute** option in **resources/vhost.rb:**



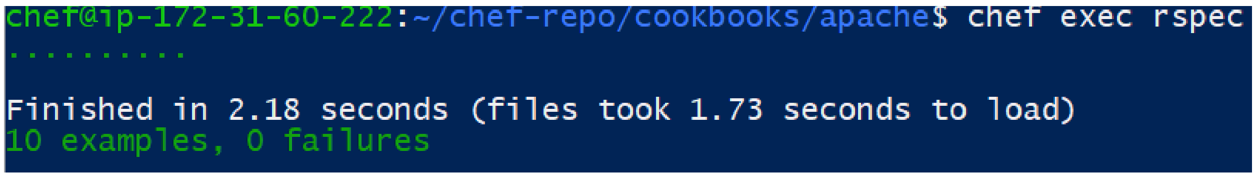
```
property :site_name, String, name_attribute: true
property :site_port, Integer

unified_mode true

default_action :create
```

5. Remove the **site_name** property the ‘users’ and ‘admins’ **apache_vhost** resources in the **recipes/default.rb**:

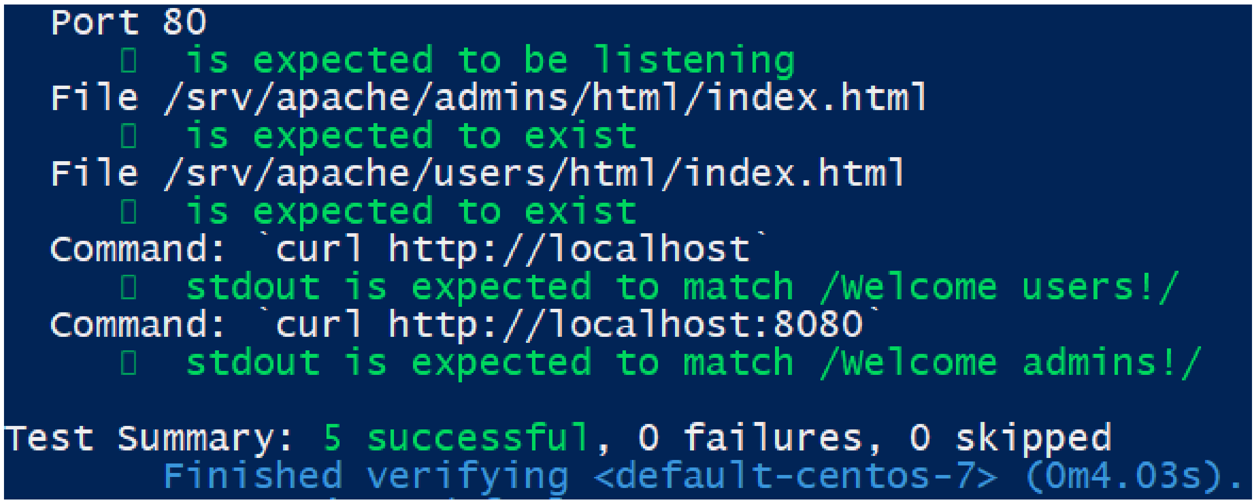


```
package node['apache']['package_name']

apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
end

apache_vhost 'users' do
  ~~site_name 'users'~~
  site_port 80
end

apache_vhost 'admins' do
  ~~site_name 'admins'~~
  site_port 8080
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

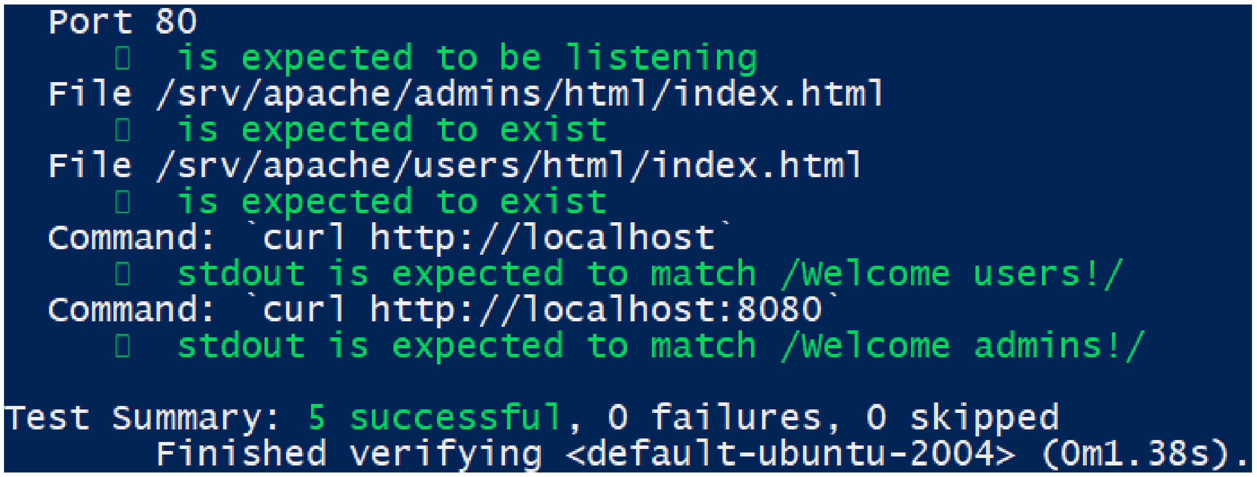
6. Use **ChefSpec** and **Test Kitchen** to verify your changes are working as expected



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..........

Finished in 2.18 seconds (files took 1.73 seconds to load)
10 examples, 0 failures
```



```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m4.03s).
```

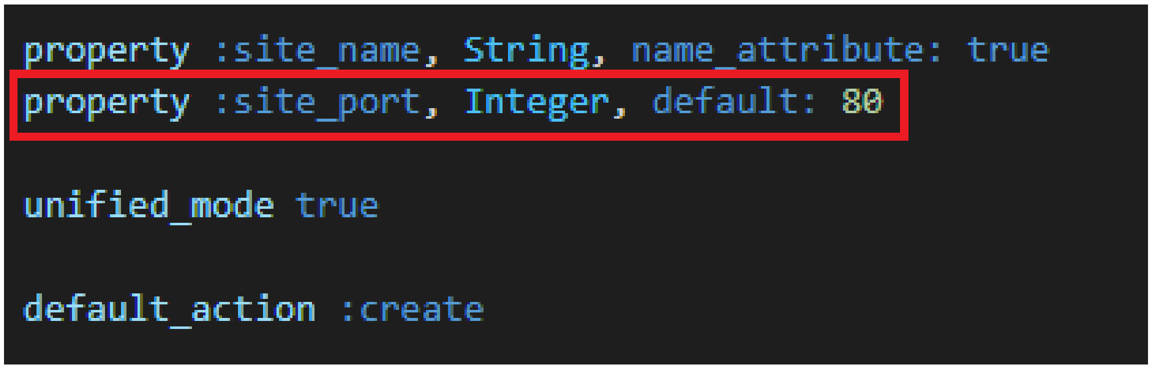


```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.38s).
```

Creating Property default values:

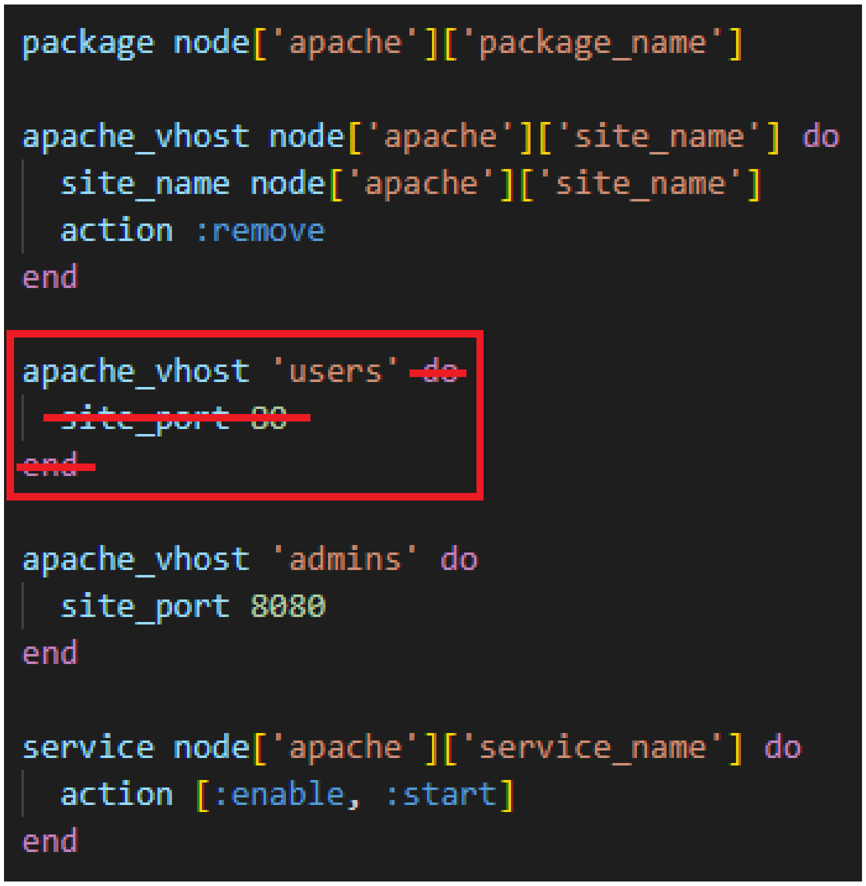
- Setting a default value for a property is optional
- Can be used to create cleaner recipes

7. Update the **site_port** property in **resources/vhost.rb** to include a default value of 80.



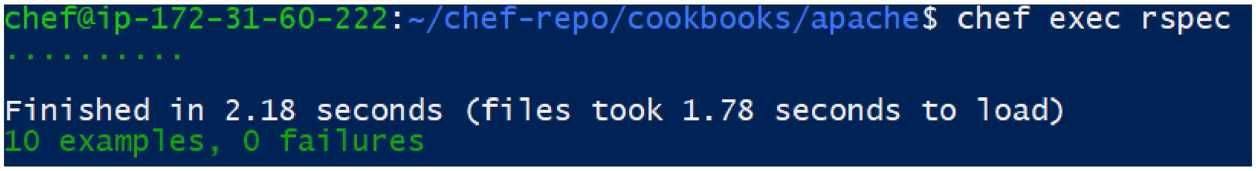
```
property :site_name, String, name_attribute: true
property :site_port, Integer, default: 80

unified_mode true

default_action :create
```

8. Remove the **site_port** property from the 'users' **apache_vhost** resource in **recipes/default.rb**.

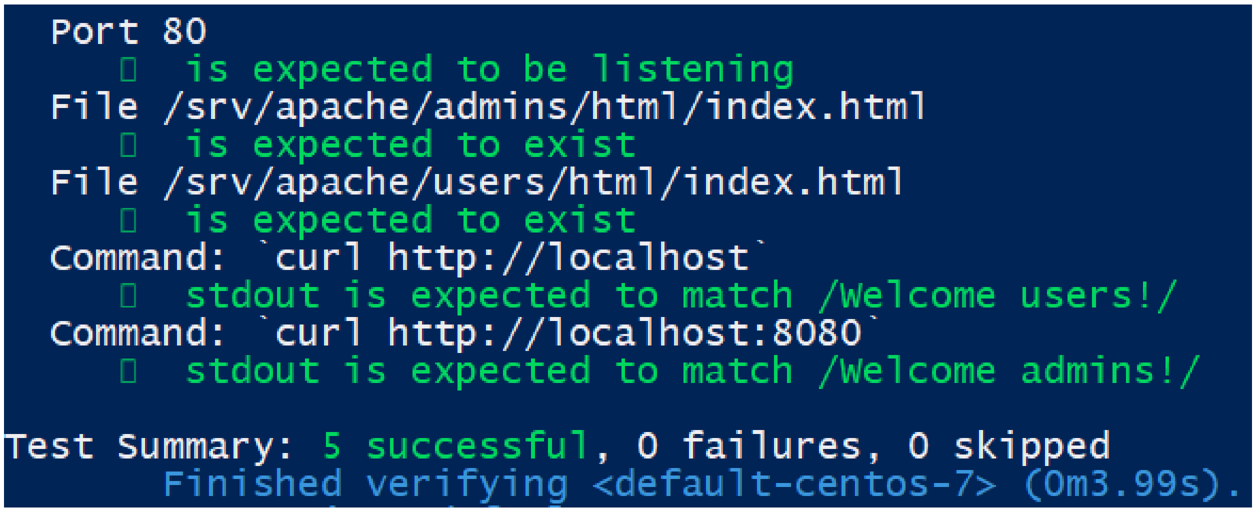


```
package node['apache']['package_name']

apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
end

apache_vhost 'users' ~~do~~
  ~~site_port 80~~
~~end~~

apache_vhost 'admins' do
  site_port 8080
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

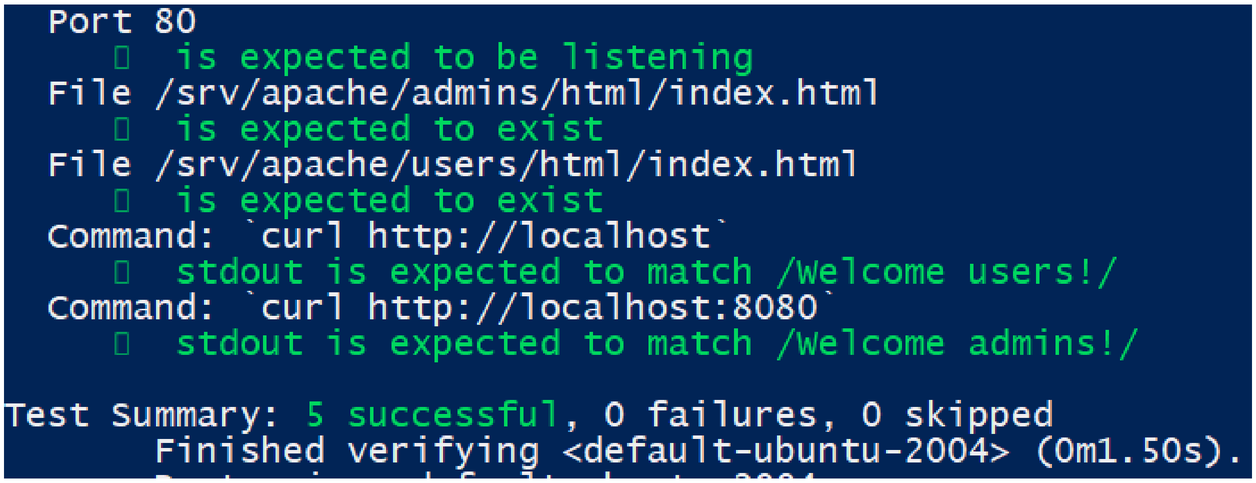
9. Use **ChefSpec** and **Test Kitchen** to verify your changes are working as expected



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..........

Finished in 2.18 seconds (files took 1.78 seconds to load)
10 examples, 0 failures
```



```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m3.99s).
```

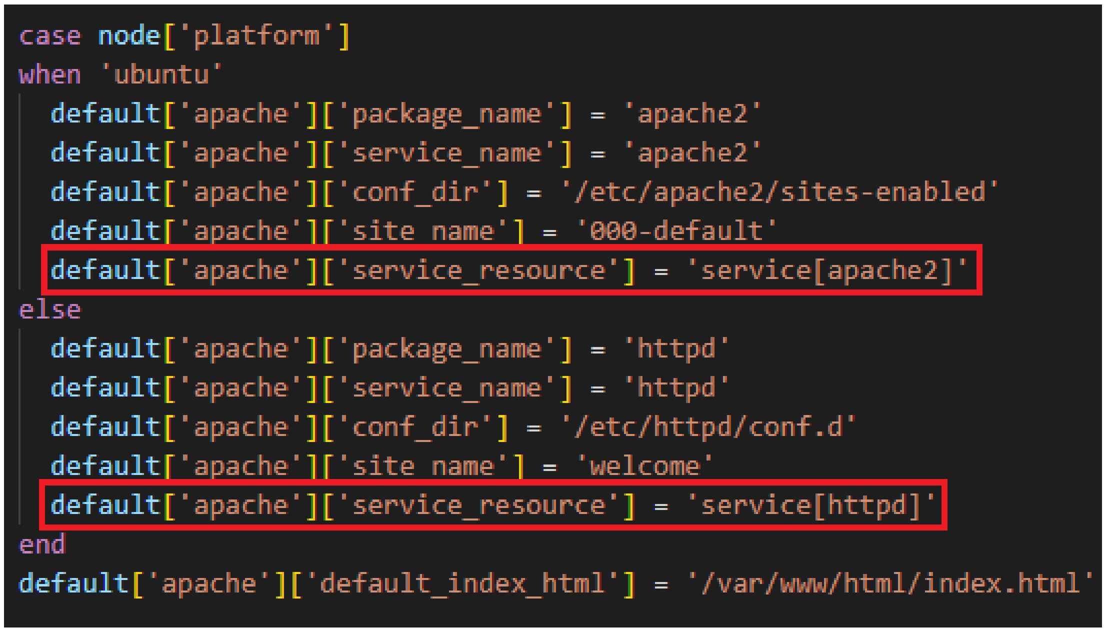


```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.50s).
```

Using notifications with custom resources:

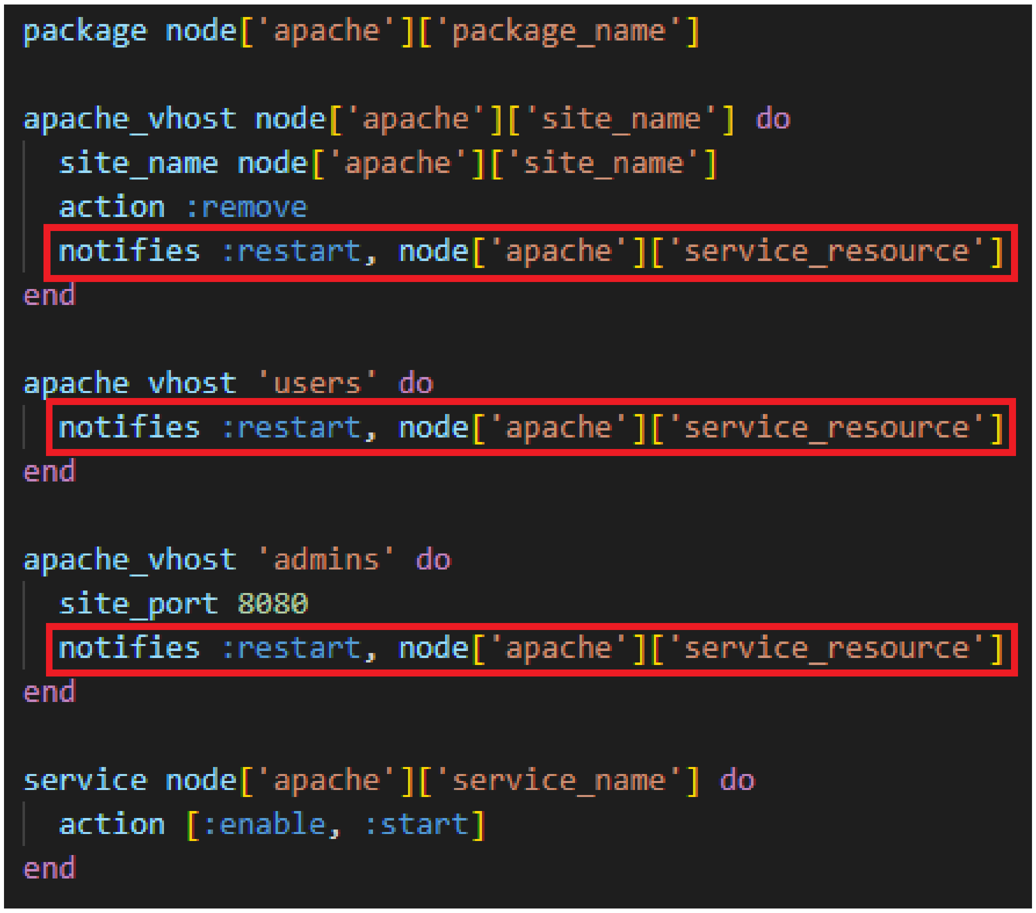
- Notifications should not be placed inside of a custom resource. Defining a notification in a resource under a custom resource creates a fragile relationship.
- To avoid this situation, any notifications needed should be placed in the recipe that uses the custom resource.

10. Create a new attribute for each of the options in the case statement in **attributes/default.rb**



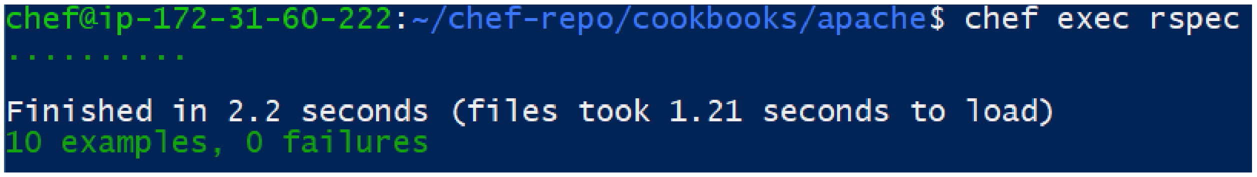
```
case node['platform']
when 'ubuntu'
  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
  default['apache']['conf_dir'] = '/etc/apache2/sites-enabled'
  default['apache']['site_name'] = '000-default'
  default['apache']['service_resource'] = 'service[apache2]'
else
  default['apache']['package_name'] = 'httpd'
  default['apache']['service_name'] = 'httpd'
  default['apache']['conf_dir'] = '/etc/httpd/conf.d'
  default['apache']['site_name'] = 'welcome'
  default['apache']['service_resource'] = 'service[httpd]'
end
default['apache']['default_index_html'] = '/var/www/html/index.html'
```

11. Add notifications to the resources in **recipes/default.rb**

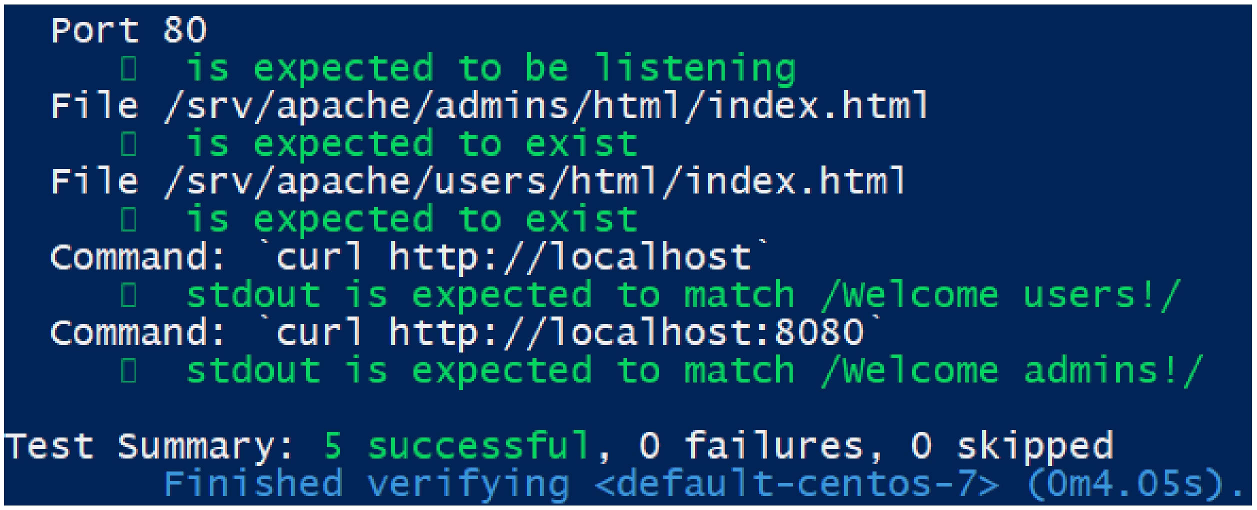


```
package node['apache']['package_name']

apache_vhost node['apache']['site_name'] do
  site_name node['apache']['site_name']
  action :remove
  notifies :restart, node['apache']['service_resource']
end

apache_vhost 'users' do
  notifies :restart, node['apache']['service_resource']
end

apache_vhost 'admins' do
  site_port 8080
  notifies :restart, node['apache']['service_resource']
end

service node['apache']['service_name'] do
  action [:enable, :start]
end
```

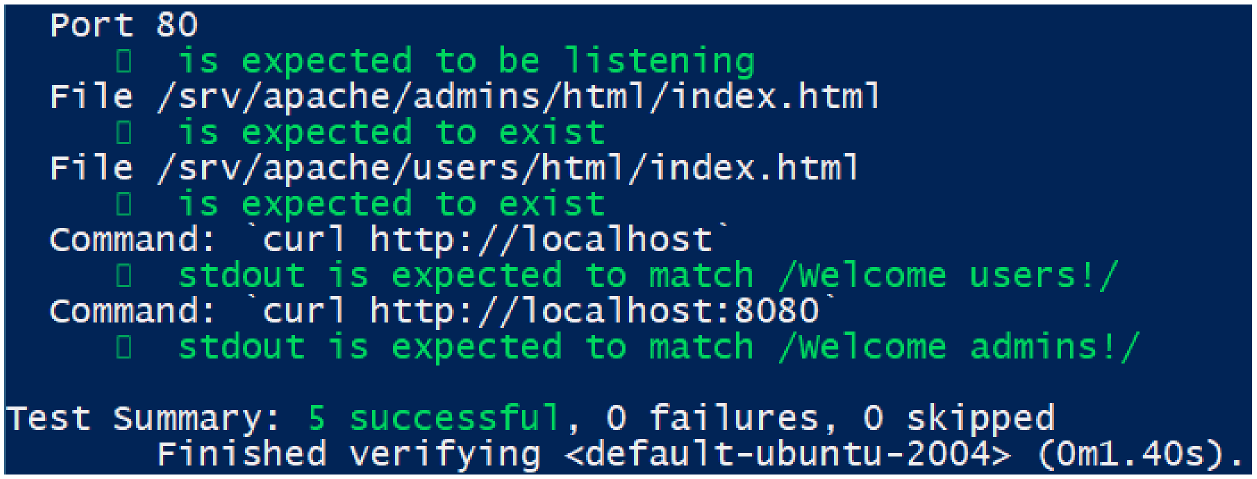
12. Use **ChefSpec** and **Test Kitchen** to verify your changes are working as expected



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..........

Finished in 2.2 seconds (files took 1.21 seconds to load)
10 examples, 0 failures
```



```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m4.05s).
```



```
  Port 80
     □  is expected to be listening
  File /srv/apache/admins/html/index.html
     □  is expected to exist
  File /srv/apache/users/html/index.html
     □  is expected to exist
  Command: `curl http://localhost`
     □  stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     □  stdout is expected to match /Welcome admins!/

Test Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m1.40s).
```

Notify your instructor that you are done with the lab

END OF LAB

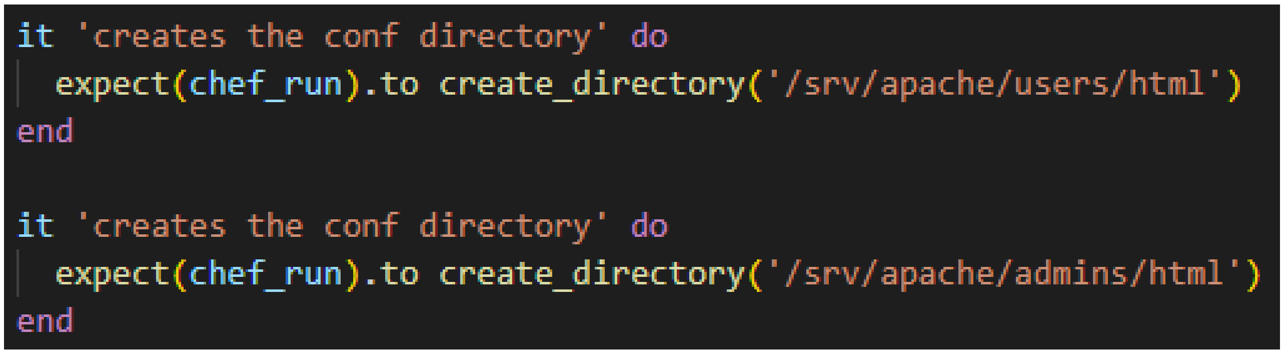
Lab 13, Objectives:

- Complete the ChefSpec test so that all resources in the custom resource are tested.

Unit testing should be as complete as possible to identify exactly where the failure is. The current unit testing suite is not verifying the **directory** or **template** resources in the custom resource.

1. Create two new tests for the Ubuntu platform. One will test that a directory named **/srv/apache/admins/html** exists, the second will test for a directory named **/srv/apache/users/html**

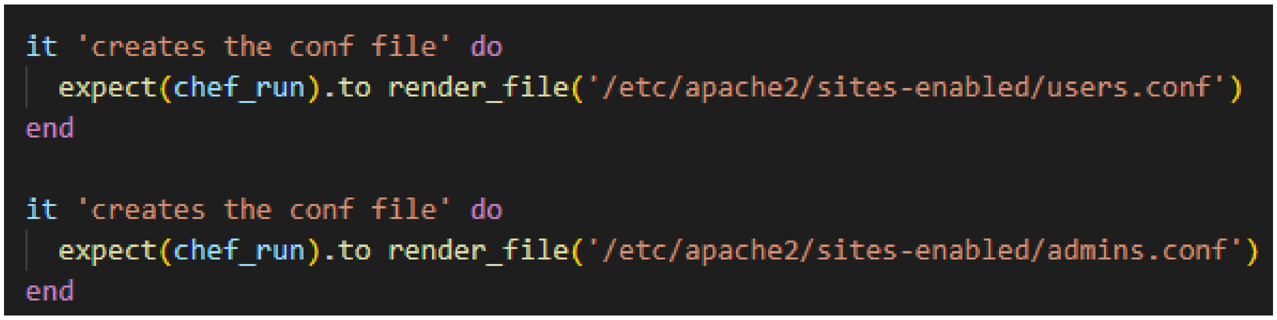
```
it 'creates the conf directory' do
  expect(chef_run).to render_file('/srv/apache/users/html')
end
it 'creates the conf directory' do
  expect(chef_run).to
render_file('/srv/apache/admins/html')
end
```



```
it 'creates the conf directory' do
  expect(chef_run).to create_directory('/srv/apache/users/html')
end

it 'creates the conf directory' do
  expect(chef_run).to create_directory('/srv/apache/admins/html')
end
```

2. Create two new tests for the Ubuntu platform. One will test that a file named **/etc/apache2/sites-enabled/users.conf** exists, the second will test for a file named **/etc/apache2/sites-enabled/admins.conf**

```
it 'creates the conf file' do
  expect(chef_run).to render_file('/etc/apache2/sites-
enabled/users.conf')
end

it 'creates the conf file' do
  expect(chef_run).to render_file('/etc/apache2/sites-
enabled/admins.conf')
end
```

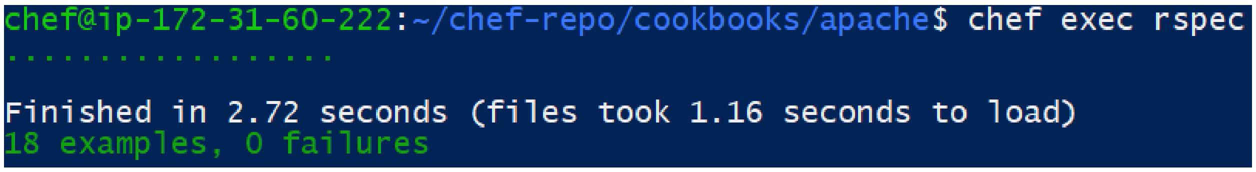


```
it 'creates the conf file' do
  expect(chef_run).to render_file('/etc/apache2/sites-enabled/users.conf')
end

it 'creates the conf file' do
  expect(chef_run).to render_file('/etc/apache2/sites-enabled/admins.conf')
end
```

3. Create four tests for the CentOS platform. Two tests will verify the directories **/srv/apache/users/html** and **/srv/apache/admins/html** exist. The next two tests should verify the files **/etc/httpd/conf.d/users.conf** and **/etc/httpd/conf.d/admins.conf** exist

...Answer at the bottom of the lab…

4. Execute **chef exec rspec** to verify everything is working as expected.



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ chef exec rspec
..................

Finished in 2.72 seconds (files took 1.16 seconds to load)
18 examples, 0 failures
```

Notify your instructor that you are done with the lab

END OF LAB

...to view the solution to step 3, keep on scrolling

. . . .

. . . .

. . . .

. . . .

. . . . . .

. . . . .

. . . .

. . .

. .

.

Keep scrolling...

. . . .

. . . .

. . . .

. . . .

. . . . . .

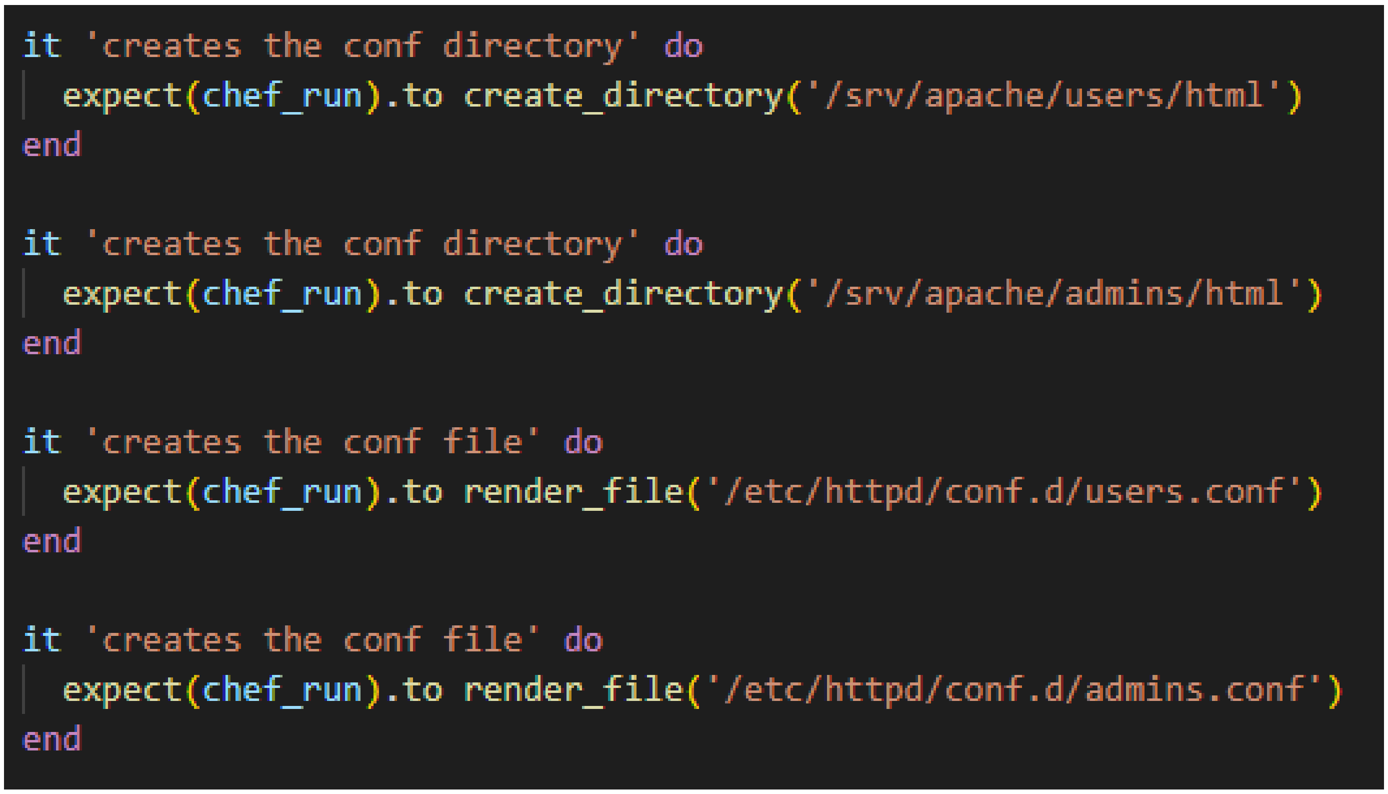
. . . . .

. . . .

. . .

. .

.



```ruby
it 'creates the conf directory' do
  expect(chef_run).to create_directory('/srv/apache/users/html')
end

it 'creates the conf directory' do
  expect(chef_run).to create_directory('/srv/apache/admins/html')
end

it 'creates the conf file' do
  expect(chef_run).to render_file('/etc/httpd/conf.d/users.conf')
end

it 'creates the conf file' do
  expect(chef_run).to render_file('/etc/httpd/conf.d/admins.conf')
end
```

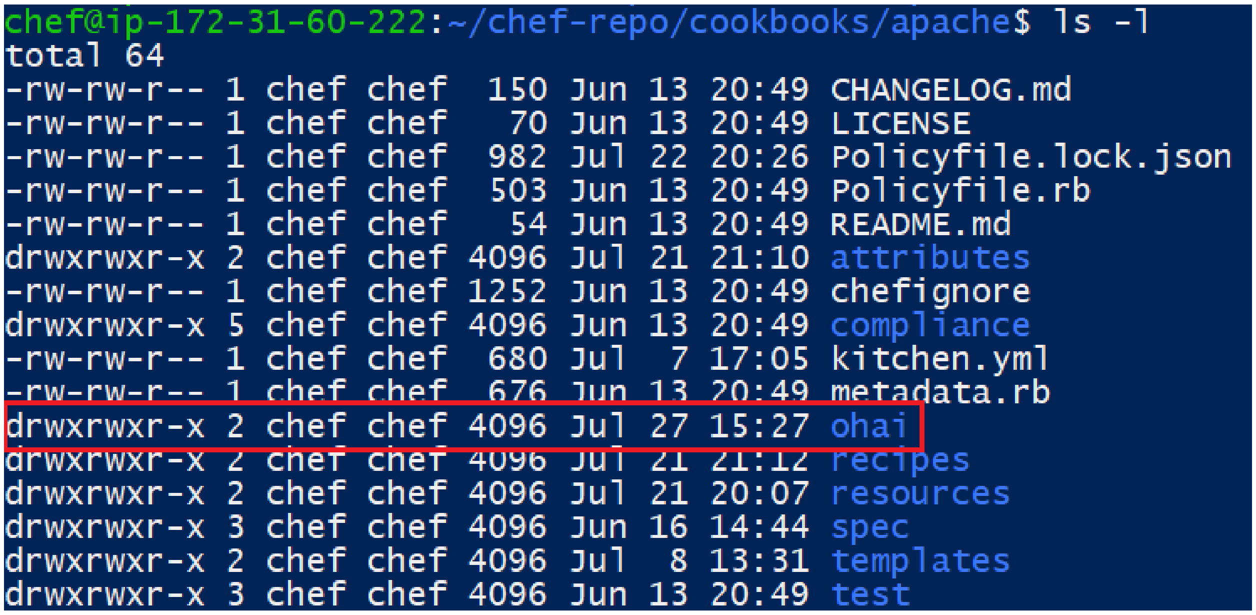
Lab 14, Objectives:

- Create a plugin to get the operating system and internal hostname.

When used, custom ohai plugins are placed in the cookbook that uses them. More information on the different components of a cookbook is available at this link:

https://docs.chef.io/cookbooks/#components

1. The **chef generate** command does not have an option for ohai plugins. Create a new directory with name **ohai** under the apache cookbook:



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ ls -l
total 64
-rw-rw-r-- 1 chef chef  150 Jun 13 20:49 CHANGELOG.md
-rw-rw-r-- 1 chef chef   70 Jun 13 20:49 LICENSE
-rw-rw-r-- 1 chef chef  982 Jul 22 20:26 Policyfile.lock.json
-rw-rw-r-- 1 chef chef  503 Jun 13 20:49 Policyfile.rb
-rw-rw-r-- 1 chef chef   54 Jun 13 20:49 README.md
drwxrwxr-x 2 chef chef 4096 Jul 21 21:10 attributes
-rw-rw-r-- 1 chef chef 1252 Jun 13 20:49 chefignore
drwxrwxr-x 5 chef chef 4096 Jun 13 20:49 compliance
-rw-rw-r-- 1 chef chef  680 Jul  7 17:05 kitchen.yml
-rw-rw-r-- 1 chef chef  676 Jun 13 20:49 metadata.rb
drwxrwxr-x 2 chef chef 4096 Jul 27 15:27 ohai
drwxrwxr-x 2 chef chef 4096 Jul 21 21:12 recipes
drwxrwxr-x 2 chef chef 4096 Jul 21 20:07 resources
drwxrwxr-x 3 chef chef 4096 Jun 16 14:44 spec
drwxrwxr-x 2 chef chef 4096 Jul  8 13:31 templates
drwxrwxr-x 3 chef chef 4096 Jun 13 20:49 test
```

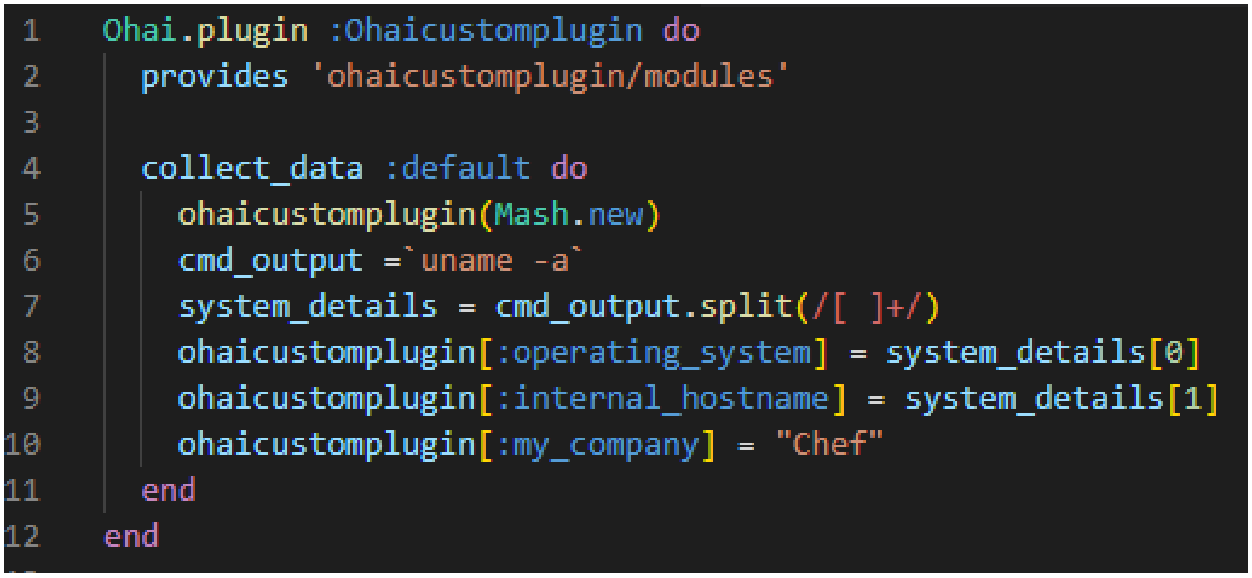
2. Create a file to write a plugin code.

   $ touch ohai/ohai_custom_plugin.rb

3. Add the code below to the **ohai/ohai_custom_plugin.rb**:

```
Ohai.plugin :Ohaicustomplugin do
  provides 'ohaicustomplugin/modules'

  collect_data :default do
    ohaicustomplugin(Mash.new)
    cmd_output =`uname -a`
    system_details = cmd_output.split(/[ ]+/)
    ohaicustomplugin[:operating_system] = system_details[0]
    ohaicustomplugin[:internal_hostname] =
system_details[1]
    ohaicustomplugin[:my_company] = "Chef"
  end
end
```

Note: `uname -a` has backticks around it, not single quotes



```
1   Ohai.plugin :Ohaicustomplugin do
2     provides 'ohaicustomplugin/modules'
3
4     collect_data :default do
5       ohaicustomplugin(Mash.new)
6       cmd_output =`uname -a`
7       system_details = cmd_output.split(/[ ]+/)
8       ohaicustomplugin[:operating_system] = system_details[0]
9       ohaicustomplugin[:internal_hostname] = system_details[1]
10      ohaicustomplugin[:my_company] = "Chef"
11    end
12  end
```

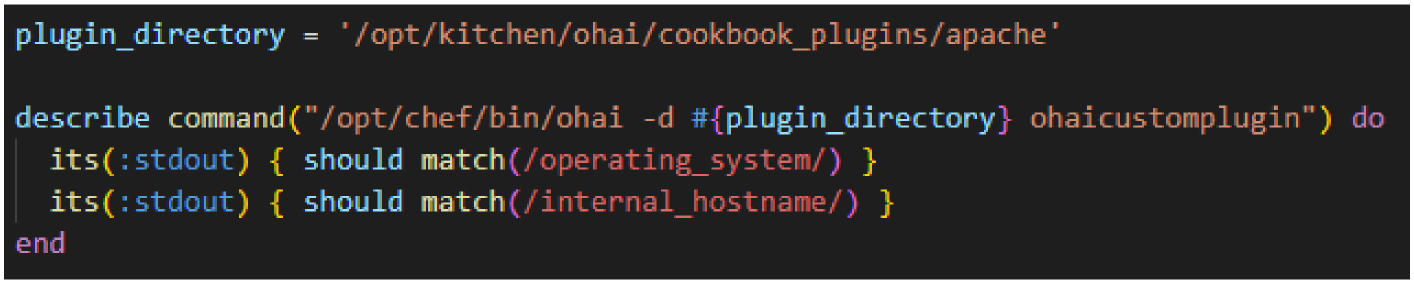
Before moving on, examine the code line by line:

1. Line 1 calls the **Ohai.plugin** class with a single parameter that is the name of the plugin. The plugin name must begin with an upper-case letter. The plugin name is a required field
    a. The name of this plugin in **Ohaicustomplugin**

2. Line 2 defines what the plugin provides. This is a comma-separated list of one (or more) attributes that are defined by this plugin. **provides** is a required field
3. Line 4 creates the **collect_data** block. This is a block of Ruby code that is called by Ohai when it runs. One (or more) **collect_data** blocks can be defined in a plugin, but only a single **collect_data** block is ever run.
    a. **collect_data** blocks can be created for different platforms
    b. This **collect_data** block is named **:default**, so it will run when another block matching the platform is not found.
4. Inside the **collect_data** block:
    a. Line 5 creates a new Ruby mash named **ohaicustomplugin**
    b. Line 6 assigns the output of `uname -a` to a variable
    c. Line 7 creates an array by splitting the variable on the spaces
    d. Lines 8 and 9 take elements from the previous array and adds it to the Ruby mash.
    e. Line 10 adds a custom key:value pair to the Ruby mash.

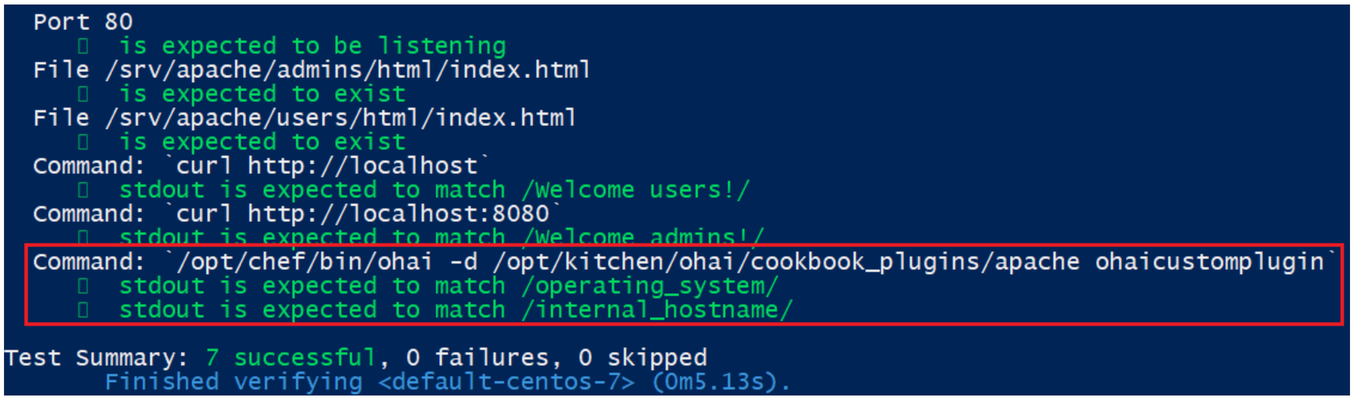
4. Next, add the following integration test to **test/integration/default/default_test.rb**:

```
plugin_directory =
'/opt/kitchen/ohai/cookbook_plugins/apache'


describe command("/opt/chef/bin/ohai -d #{plugin_directory}
ohaicustomplugin") do

  its(:stdout) { should match(/operating_system/) }

  its(:stdout) { should match(/internal_hostname/) }

end
```

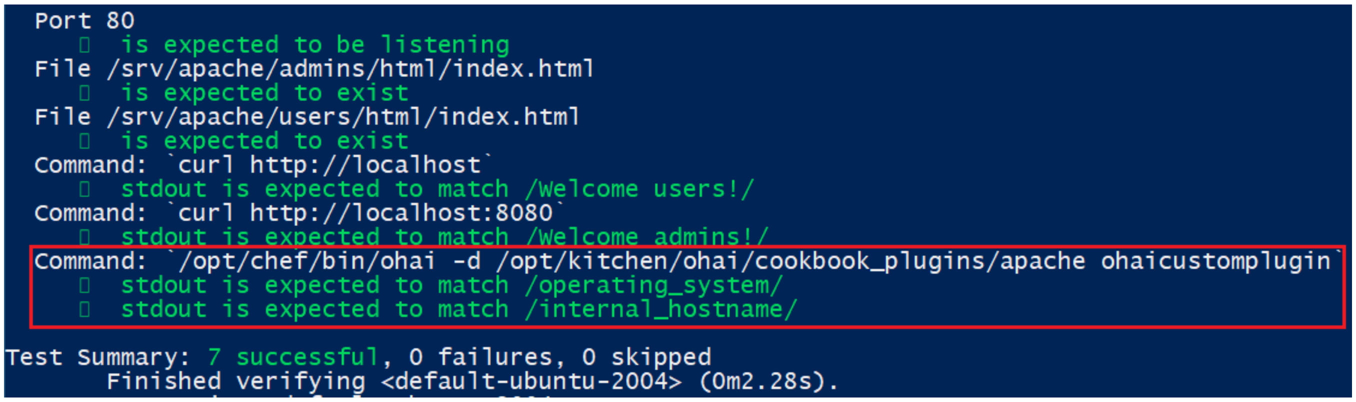


```
plugin_directory = '/opt/kitchen/ohai/cookbook_plugins/apache'

describe command("/opt/chef/bin/ohai -d #{plugin_directory} ohaicustomplugin") do
  its(:stdout) { should match(/operating_system/) }
  its(:stdout) { should match(/internal_hostname/) }
end
```

5. Use **kitchen converge** and **kitchen verify** to ensure the plugin is working.

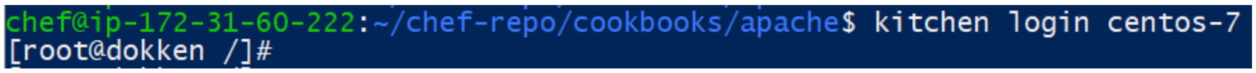


```
  Port 80
     is expected to be listening
  File /srv/apache/admins/html/index.html
     is expected to exist
  File /srv/apache/users/html/index.html
     is expected to exist
  Command: `curl http://localhost`
     stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     stdout is expected to match /Welcome admins!/
  Command: `/opt/chef/bin/ohai -d /opt/kitchen/ohai/cookbook_plugins/apache ohaicustomplugin`
     stdout is expected to match /operating_system/
     stdout is expected to match /internal_hostname/

Test Summary: 7 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-7> (0m5.13s).
```



```
  Port 80
     is expected to be listening
  File /srv/apache/admins/html/index.html
     is expected to exist
  File /srv/apache/users/html/index.html
     is expected to exist
  Command: `curl http://localhost`
     stdout is expected to match /Welcome users!/
  Command: `curl http://localhost:8080`
     stdout is expected to match /Welcome admins!/
  Command: `/opt/chef/bin/ohai -d /opt/kitchen/ohai/cookbook_plugins/apache ohaicustomplugin`
     stdout is expected to match /operating_system/
     stdout is expected to match /internal_hostname/

Test Summary: 7 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-2004> (0m2.28s).
```

6. Log in to the CentOS virtual machine using **kitchen login centos-7**.



```
chef@ip-172-31-60-222:~/chef-repo/cookbooks/apache$ kitchen login centos-7
[root@dokken /]#
```

7. The command below will verify that the attributes created by the custom ohai plugin exist. A breakdown of the command is provided below.

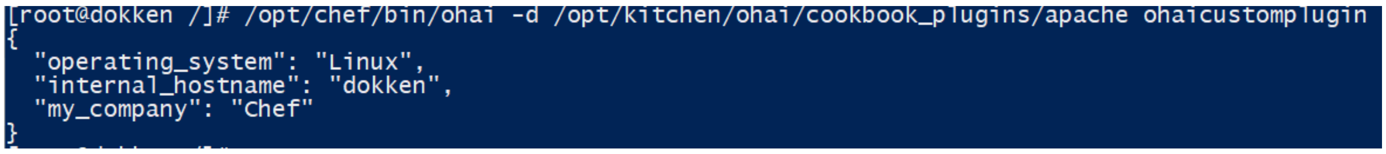
/opt/chef/bin/ohai -d /opt/kitchen/ohai/cookbook_plugins/apache/ ohaicustomplugin

Command breakdown:

/opt/chef/bin/ohai → path to the ohai command

-d /opt/kitchen/ohai/cookbook_plugins/apache/ → pointer to the custom plugin path (versus the location of the core plugins that come with Ohai)

ohaicustomplugin → name of the plugin to display. Without this, ALL ohai data would be displayed. Adding this specifies to ONLY display this one plugin.



```
[root@dokken /]# /opt/chef/bin/ohai -d /opt/kitchen/ohai/cookbook_plugins/apache ohaicustomplugin
{
  "operating_system": "Linux",
  "internal_hostname": "dokken",
  "my_company": "Chef"
}
```

Notify your instructor that you are done with the lab

END OF LAB