



CHEF™

INTERMEDIATE

Introduce Yourselves

Name

Current job role

Previous job roles / Background

Experience with Chef

Favorite Text Editor

Expectations

You will leave this class with the ability to extend the components of Cookbooks.

You bring with you your own domain expertise and problems. Chef is a framework for solving those problems. Our job is to teach you how to express solutions to your problems with Chef.

Expectations

Ask Me Anything: It is important that we answer your questions and set you on the path to be able to find more answers.

Break It: If everything works the first time go back and make some changes. Explore! Discovering the boundaries will help you when you continue on your journey.

Group Exercises, Labs, and Discussion

This course is designed to be hands on. You will run lots of commands, write lots of code, and express your understanding.

- **Group Exercises:** All participants and the instructor will work through the content together. The instructor will often lead the way and explain things as we proceed.
- **Lab:** You will be asked to perform the task on your own or in groups.
- **Discussion:** As a group we will talk about the concepts introduced and the work that we have completed.

Day 1

Introduction

Why Write Tests? Why is that Hard?

Writing a Test First

Refactoring Cookbooks with Tests

Faster Feedback with Unit Testing

Testing Resources in Recipes

Refactoring to Attributes

Refactoring to Multiple Platforms

Day 2

Approaches to Extending Resources

Why Use Custom Resources

Creating a Custom Resource

Refining a Custom Resource

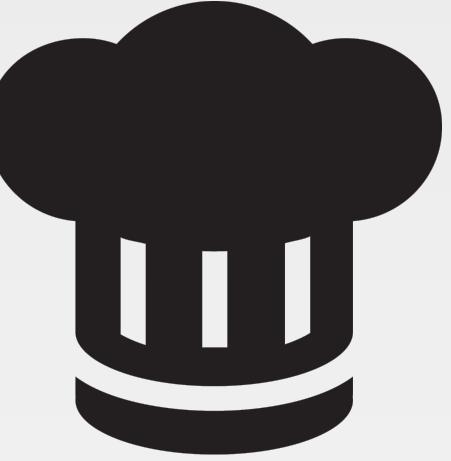
Ohai

Ohai Plugins

Creating an Ohai Plugin

Tuning Ohai

EXERCISE



Pre-built Workstation

We will provide for you a workstation with all the tools installed.

Objective:

- Login to the Remote Workstation

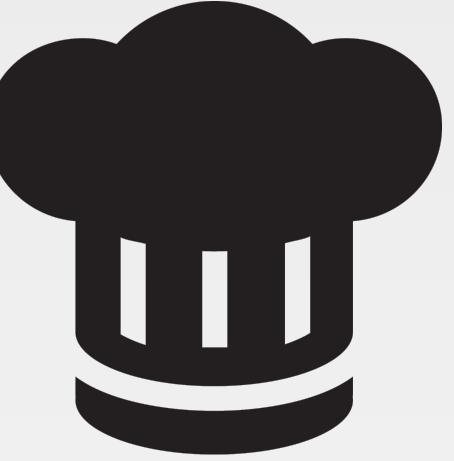
Login to the Workstation



```
> ssh IPADDRESS -l USERNAME
```

```
The authenticity of host '54.209.164.144 (54.209.164.144)' can't  
be established. RSA key fingerprint is  
SHA256:tKoTsPbn6ER9BLThZqntXTxIYem3zV/iTQWvhLrBIBQ. Are you sure  
you want to continue connecting (yes/no)? yes  
chef@54.209.164.144's password: PASSWORD  
chef@ip-172-31-15-97 ~]$
```

EXERCISE



Pre-built Workstation

We will provide for you a workstation with all the tools installed.

Objective:

- ✓ Login to the Remote Workstation

DISCUSSION

Discussion

What topics are you most interested in learning?

What topics are missing that you want to learn about?

DISCUSSION

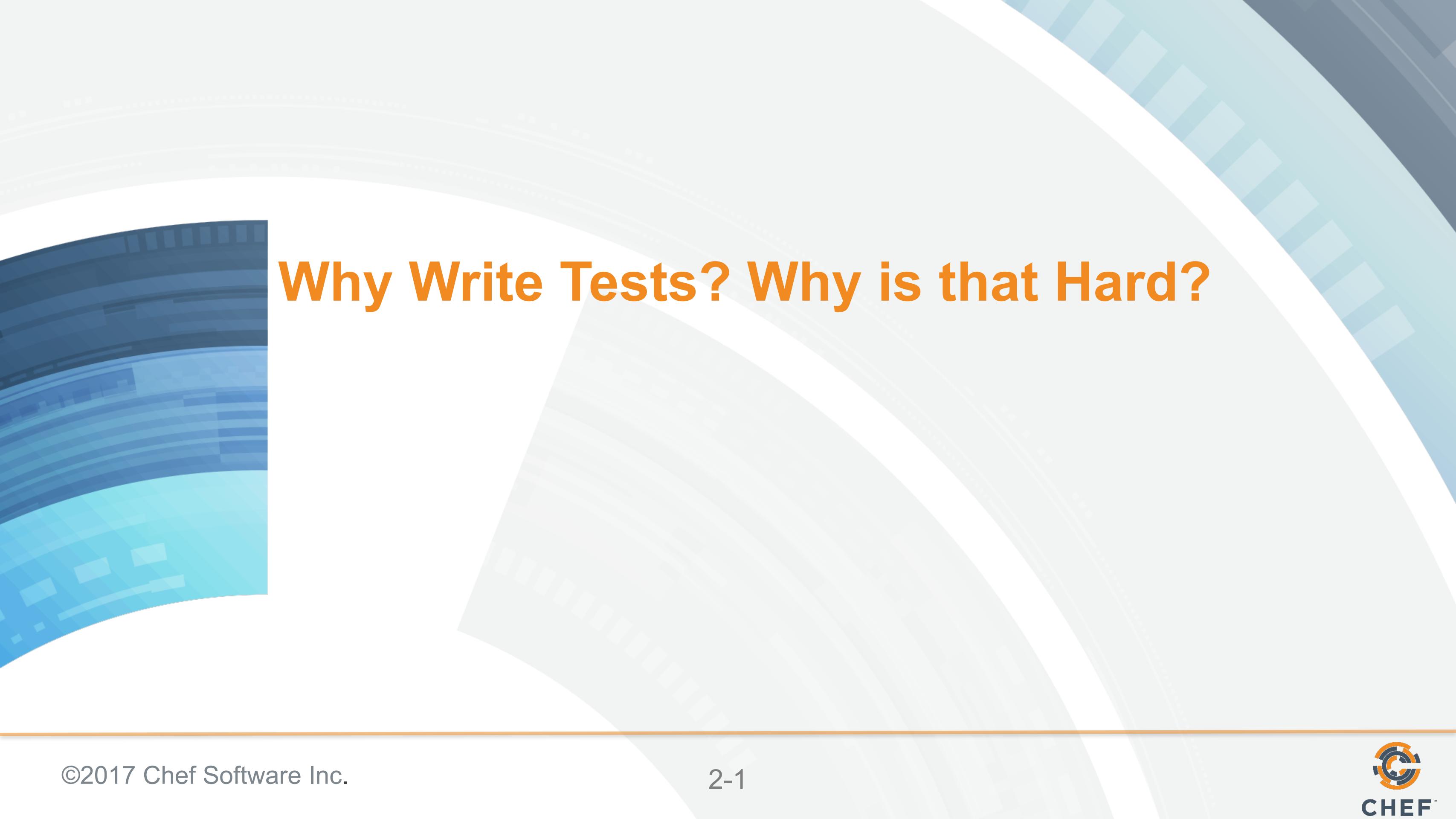


Q&A

What questions can we answer for you?

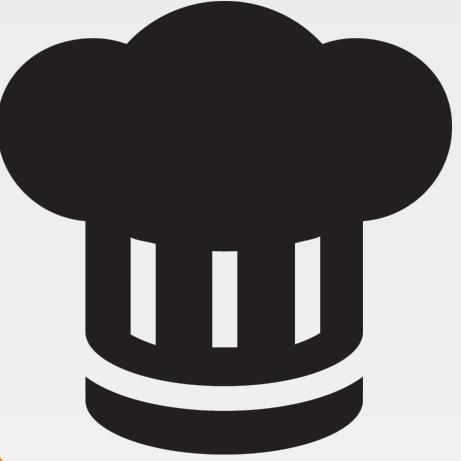


CHEF™



Why Write Tests? Why is that Hard?

EXERCISE



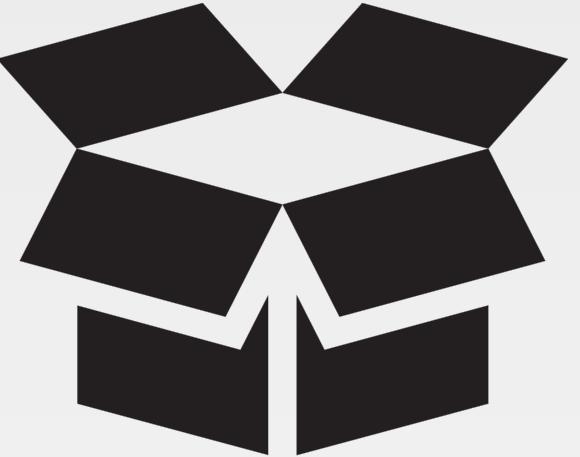
Why Write Tests? Why is that Hard?

Should I write a test? Perhaps the answer to that question lies in: why write tests?

Objective:

- Discussion about Writing Tests
- Discussion about Why Writing Tests is Hard

CONCEPT

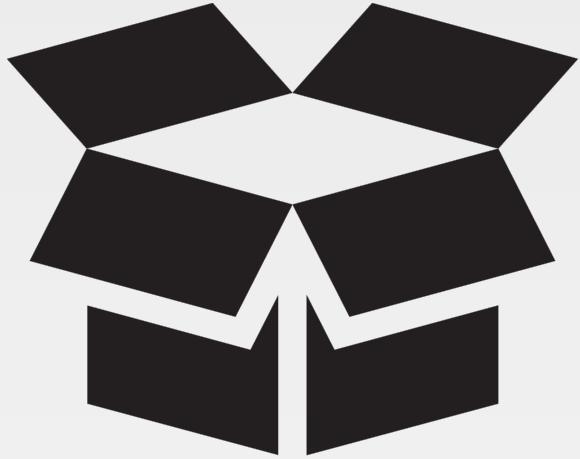


Current Cookbook Development Workflow



1. Write some cookbook code
2. Perform ad-hoc verification
3. Upload the cookbook to the Chef Server

CONCEPT

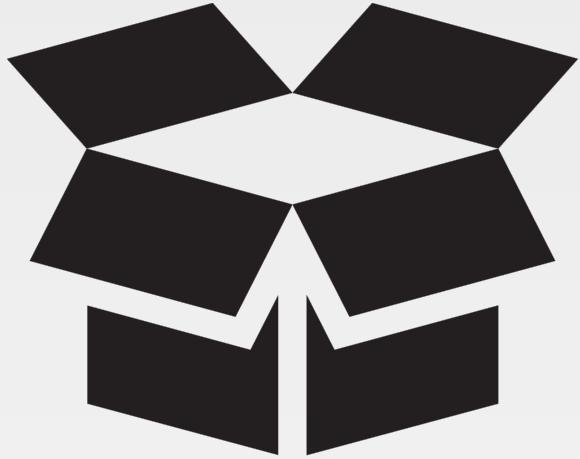


Current Cookbook Development Workflow



1. Chef-client run retrieves the updated cookbook
2. Perform ad-hoc verification
3. Promote the cookbook to the next environment

CONCEPT



Current Cookbook Development Workflow



1. Chef-client run retrieves the updated cookbook
2. Perform ad-hoc verification
3. Establish monitoring for deployed services

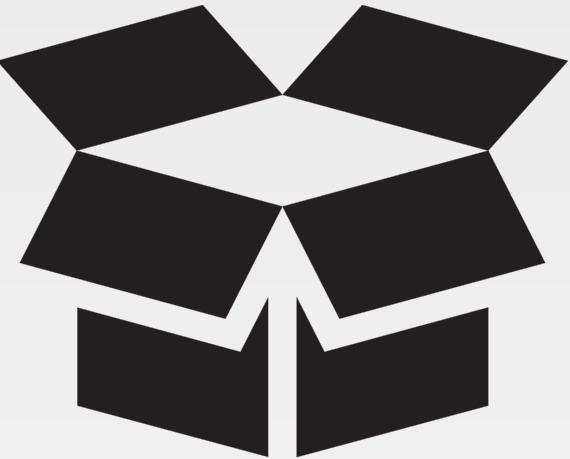
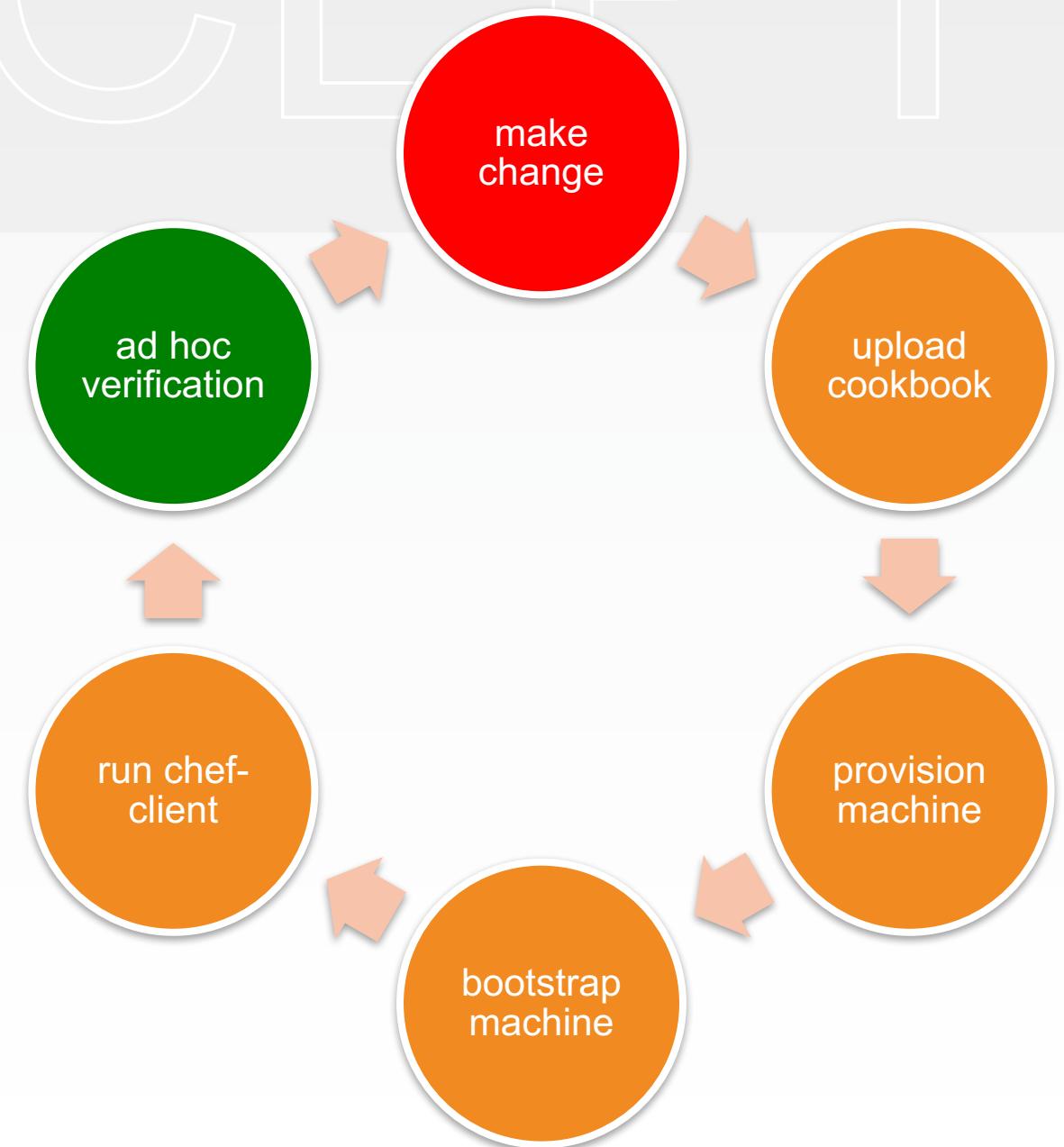
PROBLEM



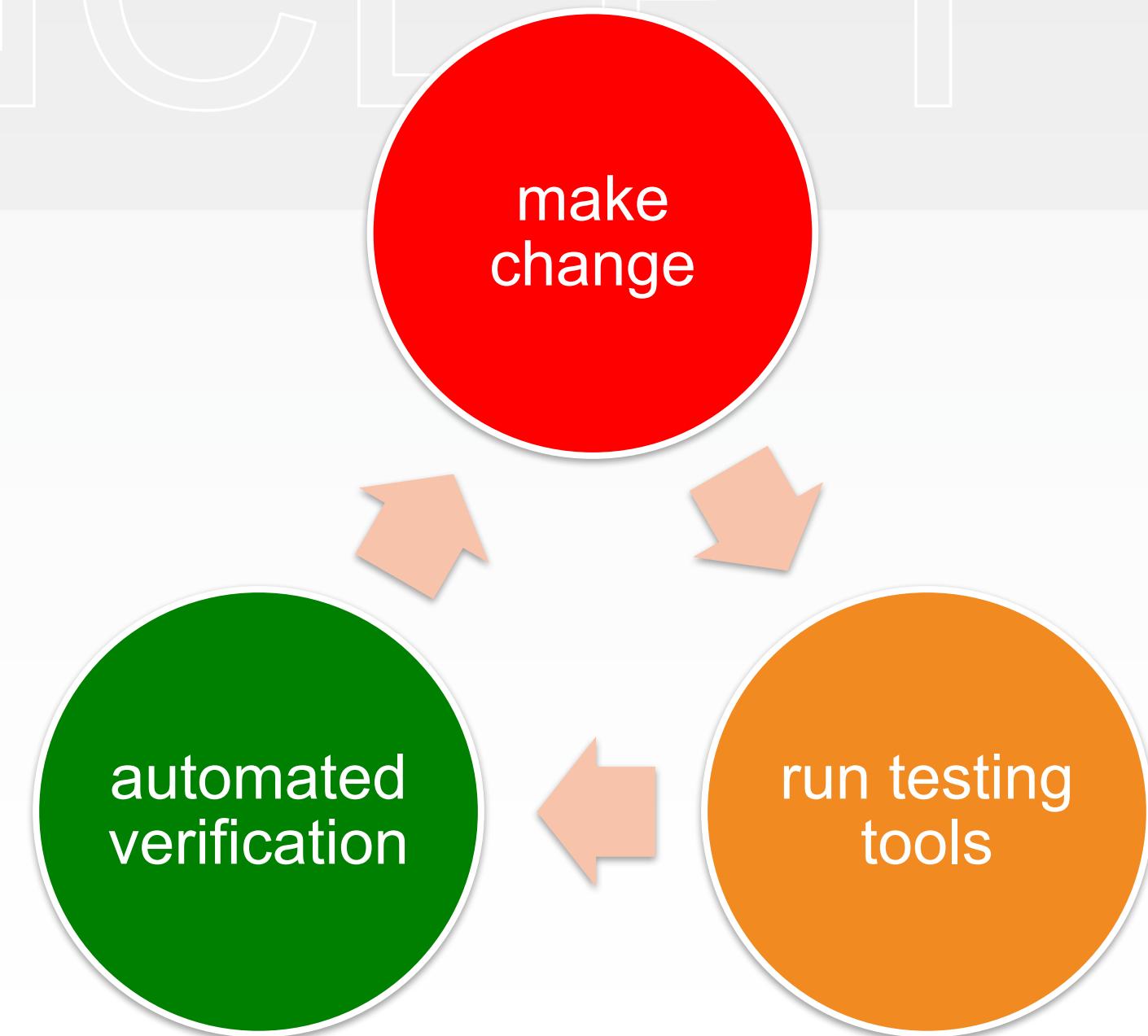
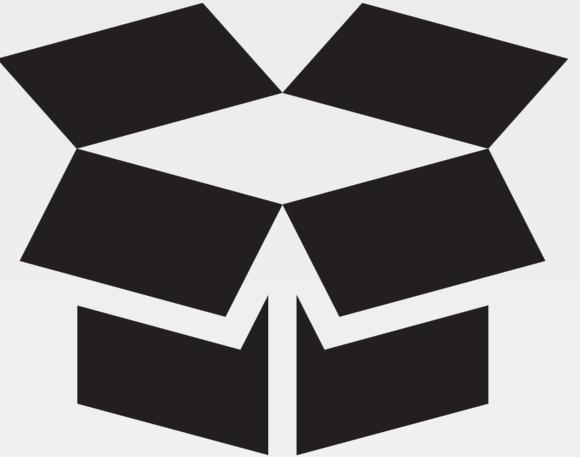
Risk

Every change to our cookbooks introduce risk. Validating every change would take too long in this system. To alleviate that we often batch these changes up. Batching up the changes make it harder to discover when we introduced an error.

CONCEPT



CONCEPT



DISCUSSION



Discussion

What are reasons you see for writing tests?

What are reasons you see for not writing tests?

EXERCISE



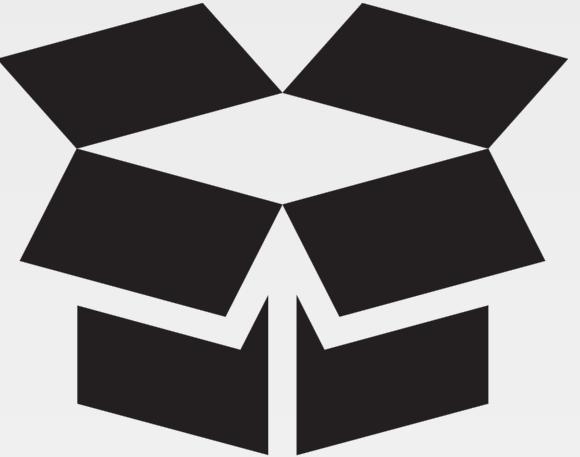
Why Write Tests? Why is that Hard?

To test or not to tests. It's no longer the question. Now I need to think: what makes writing tests challenging?

Objective:

- Discussion about Writing Tests
- Discussion about Why Writing Tests is Hard

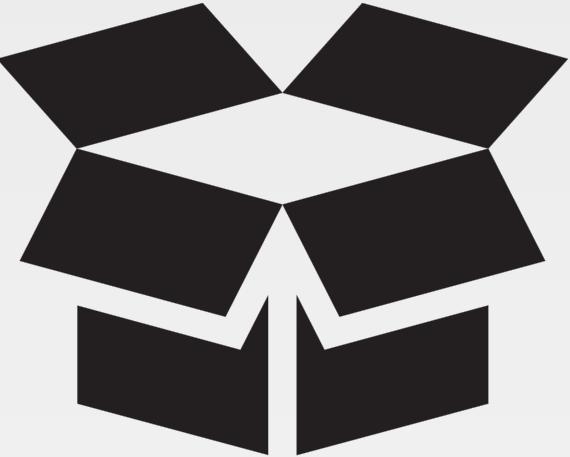
CONCEPT



Another Language

Learning to write tests require you to learn a whole new language that you must understand grammatically.

CONCEPT



Another Workflow

Executing tests requires learning new tools, commands, flags and configurations with entirely new mechanisms that provide you feedback.

DISCUSSION

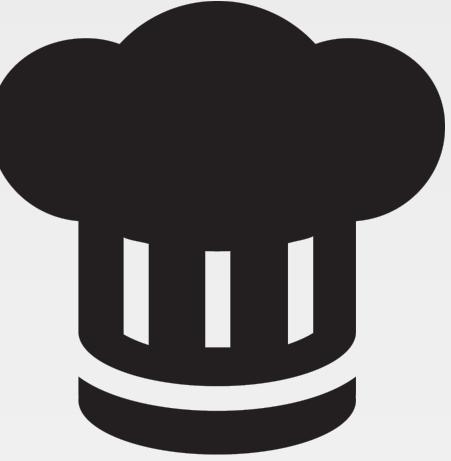


Discussion

What are reasons you see that make testing hard?

What are some of the ways in which you have made it less hard?

EXERCISE



Why Write Tests? Why is that Hard?

I may or may not be convinced. The important thing is I understand what others around me think ... now I have more information to make up my mind.

Objective:

- ✓ Discussion about Writing Tests
- ✓ Discussion about Why Writing Tests is Hard

DISCUSSION



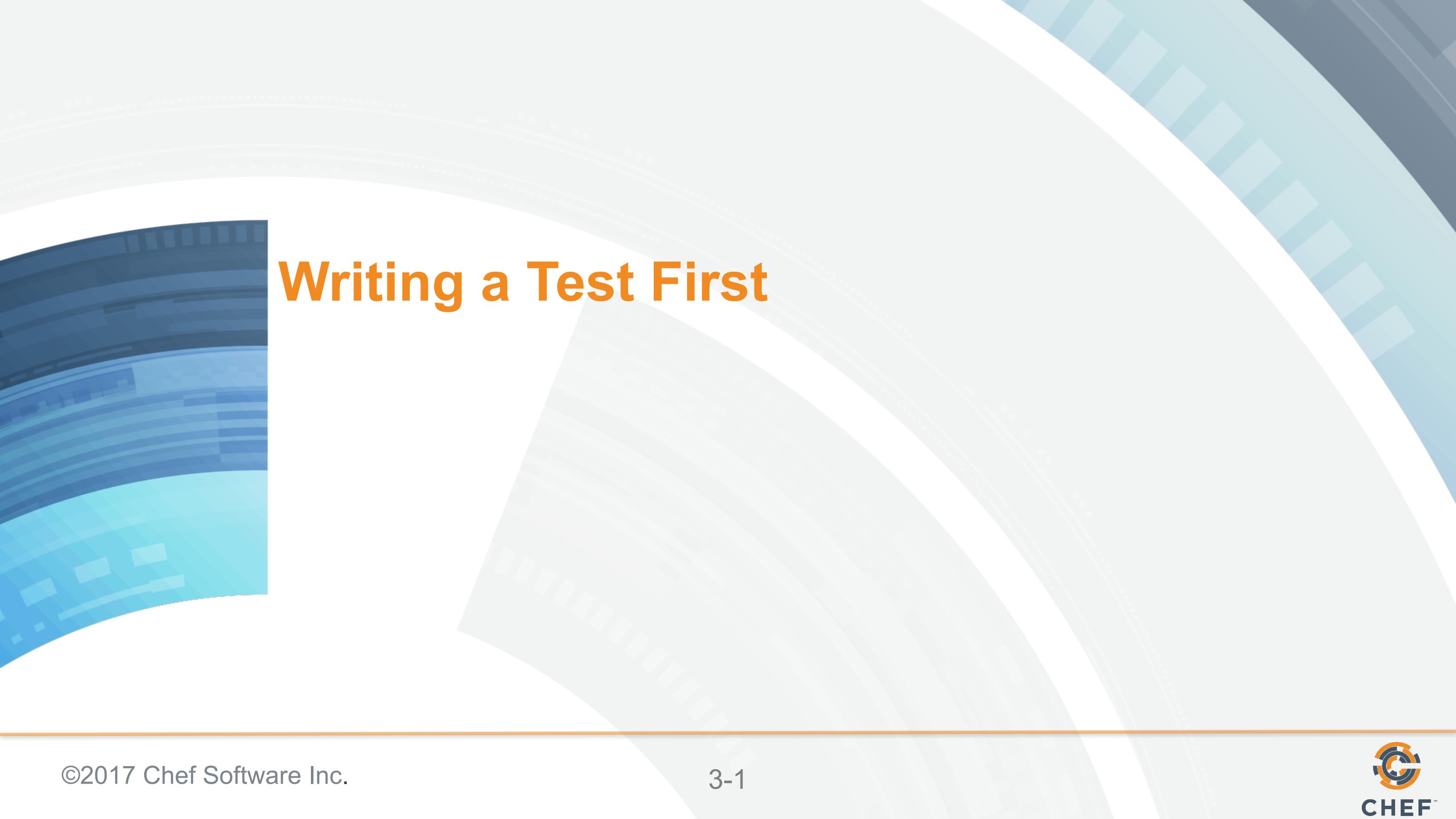
Q&A

What questions can we answer for you?



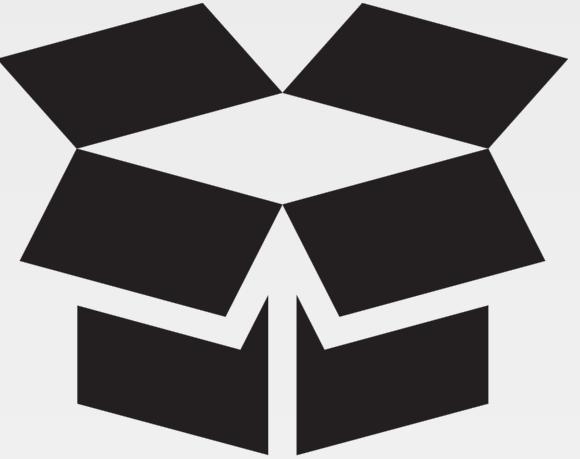
CHEF™





Writing a Test First

CONCEPT



Test Driven Development

1. Define a test set for the unit first
2. Then implement the unit
3. Finally verify that the implementation of the unit makes the tests succeed.
4. Refactor

CONCEPT



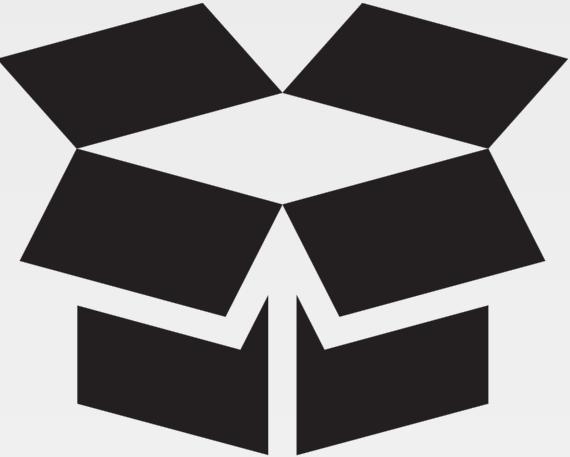
Behavior Driven Development (BDD)

Behavior-driven development (BDD) specifies that tests of any unit of software should be specified in terms of the desired behavior of the unit.

Borrowing from [agile software development](#) the "desired behavior" in this case consists of the requirements set by the business — that is, the desired behavior that has [business value](#) for whatever entity commissioned the software unit under construction.

Within BDD practice, this is referred to as BDD being an "outside-in" activity.

CONCEPT



TDD and BDD

TDD is a workflow process.

BDD influences the language we use to write tests and how we focus on the tests that matter.

Objectives

After completing this module, you should be able to:

- Write an integration test
- Use Test Kitchen to create, converge, and verify a recipe
- Develop a cookbook with a test-driven approach

Building a Web Server

1. Install the httpd package
2. Write out a test page
3. Start and enable the httpd service

Defining Scenarios

Given SOME CONDITIONS

When an EVENT OCCURS

Then I should EXPECT THIS RESULT

The Why Stack?

You should discuss...the feature and [pop the why stack](#) max 5 times (ask why recursively) until you end up with one of the following business values:

- Protect revenue
- Increase revenue
- Manage cost

If you're about to implement a feature that doesn't support one of those values, chances are you're about to implement a non-valuable feature. Consider tossing it altogether or pushing it down in your backlog.

- Aslak Hellesøy, creator of Cucumber

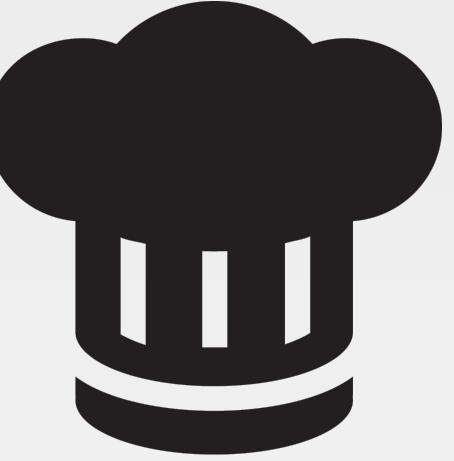
Scenario: Potential User Visits Website

Given that I am a potential user

When I visit the company website in my browser

Then I should see a welcome message

EXERCISE



Build a Reliable Cookbook

This time it will be different.

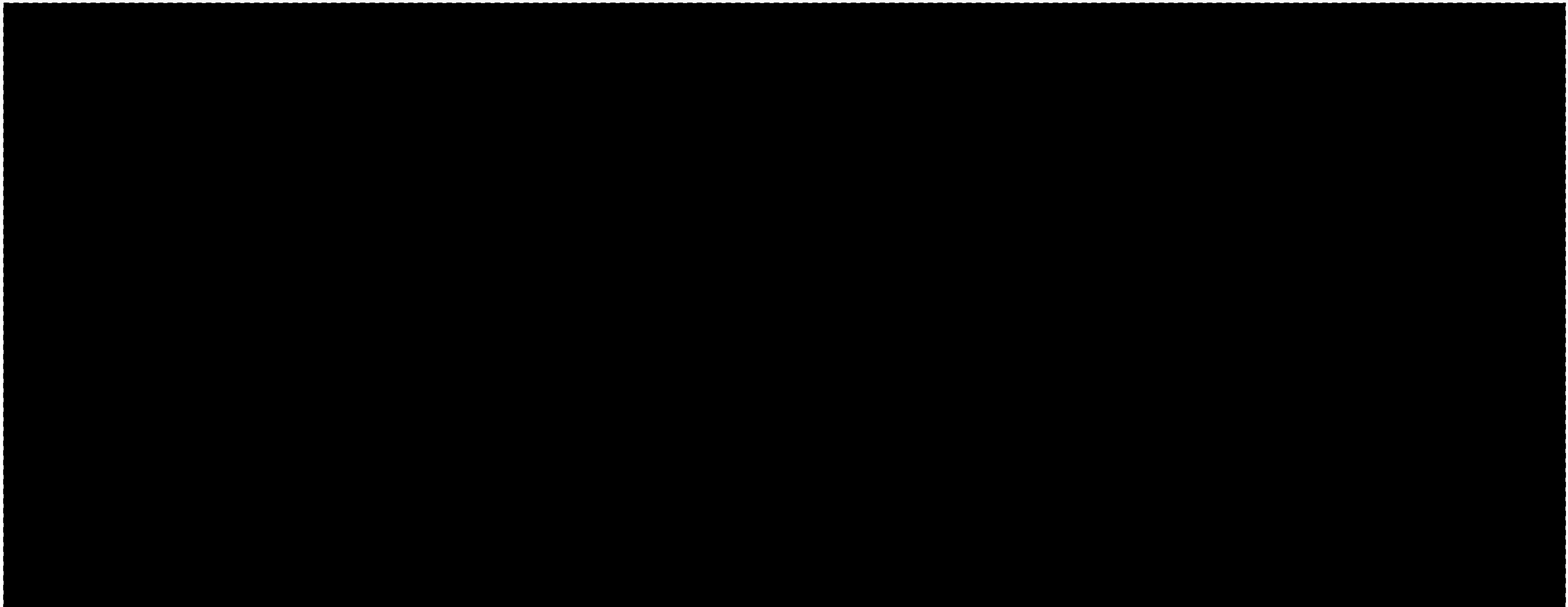
Objective:

- Examine the cookbook
- Write tests that verifies the cookbook does what we want it to do
- Execute the tests and see failure
- Write the recipe to make the test pass
- Execute the tests and see success

Let's Start this Journey in the Home Directory



```
> cd ~
```



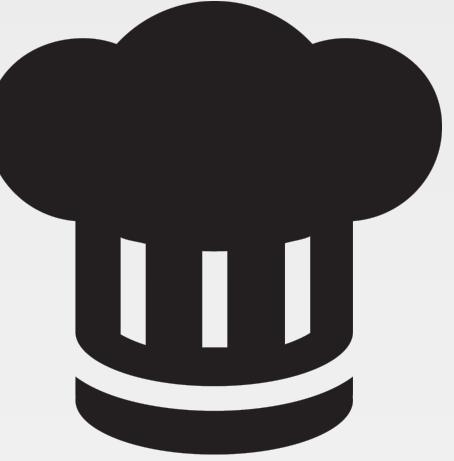
View the Tests in the Generated Cookbook



```
> tree apache
```

```
apache/
├── Berksfile
├── cheffignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
└── spec
    ...
6 directories, 8 files
```

EXERCISE



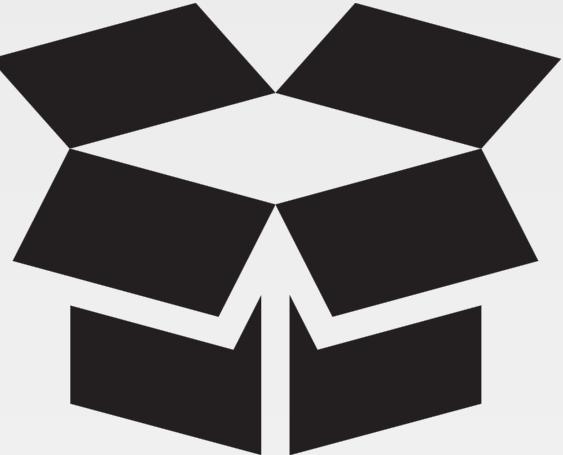
Build a Reliable Cookbook

This time it will be different.

Objective:

- ✓ Examine the cookbook
- Write tests that verifies the cookbook does what we want it to do
- Execute the tests and see failure
- Write the recipe to make the test pass
- Execute the tests and see success

CONCEPT



RSpec and InSpec

RSpec is a Domain Specific Language (DSL) that allows you to express and execute expectations. These expectations are expressed in examples that are asserted in different example groups.

InSpec provides helpers and tools that allow you to express expectations about the state of infrastructure.

InSpec

RSpec

Chef

Ruby

Auto-generated Spec File in Cookbook

~/apache/test/smoke/default/default_test.rb

```
unless os.windows?

  describe user('root'), :skip do
    it { should exist }

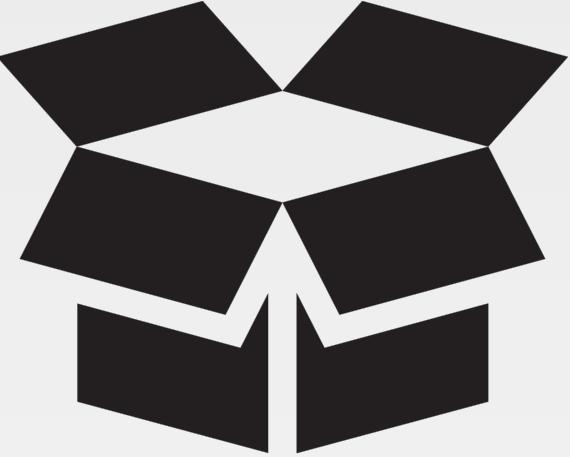
  end

end

describe port(80), :skip do
  it { should_not be_listening }

end
```

CONCEPT

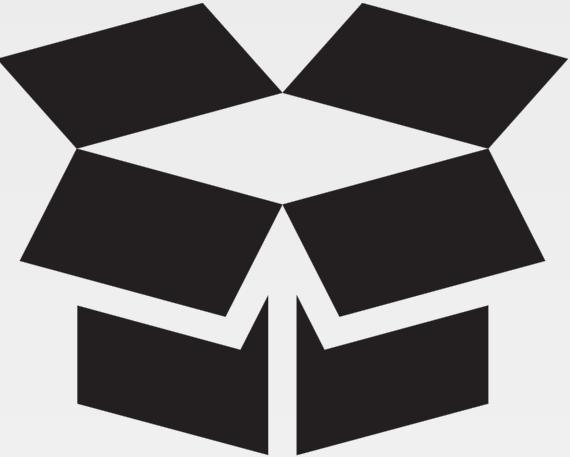


Where do Tests Live?

```
~/apache/test/smoke/default/default_test.rb
```

Test Kitchen will look for tests to run under this directory. This corresponds to the value specified in the Test Kitchen configuration file (.kitchen.yml) in the suites section.

CONCEPT



Where do Tests Live?

```
~/apache/test/smoke/default/default_test.rb
```

The default_test.rb file is a Ruby file that contains the tests that we want to run when we spin up a test instance.

Components of a InSpec Example

```
unless os.windows?  
  describe user('root'), :skip do  
    it { should exist }  
  end  
end
```

OS conditional

InSpec resource

expectation

When not on Windows, I expect the user named 'root', to exist.

Components of a InSpec Example

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

InSpec resource

expectation

When on any platform, I expect the port 80 **not** to be listening for incoming connections.

Remove the Test for the root User

~/apache/test/smoke/default/default_test.rb

```
unless os.windows?  
  describe user('root'), :skip do  
    it { should exist }  
  end  
end  
  
describe port(80), :skip do  
  it { should_not be_listening }  
end
```

Update the Test for Port 80

~/apache/test/smoke/default/default_test.rb

```
# ... FIRST EXAMPLE DELETED ...

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

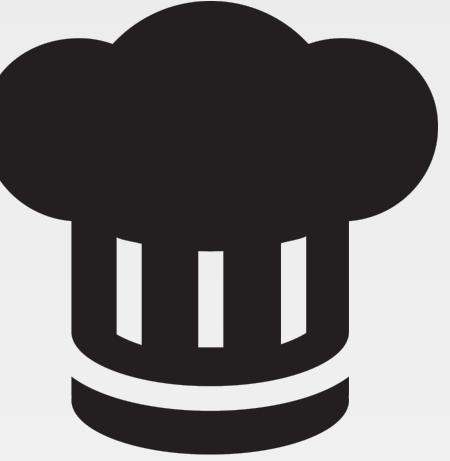
Add a Test to Validate a Working Website

```
~/apache/test/smoke/default/default_test.rb
```

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

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

EXERCISE



Build a Reliable Cookbook

This time it will be different.

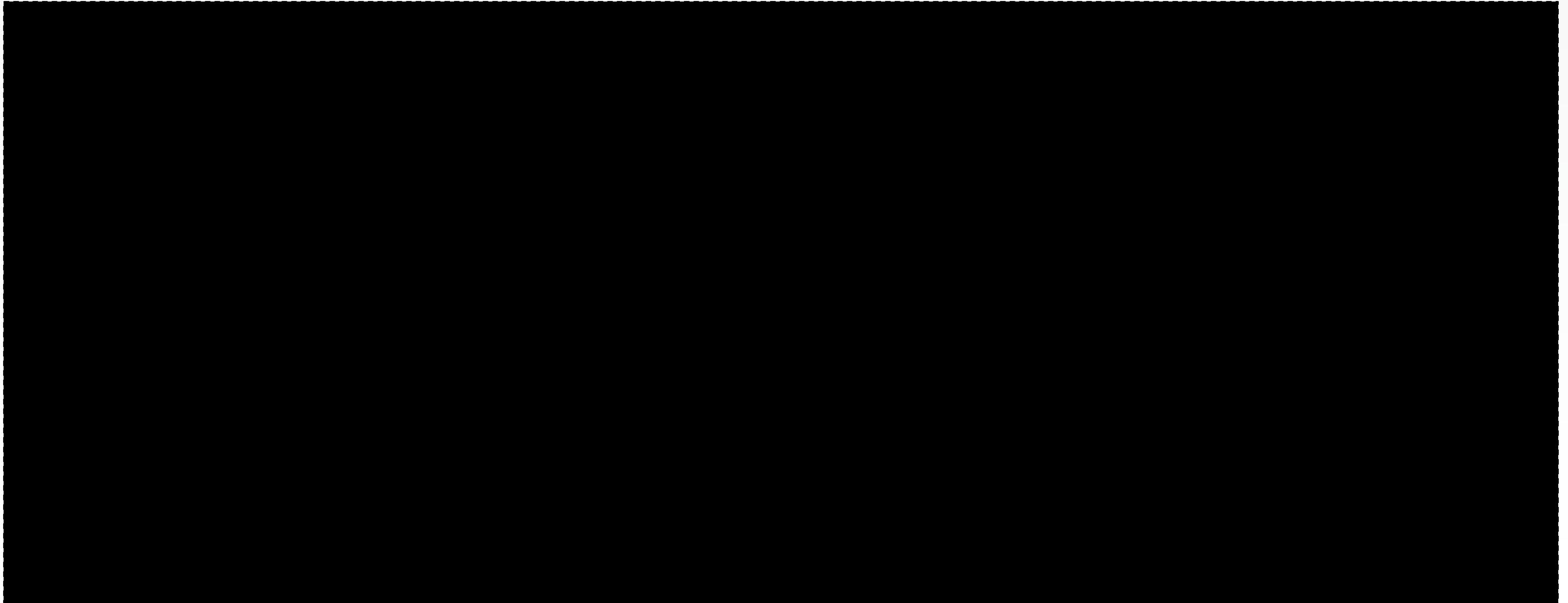
Objective:

- ✓ Examine the cookbook
- ✓ Write tests that verifies the cookbook does what we want it to do
- Execute the tests and see failure
- Write the recipe to make the test pass
- Execute the tests and see success

Move into the Cookbook Directory



```
> cd apache
```



Review the Existing Kitchen Configuration



```
> cat .kitchen.yml
```

```
---
```

```
driver:
```

```
  name: vagrant
```

```
provisioner:
```

```
  name: chef_zero
```

```
  # You may wish to disable always updating cookbooks in CI or...
```

```
  # For example:
```

```
  #   always_update_cookbooks: <%= !ENV['CI'] %>
```

```
  always_update_cookbooks: true
```

The Kitchen Driver

```
--  
driver:  
  name: vagrant  
  
provisioner:  
  name: chef_zero  
  
verifier:  
  name: inspec  
  
platforms:  
  - name: ubuntu-16.04  
  - name: centos-7.3
```

The driver is responsible for creating a machine that we'll use to test our cookbook.

Example Drivers:

- docker
- vagrant

The Kitchen Provisioner

```
---  
driver:  
  name: vagrant  
  
provisioner:  
  name: chef_zero  
  
verifier:  
  name: inspec  
  
platforms:  
  - name: ubuntu-16.04  
  - name: centos-7.3
```

This tells Test Kitchen how to run Chef, to apply the code in our cookbook to the machine under test.

The default and simplest approach is to use `chef_zero`.

The Kitchen Verifier

```
--  
driver:  
  name: vagrant  
  
provisioner:  
  name: chef_zero  
  
verifier:  
  name: inspec  
  
platforms:  
  - name: ubuntu-16.04  
  - name: centos-7.3
```

This is the framework that is used to verify the state of the system meets the expectations defined.

The Kitchen Platforms

```
--  
driver:  
  name: vagrant  
  
provisioner:  
  name: chef_zero  
  
verifier:  
  name: inspec  
  
platforms:  
  - name: ubuntu-16.04  
  - name: centos-7.3
```

This is a list of platforms on which we want to apply our recipes.

The Kitchen Suites

```
platforms:
```

- name: ubuntu-16.04
- name: centos-7.3

```
suites:
```

- name: default

```
  run_list:
```

- recipe[apache::default]

```
  verifier:
```

```
    inspec_tests:
```

- test/smoke/default

```
  attributes:
```

This section defines what we want to test. It includes the Chef run-list of recipes that we want to test.

We define a single suite named "default".

The Kitchen Suites' Run List

```
platforms:
  - name: ubuntu-16.04
  - name: centos-7.3

suites:
  - name: default
    run_list:
      - recipe[apache::default]

verifier:
  inspec_tests:
    - test/smoke/default

attributes:
```

The suite named "default" defines a `run_list`.

Run the "apache" cookbook's "default" recipe file.

The Kitchen Suites' Tests

```
platforms:  
  - name: ubuntu-16.04  
  - name: centos-7.3
```

```
suites:  
  - name: default  
    run_list:  
      - recipe[apache::default]
```

```
    verifier:  
      inspec_tests:  
        - test/smoke/default
```

```
      attributes:
```

This is the path where the InSpec tests can be found.

Remove Settings from the Kitchen Configuration

~/apache/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.3
```

Add Settings to the Kitchen Configuration

~/apache/.kitchen.yml

```
---
```

```
driver:
```

```
  name: docker
```

```
provisioner:
```

```
  name: chef_zero
```

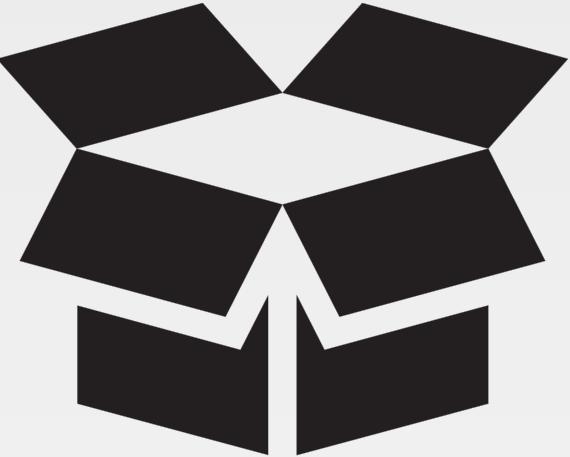
```
verifier:
```

```
  name: inspec
```

```
platforms:
```

```
  - name: centos-6.7
```

CONCEPT



Kitchen List

Kitchen defines a list of instances, or test matrix, based on the **platforms** multiplied by the **suites**.

PLATFORMS x SUITES

Running `kitchen list` will show that matrix.

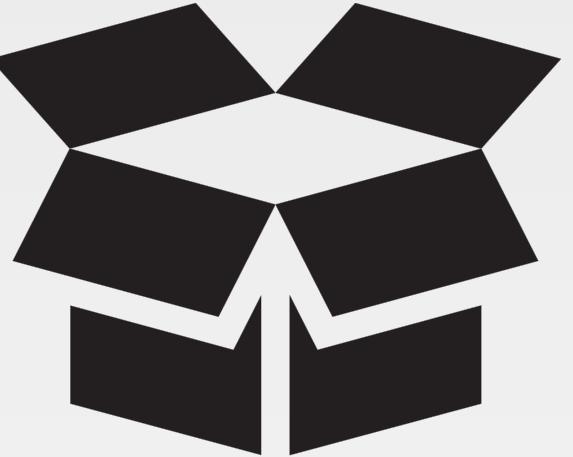
View the Test Matrix for Test Kitchen



```
> kitchen list
```

Instance	Driver	Provisioner	Verifier	Transport	Last Action
default-centos-67	Docker	ChefZero	InSpec	Ssh	<Not Created>

CONCEPT



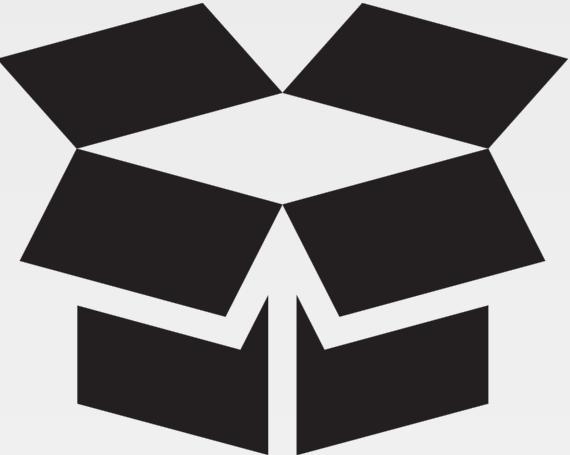
Kitchen Create

```
$ kitchen create [INSTANCE | REGEXP | all]
```

Create one or more instances.

[Create CentOS Instance](#)

CONCEPT



Kitchen Converge

```
$ kitchen converge [INSTANCE | REGEXP | all]
```

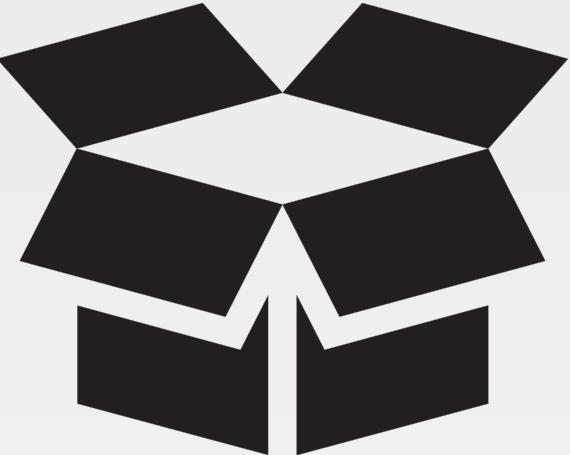
Create the instance (if necessary) and then apply the run list to one or more instances.

Create CentOS Instance

Install Chef

Apply the Run List

CONCEPT



Kitchen Verify

```
$ kitchen verify [INSTANCE | REGEXP | all]
```

Create, converge, and verify one or more instances.

Create CentOS Instance

Install Chef

Apply the Run List

Execute Tests

Create the Virtual Instance



```
> kitchen create
```

```
-----> Starting Kitchen (v1.11.1)
-----> Creating <default-centos-67>...
      Sending build context to Docker daemon  193 kB
      Sending build context to Docker daemon
      Step 0 : FROM centos:centos6
      centos6: Pulling from centos
      3690474eb5b4: Pulling fs layer
      c12ea02d7eb2: Pulling fs layer
      334af8693ca8: Verifying Checksum
      334af8693ca8: Download complete
      273a1eca2d3a: Verifying Checksum
```

Inspect the Virtual Instance



```
> kitchen login
```

```
$$$$$$ Running legacy login for 'Docker' Driver
Last login: Thu Feb 18 21:21:39 2017 from 172.17.42.1
[kitchen@4eae2dd9e741 ~] $
```

Exit the Virtual Instance



```
[kitchen@4eae2dd9e741 ~]$ exit
```

```
logout
```

```
Connection to localhost closed.
```

```
[chef@ip-172-31-14-170 apache]$
```

Converge the Virtual Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.11.1)
----> Converging <default-centos-67>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["apache::default"]
...
Finished converging <default-centos-67> (0m27.64s) .
----> Kitchen is finished. (0m28.58s)
```

Execute the Tests Against the Virtual Instance



```
> kitchen verify
```

```
-----> Starting Kitchen (v1.11.1)
-----> Setting up <default-centos-67>...
-----> Verifying <default-centos-67>...
Use `/home/chef/apache/test/smoke/default` for testing
```

Target: ssh://kitchen@localhost:32768

- ✖ Port 80 should be listening (expected `Port 80.listening?...`)
- ✖ Command curl localhost stdout should match /Hello, world/...

Understanding the Failure Message

```
Target: ssh://kitchen@localhost:32768
```

```
* Port 80 should be listening (expected `Port 80.listening?` to return true, got false)
* Command curl localhost stdout should match /Welcome Home/ (expected "" to match /Welcome
Home/)
```

```
Diff:
```

```
@@ -1,2 +1,2 @@
-/Welcome Home/
+""
```

```
Summary: 0 successful, 2 failures, 0 skipped
```

```
>>>> -----Exception-----
```

```
>>>> Class: Kitchen::ActionFailed
```

```
>>>> Message: 1 actions failed.
```

```
>>>> Verify failed on instance <default-centos-67>. Please see .kitchen/logs/defau...
```

Examine Failure #1

```
* Port 80 should be listening (expected `Port 80.listening?` to return true, got false)
* Command curl localhost stdout should match /Welcome Home/ (expected "" to match /Welcome Home/)
```

Diff.

```
@@ -1,2 +1,2 @@
-/Welcome Home/
```

```
+"""
)
```

actual results

difference

Examine the Test Summary

```
* Port 80 should be listening (expected `Port 80.listening?` to return true, got false)
* Command curl localhost stdout should match /Welcome Home/ (expected "" to match /Welcome Home/)
```

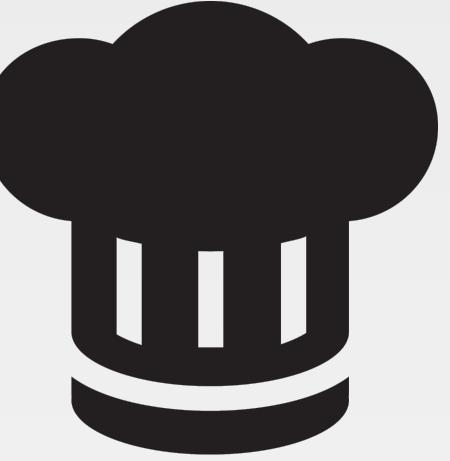
Diff:

```
@@ -1,2 +1,2 @@
-/Welcome Home/
+"""
)
```

Summary: 0 successful, 2 failures, 0 skipped

A final summary contains the length of execution time with the results shows that RSpec verified 2 examples and found 2 failures.

EXERCISE



Build a Reliable Cookbook

This time it will be different.

Objective:

- ✓ Examine the cookbook
- ✓ Write tests that verifies the cookbook does what we want it to do
- ✓ Execute the tests and see failure
- Write the recipe to make the test pass
- Execute the tests and see success

Write the Default Recipe for the Cookbook

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 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
```

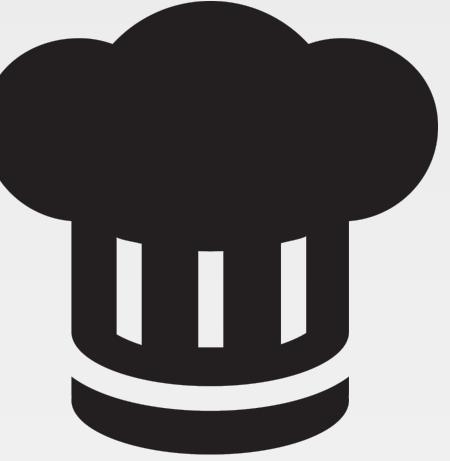
Re-Converge the Virtual Instance



```
> kitchen converge
```

```
-----> Starting Kitchen (v1.11.1)
Converging 3 resources
Recipe: apache::default
  * package[httpd] action install
    - install version 2.2.15-47.el6.centos of package httpd
  * file[/var/www/html/index.html] action create
    - ...
  * service[httpd] action enable
    - enable service service[httpd]
  * service[httpd] action start
    - start service service[httpd]
```

EXERCISE



Build a Reliable Cookbook

This time it will be different.

Objective:

- ✓ Examine the cookbook
- ✓ Write tests that verifies the cookbook does what we want it to do
- ✓ Execute the tests and see failure
- ✓ Write the recipe to make the test pass
- Execute the tests and see success

Re-Verify the Virtual Instance



```
> kitchen verify
```

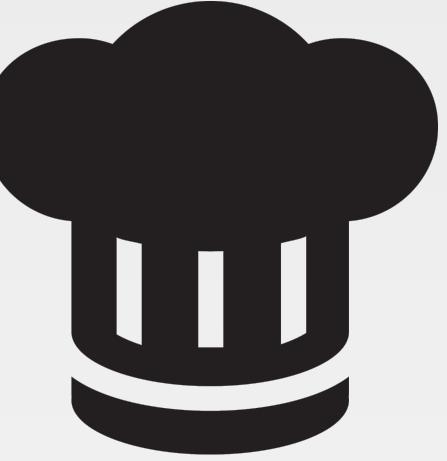
```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-67>...
      Use `~/home/chef/apache/test/smoke/default` for testing
```

Target: ssh://kitchen@localhost:32768

- ✓ Port 80 should be listening
- ✓ Command curl localhost stdout should match /Welcome Home/

Summary: 2 successful, 0 failures, 0 skipped

EXERCISE



Build a Reliable Cookbook

This time it will be different.

Objective:

- ✓ Examine the cookbook
- ✓ Write tests that verifies the cookbook does what we want it to do
- ✓ Execute the tests and see failure
- ✓ Write the recipe to make the test pass
- ✓ Execute the tests and see success

DISCUSSION



Discussion

What value is there in writing the tests before writing the recipes?

Why is it hard to write the tests before you write the recipe?

DISCUSSION



Q&A

What questions can we answer for you?

Morning

Introduction

Why Write Tests? Why is that Hard?

Writing a Test First

Refactoring Cookbooks with Tests

Afternoon

Faster Feedback with Unit Testing

Testing Resources in Recipes

Refactoring to Attributes

Refactoring to Multiple Platforms

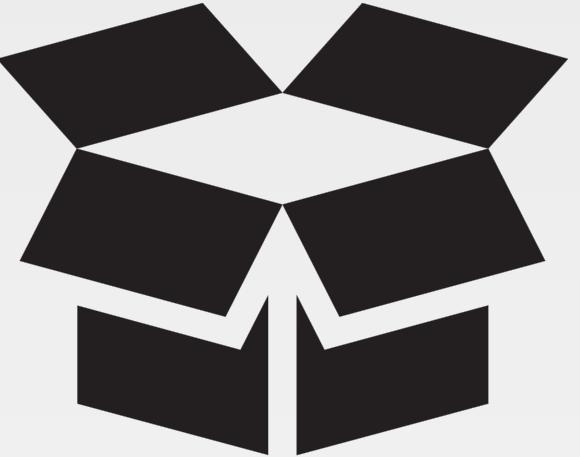


CHEF™

Refactoring Cookbooks with Tests



CONCEPT



Test Driven Development

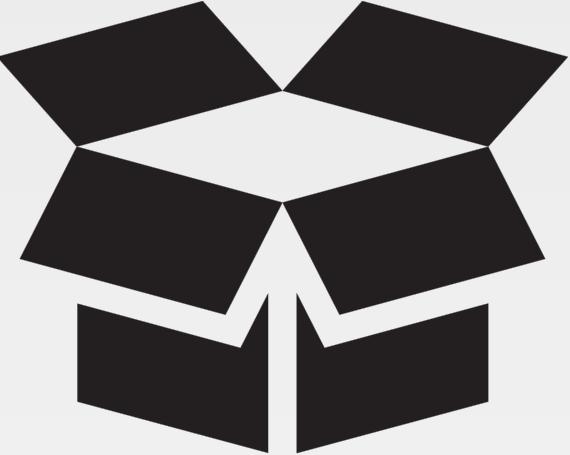
1. Define a test set for the unit first
2. Then implement the unit
3. Finally verify that the implementation of the unit makes the tests succeed.
4. **Refactor**

Objectives

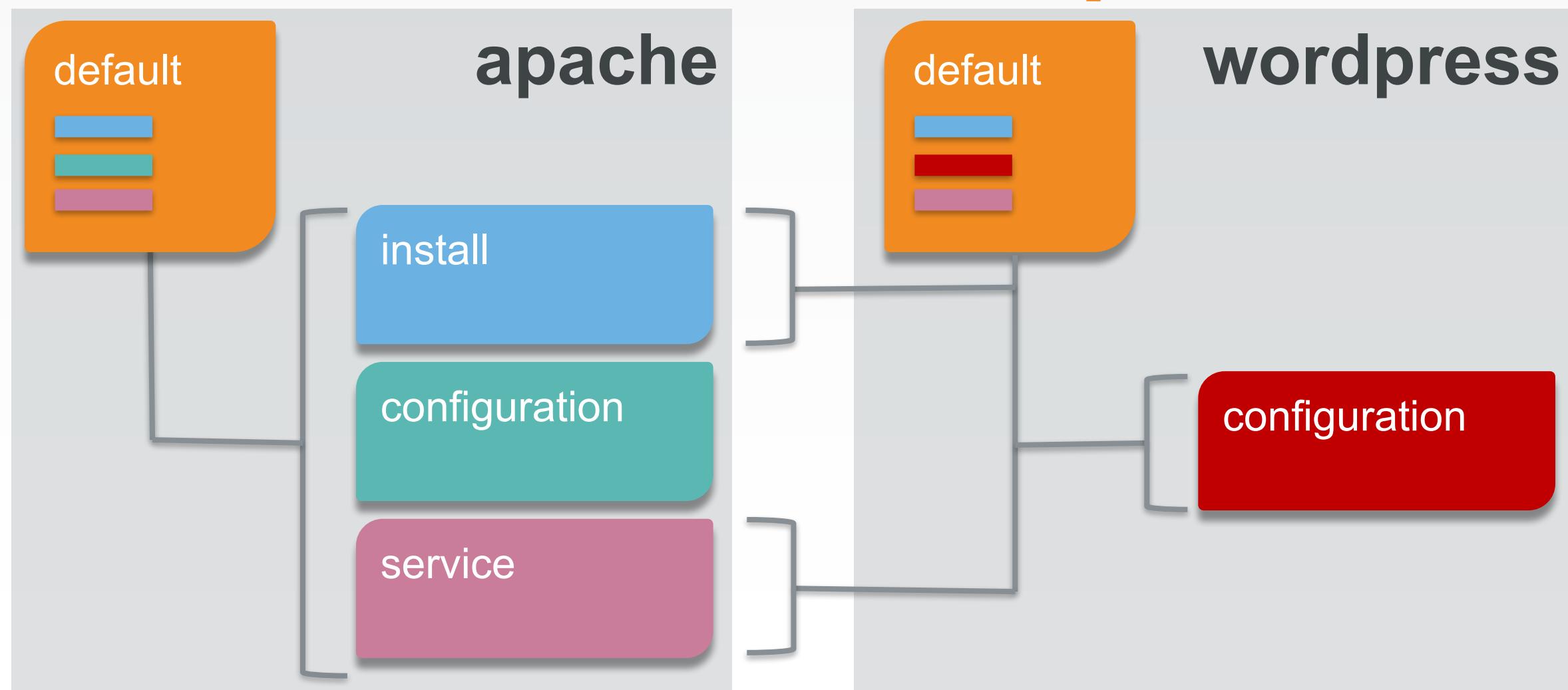
After completing this module, you should be able to:

- Refactor a recipe using `include_recipe`
- Use Test Kitchen to validate the code you refactored
- Explain when to use `kitchen converge`, `kitchen verify` and `kitchen test`.

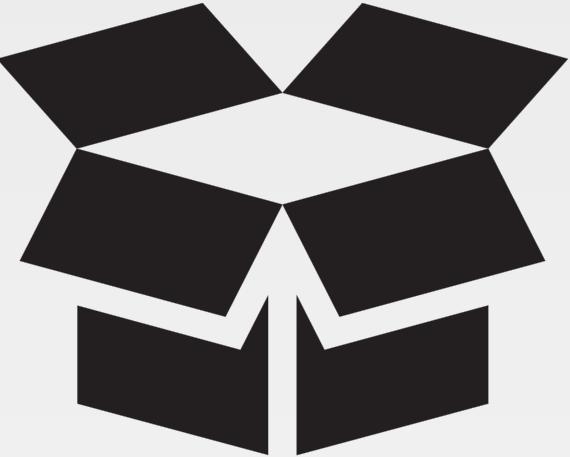
CONCEPT



Modular Cookbook Recipes



CONCEPT

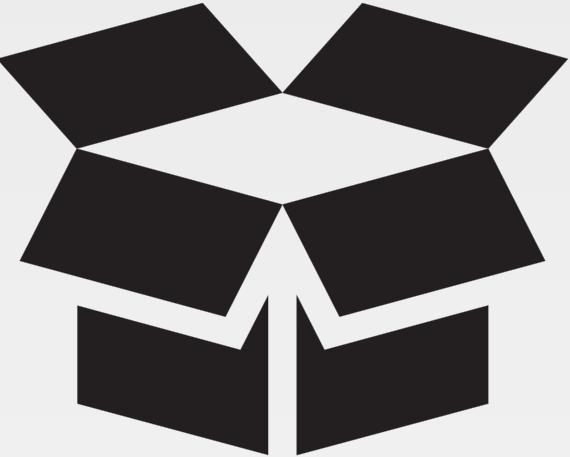


include_recipe

A recipe can include one (or more) recipes located in cookbooks by using the `include_recipe` method. When a recipe is included, the resources found in that recipe will be inserted (in the same exact order) at the point where the `include_recipe` keyword is located.

<https://docs.chef.io/recipes.html#include-recipes>

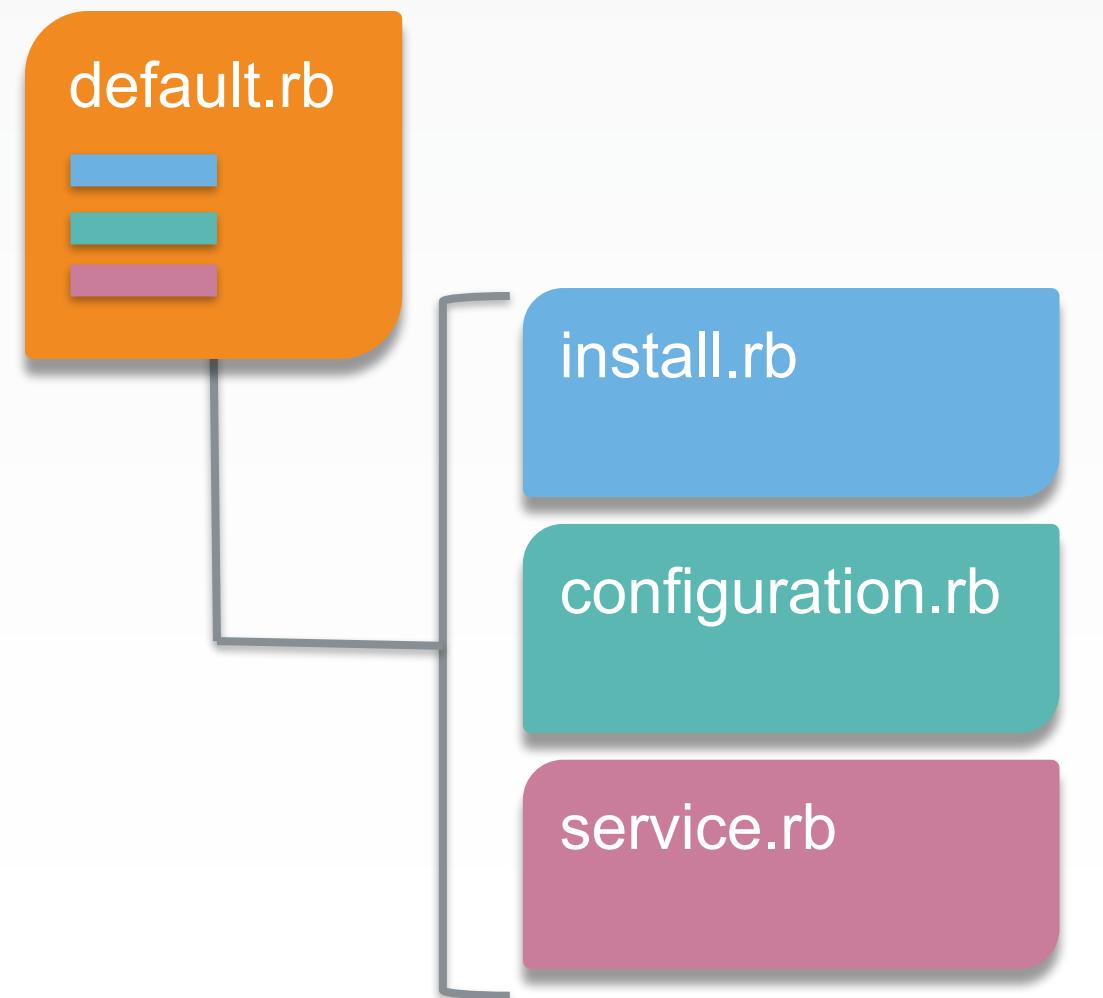
CONCEPT



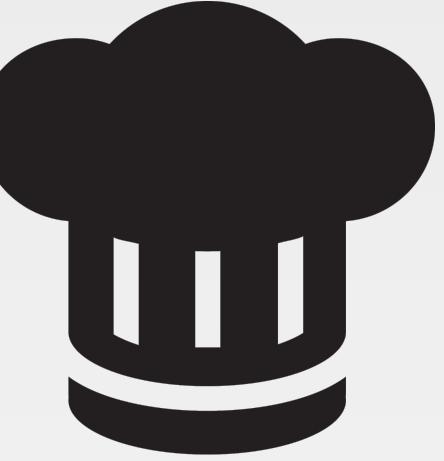
Recipe Organization

recipes/default.rb

```
include_recipe 'cookbook::install'  
include_recipe 'cookbook::configuration'  
include_recipe 'cookbook::service'
```



EXERCISE



Refactor to Modular Recipes

*This is why we **can** have nice things!*

Objective:

- Refactor the installation into a separate recipe
- Converge the cookbook and execute the tests

You called?



Ask Chef About Generating a Recipe



```
> chef generate recipe --help
```

Usage: **chef generate recipe [path/to/cookbook] NAME [options]**

-C, --copyright COPYRIGHT	Name of the copyright hol...
-m, --email EMAIL	Email address of the auth...
-a, --generator-arg KEY=VALUE	Use to set arbitrary ...
-I, --license LICENSE	all_rights, apache2, mit,...
-g GENERATOR_COOKBOOK_PATH, --generator-cookbook	Use GENERATOR_COOKBOOK_PA...

Generate an Install Recipe



```
> chef generate recipe install
```

```
Recipe: code_generator::recipe
  * directory[/home/chef/apache/spec/unit/recipes] action create
    (up to date)
      * cookbook_file[/home/chef/apache/spec/spec_helper.rb] action
        create_if_missing (up to date)
          * template[/home/chef/apache/spec/unit/recipes/install_spec.rb]
            action create_if_missing
              - create new file
                /home/chef/apache/spec/unit/recipes/install_spec.rb
                  - update content in file
                    /home/chef/apache/spec/unit/recipes/install_spec.rb from none to
                      187413
```

Removing the Generated Test File



```
> rm test/smoke/default/install.rb
```

Write the Install Recipe

~/apache/recipes/install.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: install  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
package 'httpd'
```

Remove the Resource from the Default Recipe

~/apache/recipes/default.rb

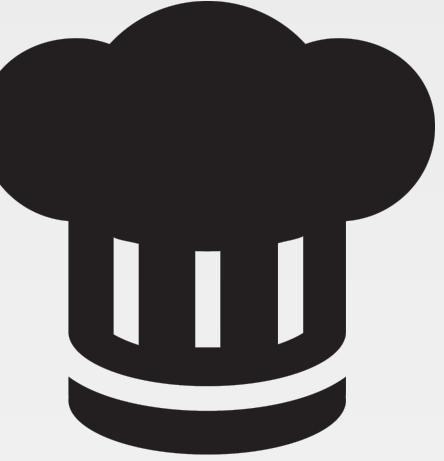
```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
package 'httpd'  
  
file '/var/www/html/index.html' do  
  content '<h1>Welcome Home!</h1>'  
end  
  
service 'httpd' do  
  action [:enable, :start]  
end
```

Include the Install Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
  
file '/var/www/html/index.html' do  
  content '<h1>Welcome Home!</h1>'  
end  
  
service 'httpd' do  
  action [:enable, :start]  
end
```

EXERCISE



Refactor to Modular Recipes

*This is why we **can** have nice things!*

Objective:

- ✓ Refactor the installation into a separate recipe
- ❑ Converge the cookbook and execute the tests

I see what you did there.



Re-Converge the Test Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.11.1)
-----> Converging <default-centos-67>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
-----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["apache::default"]
...
      Finished converging <default-centos-67> (0m27.64s) .
-----> Kitchen is finished. (0m28.58s)
```

Re-Verify the Test Instance



```
> kitchen verify
```

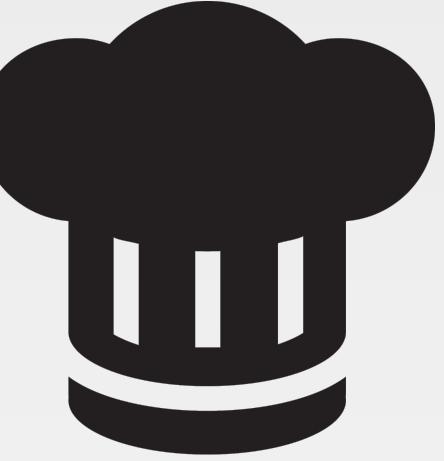
```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

Target: ssh://kitchen@localhost:32770

- ✓ Port 80 should be listening
- ✓ Command curl localhost stdout should match /Welcome Home/

Summary: 2 successful, 0 failures, 0 skipped

EXERCISE



Refactor to Modular Recipes

*This is why we **can** have nice things!*

Objective:

- ✓ Refactor the installation into a separate recipe
- ✓ Converge the cookbook and execute the tests

Nice shave!





The Configuration

- Create a configuration recipe that defines the policy:

The file named '/var/www/html/index.html' contains the content '<h1>Welcome Home!</h1>'

- Delete the automatically generated InSpec test
- Within the default recipe replace the file resource with an include recipe
- Converge and verify the test instance to ensure there are no failures

Generate a Service Recipe



```
> chef generate recipe configuration
```

```
Recipe: code_generator::recipe
  * directory[/home/chef/apache/spec/unit/recipes] action create
    (up to date)
  * cookbook_file[/home/chef/apache/spec/spec_helper.rb] action
    create_if_missing (up to date)
  *
  template[/home/chef/apache/spec/unit/recipes/configuration_spec.rb
  ] action create_if_missing
    - create new file
/home/chef/apache/spec/unit/recipes/configuration_spec.rb
    - update content in file
/home/chef/apache/spec/unit/recipes/configuration_spec.rb from
```

Remove the Generated Test File



```
> rm test/smoke/default/configuration.rb
```

Write the Configuration Recipe

~/apache/recipes/configuration.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: configuration  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
file '/var/www/html/index.html' do  
  content '<h1>Welcome Home!</h1>'  
end
```

Remove the Resource from the Default Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
  
file '/var/www/html/index.html' do  
  content '<h1>Welcome Home!</h1>'  
end  
  
service 'httpd' do  
  action [:enable, :start]  
end
```

Include the Configuration Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
  
service 'httpd' do  
  action [:enable, :start]  
end
```

Re-Converge the Test Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.11.1)
-----> Converging <default-centos-67>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
-----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["apache::default"]
...
      Finished converging <default-centos-67> (0m27.64s) .
-----> Kitchen is finished. (0m28.58s)
```

Re-Verify the Test Instance



```
> kitchen verify
```

```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

Target: ssh://kitchen@localhost:32770

- ✓ Port 80 should be listening
- ✓ Command curl localhost stdout should match /Welcome Home/

Summary: 2 successful, 0 failures, 0 skipped

LAB



The Configuration

- ✓ Create a configuration recipe that defines the policy:

The file named '/var/www/html/index.html' contains the content '<h1>Welcome Home!</h1>'

- ✓ Delete the automatically generated InSpec test
- ✓ Within the default recipe replace the file resource with an include recipe
- ✓ Converge and verify the test instance to ensure there are no failures

LAB



The Service

- ❑ Create a service recipe that defines the policy:

The service named 'httpd' is started and enabled

- ❑ Delete the automatically generated InSpec test
- ❑ Within the default recipe replace the service resource with an include recipe
- ❑ Converge and verify the test instance to ensure there are no failures

One last time!



Generate a Service Recipe



```
> chef generate recipe service
```

```
Recipe: code_generator::recipe
  * directory[/home/chef/apache/spec/unit/recipes] action create
    (up to date)
  * cookbook_file[/home/chef/apache/spec/spec_helper.rb] action
    create_if_missing (up to date)
  * template[/home/chef/apache/spec/unit/recipes/service_spec.rb]
    action create_if_missing
      - create new file
/home/chef/apache/spec/unit/recipes/service_spec.rb
      - update content in file
/home/chef/apache/spec/unit/recipes/service_spec.rb from none to
1f669c
```

Remove the Generated Test File



```
> rm test/smoke/default/service.rb
```

Write the Services Recipe

~/apache/recipes/service.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: service  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
service 'httpd' do  
  action [:enable, :start]  
end
```

Remove the Resource from the Default Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
  
service 'httpd' do  
  action [:enable, :start]  
end
```

Remove the Resource from the Default Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

Re-Converge the Test Instance



```
> kitchen converge
```

```
----> Starting Kitchen (v1.11.1)
-----> Converging <default-centos-67>...
$$$$$$ Running legacy converge for 'Docker' Driver
...
-----> Installing Chef Omnibus (install only if missing)
      Downloading https://www.chef.io/chef/install.sh to file...
      resolving cookbooks for run list: ["apache::default"]
...
      Finished converging <default-centos-67> (0m27.64s) .
-----> Kitchen is finished. (0m28.58s)
```

Re-Verify the Test Instance



```
> kitchen verify
```

```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

Target: ssh://kitchen@localhost:32770

- ✓ Port 80 should be listening
- ✓ Command curl localhost stdout should match /Welcome Home/

Summary: 2 successful, 0 failures, 0 skipped

LAB



The Service

- ✓ Create a service recipe that defines the policy:

The service named 'httpd' is started and enabled.

- ✓ Delete the automatically generated InSpec test
- ✓ Within the default recipe replace the service resource with an include recipe
- ✓ Converge and verify the test instance to ensure there are no failures

My hair will grow back!



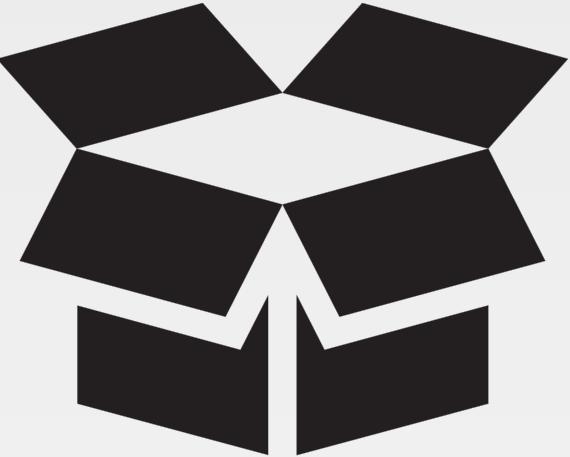
DISCUSSION



Do Our Tests Really Work?

What if we removed code from within the recipes and ran the tests?

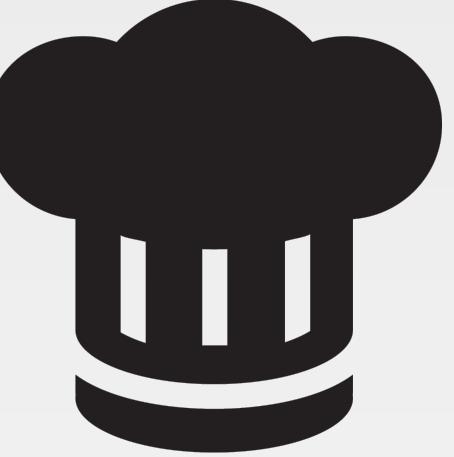
CONCEPT



Heckling Your Code

Mutation testing is used to design new software tests and evaluate the quality of existing software tests. Mutation testing involves modifying a program in small ways.

EXERCISE



Heckle That Code

It could be a game show. Maybe on Twitch?

Objective:

- Remove / Comment source code
- Converge the cookbook and execute the tests

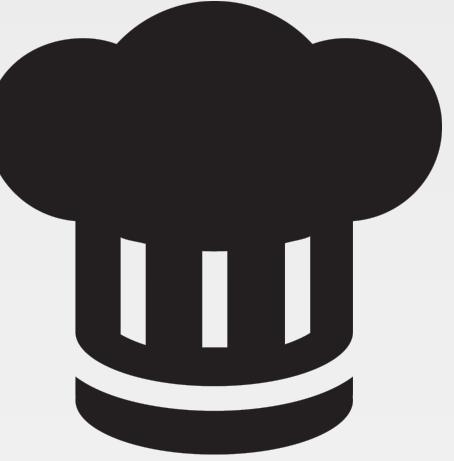
Comment Out Key Code Within the Default Recipe



~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2015 The Authors, All Rights Reserved.  
# include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

EXERCISE



Heckle That Code

It could be a game show. Maybe on Twitch?

Objective:

- Remove / Comment source code
- Converge the cookbook and execute the tests

Re-Converge the Test Instance



```
> kitchen converge
```

```
-----> Converging <default-centos-67>...
Synchronizing Cookbooks:
  - apache (0.1.0)
Compiling Cookbooks...
Converging 3 resources
Recipe: apache::configuration
(up to date)
Recipe: apache::service
(up to date)
* service[apache] action enable (up to date)
```

Re-Verify the Test Instance



```
> kitchen verify
```

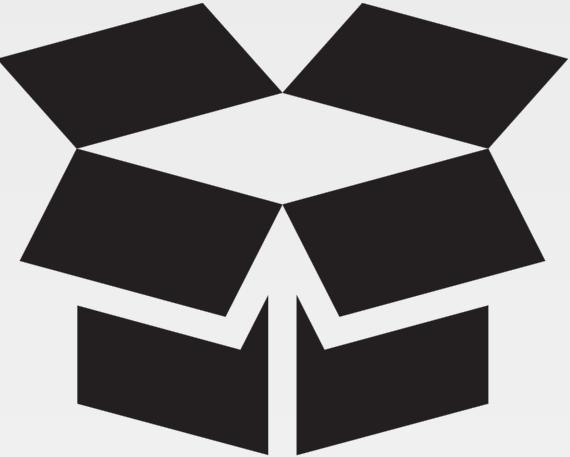
```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

Target: ssh://kitchen@localhost:32770

- ✓ Port 80 should be listening
- ✓ Command curl localhost stdout should match /Welcome Home/

Summary: 2 successful, 0 failures, 0 skipped

CONCEPT

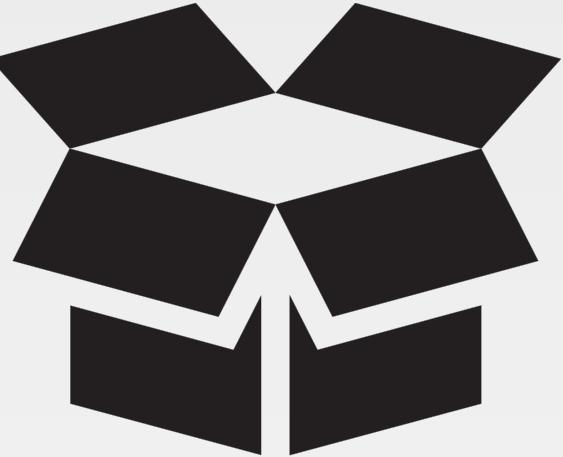


Kitchen Converge & Verify

Running converge or verify will create a new instance the first time it is run.
The same instance is used for each additional converge or verify.

The test instance policy changed, but no resource explicitly removed or
uninstalled the resources defined in the install recipe.

CONCEPT



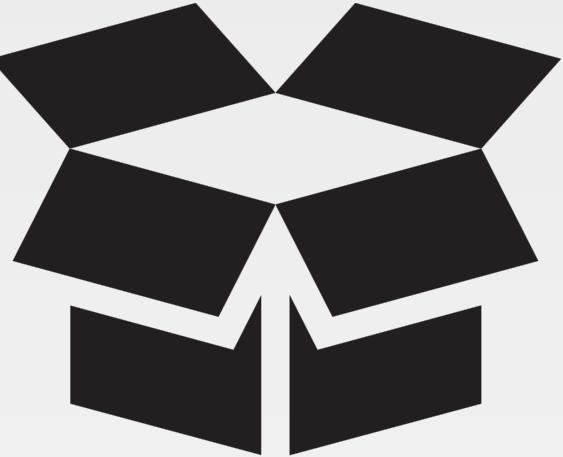
Kitchen Destroy

```
$ kitchen destroy [INSTANCE | REGEXP | all]
```

Destroys one or more instances.



CONCEPT



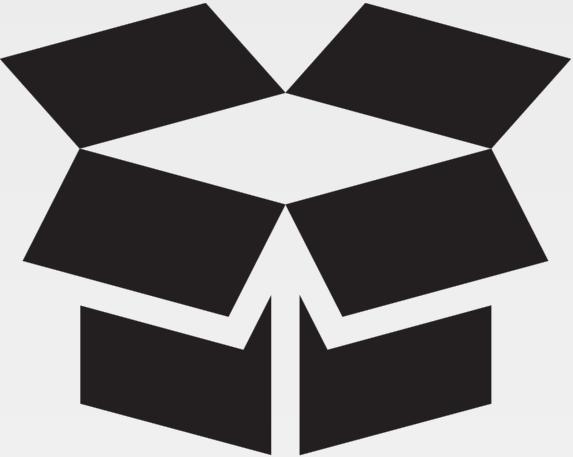
Kitchen Test

```
$ kitchen test [INSTANCE | REGEXP | all]
```

Destroys (for clean-up), creates, converges, verifies and then destroys one or more instances.



CONCEPT



Kitchen Test

Destroying the instance ensures that the policy is being applied to a new instance.

The test instance is re-created and then the updated policy is applied to the new instance. The new policy is incomplete causing an error.

Test the Cookbook Against a New Instance



```
> kitchen test
```

```
----> Starting Kitchen (v1.11.1)
-----> Cleaning up any prior instances of <default-centos-67>
-----> Destroying <default-centos-67>...
...
[2017-08-29T20:29:16+00:00] FATAL:
Chef::Exceptions::ChildConvergeError: Chef run process exited
unsuccessfully (exit code 1)
>>>> -----Exception-----
>>>> Class: Kitchen::ActionFailed
>>>> Message: 1 actions failed.
>>>>           Converge failed on instance <default-centos-67>.
```

Converge & Verify

Faster execution time

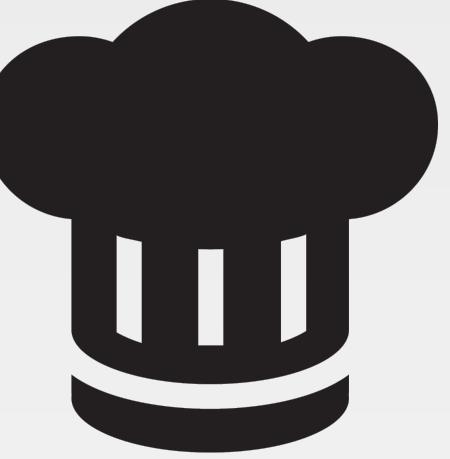
Running converge twice will ensure your policy applies without error to **existing instances**

Test

Slower execution time

Running test will ensure your policy applies without error to any **new instances**

EXERCISE



Heckle That Code

It could be a game show. Maybe on Twitch?

Objective:

- ✓ Remove / Comment source code
- ✓ Converge the cookbook and execute the tests

DISCUSSION



Discussion

What is happening when running kitchen test?

What types of bugs would kitchen converge & kitchen verify find when running?

What is the difference between kitchen test and running both kitchen converge & kitchen verify together?

How long do each of these approaches take?

DISCUSSION



Q&A

What questions can we answer for you?

Morning

Introduction

Why Write Tests? Why is that Hard?

Writing a Test First

Refactoring Cookbooks with Tests

Afternoon

Faster Feedback with Unit Testing

Testing Resources in Recipes

Refactoring to Attributes

Refactoring to Multiple Platforms



CHEF™

Faster Feedback with Unit Testing

PROBLEM



Slower Feedback Cycle

The slower the feedback loop the less value it provides to you while developing your cookbooks. You are less inclined to run the test suite. Which means you will likely miss issues as they happen.

Objectives

After completing this module, you should be able to:

- Explain the importance and limitations of unit testing
- Write and execute a unit test

CONCEPT

External Dependencies

The speed of the test suite is affected by the external dependency on the creation of the test instance, installing chef, and applying the run list.

Create CentOS Instance

Install Chef

Apply the Run List

Build Node (ohai)

Synchronize Cookbooks

Build Resource Collection

Converge

Execute Tests

CONCEPT

Build Resource Collection

The resource collection is a list of all the resources and recipes loaded across all the recipes within the run list.

Create CentOS Instance

Install Chef

Apply the Run List

Build Node (ohai)

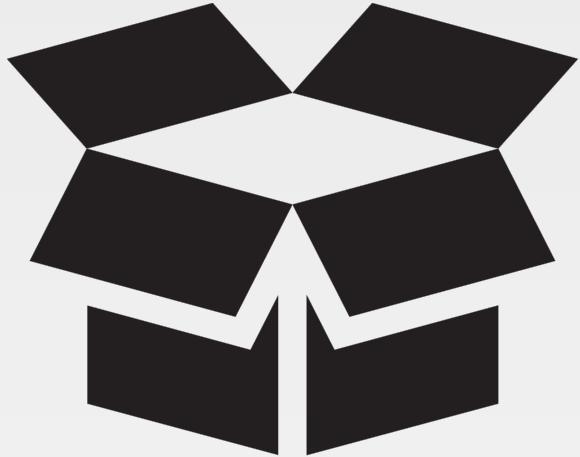
Synchronize Cookbooks

Build Resource Collection

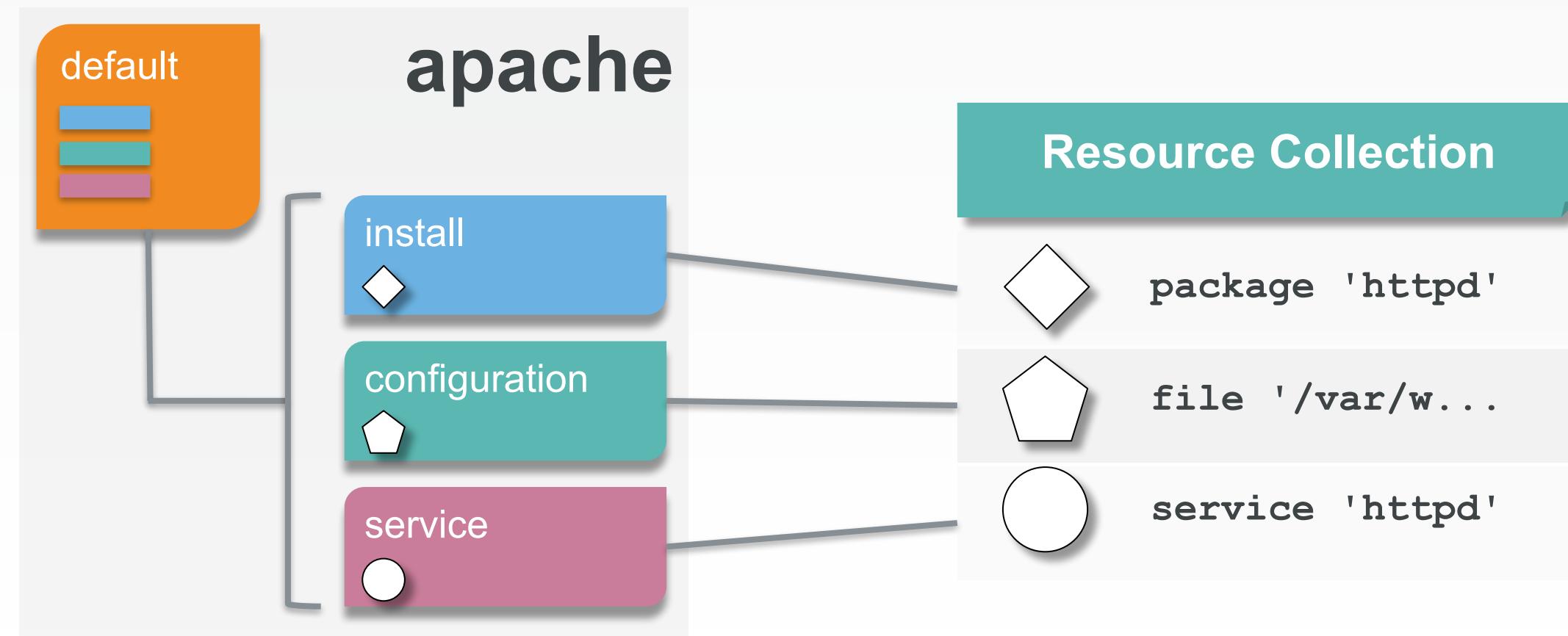
Converge

Execute Tests

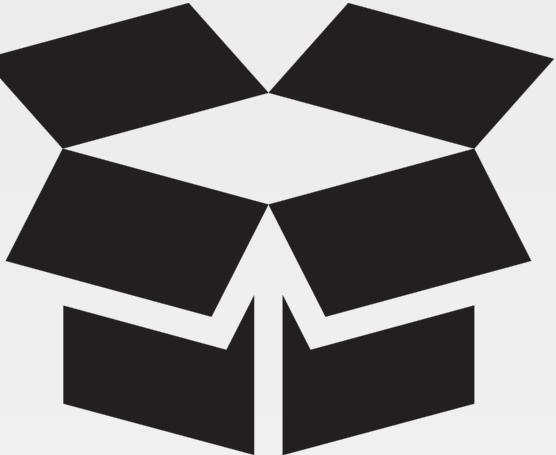
CONCEPT



Resource Collection



CONCEPT



RSpec and ChefSpec

RSpec is a Domain Specific Language (DSL) that allows you to express and execute expectations. These expectations are expressed in examples that are asserted in different example groups.

ChefSpec provides helpers and tools that allow you to express expectations about the state of **resource collection**.

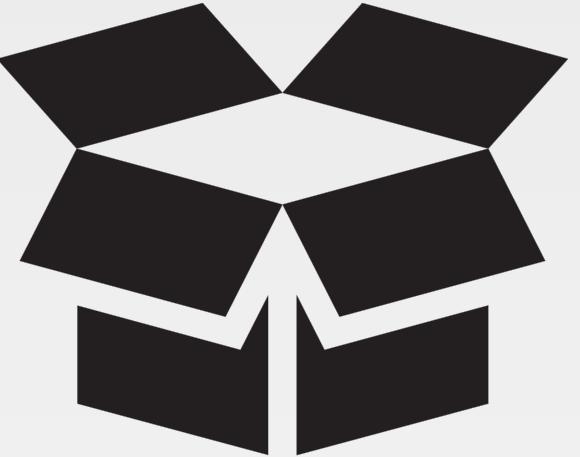
ChefSpec

RSpec

Chef

Ruby

CONCEPT



Test Kitchen versus ChefSpec

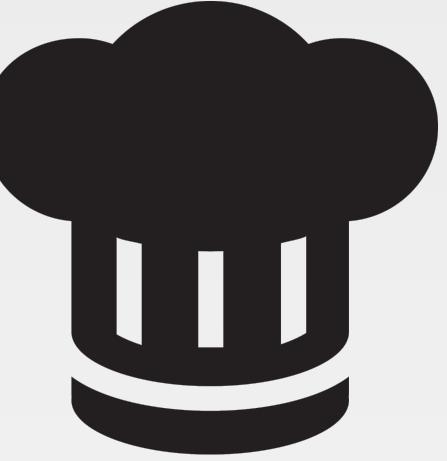
ChefSpec



Test Kitchen using InSpec



EXERCISE



Faster Feedback While Developing Cookbooks

*The faster the feedback from our tests, the more likely we are to run them.
The more likely we are to run them means they will catch more issues.*

Objective:

- Review and run the existing tests
- Identify the tests that we need to write
- Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

View the Spec Directory



```
> tree spec
```

```
spec
└── spec_helper.rb
└── unit
    └── recipes
        ├── configuration_spec.rb
        ├── default_spec.rb
        ├── install_spec.rb
        └── service_spec.rb
2 directories, 6 files
```

View the Test for the Default Recipe

~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end

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

View the Test for the Default Recipe

~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end

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

example groups

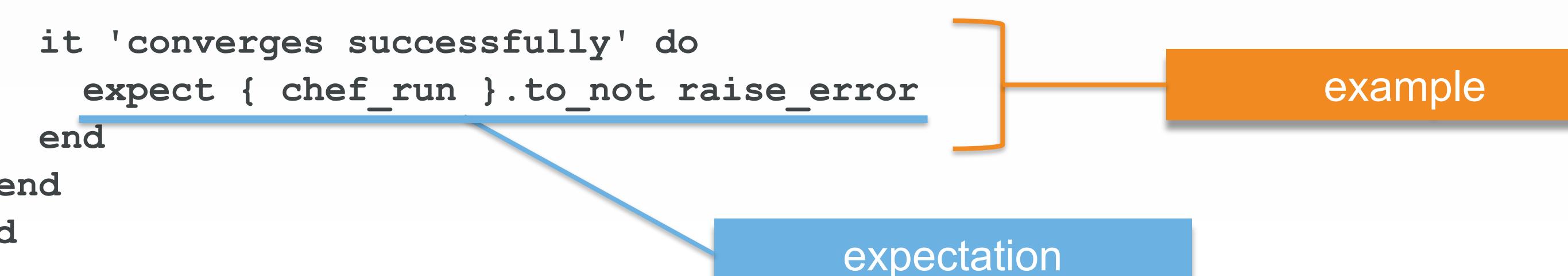
View the Test for the Default Recipe

~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end

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



View the Test for the Default Recipe

~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an Ubuntu 16.04' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '16.04')
      runner.converge(described_recipe)
    end

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

The diagram illustrates the components of the Chef test code:

- described recipe**: Points to the line `described_recipe`.
- Ruby Class**: Points to the line `ChefSpec::ServerRunner`.
- chef_run helper**: Points to the line `chef_run`.

Execute the Test for the Default Recipe



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

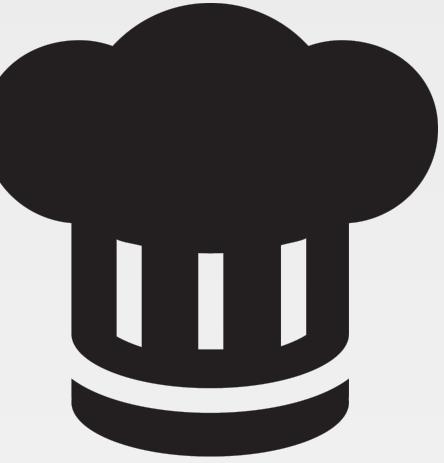
```
.
```

```
Finished in 0.44592 seconds (files took 4.35 seconds to load)
```

```
1 example, 0 failures
```

passing example

EXERCISE



Faster Feedback While Developing Cookbooks

*The faster the feedback from our tests, the more likely we are to run them.
The more likely we are to run them means they will catch more issues.*

Objective:

- Review and run the existing tests
- Identify the tests that we need to write
- Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

These are the Three Things to Test

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
# include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

Update the ChefSpec Platform

~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an CentOS 6.7' do +
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'centos', version: '6.7')+
      runner.converge(described_recipe)
    end

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

Create a Pending Test

```
~/apache/spec/unit/recipes/default_spec.rb
```

```
# ... START OF THE SPEC FILE ...

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

```
it 'includes the install recipe'
```

```
end
end
```

Execute the Tests to See the Pending Tests



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

.*

pending example

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

1) apache::default When all attributes are default, on an CentOS 6.7 includes the install recipe

Not yet implemented

./spec/unit/recipes/default_spec.rb:20

. . . OUTPUT CONTINUES ON NEXT SLIDE . . .

Execute the Tests to See the Pending Tests



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

.*

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

summary

1) apache::default When all attributes are default, on an CentOS 6.7 includes the install recipe

Not yet implemented

./spec/unit/recipes/default_spec.rb:20

... OUTPUT CONTINUES ON NEXT SLIDE ...

spec file : line number

View the Results to See the Pending Tests



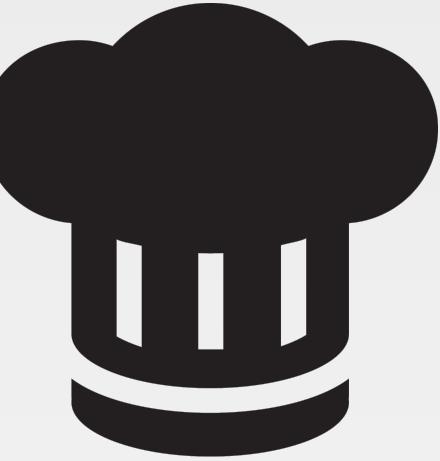
```
> chef exec rspec spec/unit/recipes/default_spec.rb
```

```
# ... OUTPUT CONTINUED FROM PREVIOUS SLIDE ...
```

```
Finished in 0.46457 seconds (files took 4.39 seconds to load)
```

```
2 examples, 0 failures, 1 pending
```

EXERCISE



Faster Feedback While Developing Cookbooks

*The faster the feedback from our tests, the more likely we are to run them.
The more likely we are to run them means they will catch more issues.*

Objective:

- ✓ Review and run the existing tests
- ✓ Identify the tests that we need to write
- Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

REFERENCE

ChefSpec Documentation



Find within the documentation examples of testing for `include_recipe`.

- Search the README
- Search through the 'examples' directory

<https://github.com/sethvargo/chefspec>

Write the Test that Verifies the Include Recipe

~/apache/spec/unit/recipes/default_spec.rb

```
# ... START OF THE SPEC FILE ...

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

it 'includes the install recipe' do
  expect(chef_run).to include_recipe('apache::install')
end

end
end
```

+

Execute the Tests to See the Failure



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

. F

Failures:

failing example

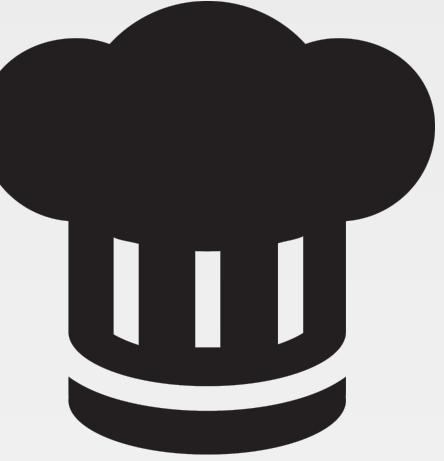
1) apache::default When all attributes are default, on an CentOS 6.7 includes the install recipe

```
Failure/Error: expect(chef_run).to
include_recipe('apache::install')
```

```
expected ["apache::default"] to include "apache::install"
```

```
# ./spec/unit/recipes/default_spec.rb:21:in `block (3 levels)
in <top (required)>'
```

EXERCISE



Faster Feedback While Developing Cookbooks

*The faster the feedback from our tests, the more likely we are to run them.
The more likely we are to run them means they will catch more issues.*

Objective:

- ✓ Review and run the existing tests
- ✓ Identify the tests that we need to write
- ✓ Write and execute the tests to identify the failure
- Fix the code and execute the tests to see success

Uncomment the Include Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
+  
include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

Execute the Tests to See it Pass

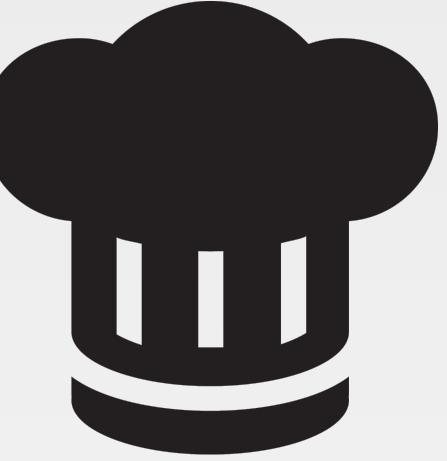


```
> chef exec rspec spec/unit/recipes/default_spec.rb
```

```
..
```

```
Finished in 0.67714 seconds (files took 4.26 seconds to load)  
2 examples, 0 failures
```

EXERCISE



Faster Feedback While Developing Cookbooks

*The faster the feedback from our tests, the more likely we are to run them.
The more likely we are to run them means they will catch more issues.*

Objective:

- ✓ Review and run the existing tests
- ✓ Identify the tests that we need to write
- ✓ Write and execute the tests to identify the failure
- ✓ Fix the code and execute the tests to see success

LAB



Continue with Mutation Testing

- Comment out the next line in the apache cookbook's default recipe
- Write the example with expectation that will generate a failure
- Verify that one example generates a failure
- Restore the code in the recipe
- Verify that all examples pass

❖ Repeat this series of steps for each line within the default recipe

Uncomment the Include Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
# include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

Write the Test that Verifies the Include Recipe

~/apache/spec/unit/recipes/default_spec.rb

```
# ... START OF THE SPEC FILE ...

it 'includes the install recipe' do
  expect(chef_run).to include_recipe('apache::install')
end

it 'includes the configuration recipe' do
  expect(chef_run).to include_recipe('apache::configuration')
end

end
end
```

Execute the Tests to See it Fail



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

.. F

Failures:

1) apache::default When all attributes are default, on an CentOS 6.7 includes the service recipe

Failure/Error: expect(chef_run).to include_recipe('apache::configuration')

expected ["apache::default", "apache::install"] to include "apache::configuration"

./spec/unit/recipes/default_spec.rb:25:in `block (3 levels)

Uncomment the Include Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

Execute the Tests to See it Pass



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

```
...
```

```
Finished in 0.97252 seconds (files took 4.33 seconds to load)  
3 examples, 0 failures
```

LAB



Continue with Mutation Testing

- ✓ Comment out the next line in the apache cookbook's default recipe
- ✓ Write the example with expectation that will generate a failure
- ✓ Verify that one example generates a failure
- ✓ Restore the code in the recipe
- ✓ Verify that all examples pass

❖ Repeat this series of steps for each line within the default recipe

DISCUSSION



Discussion

What functionality did you test in the integration tests?

What functionality did you test in these unit tests?

What do you see as the scope of unit testing versus integration testing?

What are the differences between a ChefSpec test and a InSpec test?

DISCUSSION



Q&A

What questions can we answer for you?

Morning

Introduction

Why Write Tests? Why is that Hard?

Writing a Test First

Refactoring Cookbooks with Tests

Afternoon

Faster Feedback with Unit Testing

Testing Resources in Recipes

Refactoring to Attributes

Refactoring to Multiple Platforms



CHEF™

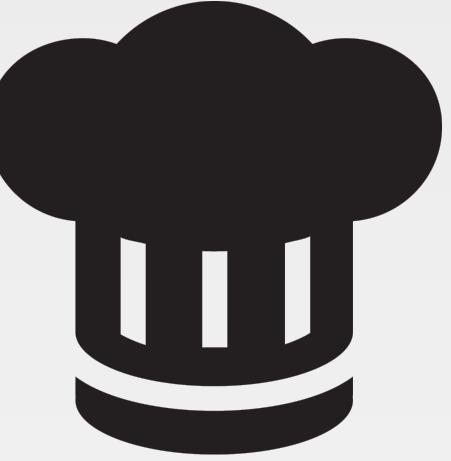
Testing Resources in Recipes

Objectives

After completing this module, you should be able to:

- Test resources within a recipe using ChefSpec

EXERCISE



Testing Remaining Resources

No resources left behind!

Objective:

- Write and execute tests for the Install recipe
- Verify the test validates the recipe

Generated Recipes Also Generate Specs



```
> tree spec
```

```
spec
└── spec_helper.rb
└── unit
    └── recipes
        ├── configuration_spec.rb
        ├── default_spec.rb
        ├── install_spec.rb
        └── service_spec.rb
```

Execute the Install Specification



```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
.
```

```
Finished in 0.46874 seconds (files took 4.24 seconds to load)
```

```
1 example, 0 failures
```

Update the ChefSpec Platform

~/apache/spec/unit/recipes/install_spec.rb

```
require 'spec_helper'

describe 'apache::install' do
  context 'When all attributes are default, on an CentOS 6.7' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'centos', version: '6.7')
      runner.converge(described_recipe)
    end

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

Add a Pending Test to Verify the Package

~/apache/spec/unit/recipes/install_spec.rb

```
# ... START OF THE SPEC FILE ...

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

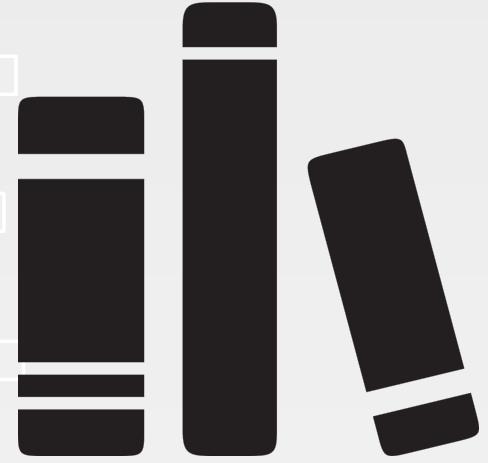
```
it 'installs the necessary package'
end
end
```

+

REFERENCE

ChefSpec Documentation

Find within the documentation examples of testing packages



<https://github.com/sethvargo/chefspec/tree/master/examples/package>

Write the Test to Verify the Package

~/apache/spec/unit/recipes/install_spec.rb

```
# ... START OF THE SPEC FILE ...

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

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

Write the Test to Verify the Package

~/apache/spec/unit/recipes/install_spec.rb

```
# ... START OF THE SPEC FILE ...
```

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

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

```
end
end
end
```

resource's action

resource

resource's name

Execute the Test to See it Pass



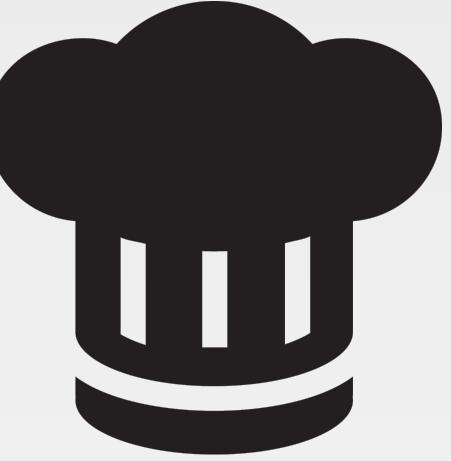
```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
..
```

```
Finished in 0.73662 seconds (files took 4.4 seconds to load)
```

```
2 examples, 0 failures
```

EXERCISE



Testing Remaining Resources

No resources left behind!

Objective:

- Write and execute tests for the Install recipe
- Verify the test validates the recipe

PROBLEM



It's Quiet. Too Quiet.

When a test passes immediately without having to write code (or if the code has already been written) it is time to be concerned. This is one of those moments we should ensure that the tests are working by mutating that code.

Comment Out the Resource

~/apache/recipes/install.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: install  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
# package 'httpd'
```

Execute the Test to See it Fail



```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

. F

Failures:

1) apache::install When all attributes are default, on an CentOS 6.7 installs the appropriate package

Failure/Error: expect(chef_run).to install_package('httpd')
expected "package[httpd]" with action :install to be in
Chef run. Other package resources:

Uncomment Out the Resource

~/apache/recipes/install.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: install  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
package 'httpd'
```

Execute the Test to See it Pass



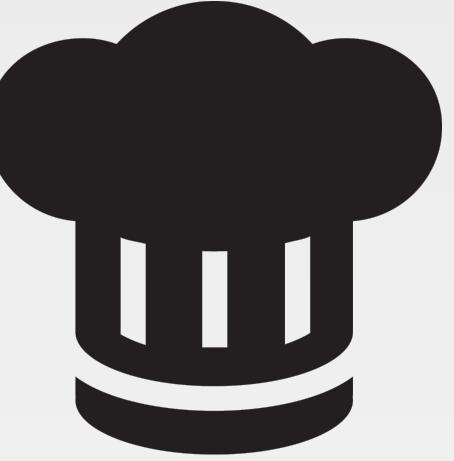
```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
..
```

```
Finished in 0.73662 seconds (files took 4.4 seconds to load)
```

```
2 examples, 0 failures
```

EXERCISE



Testing Remaining Resources

No resources left behind!

Objective:

- ✓ Write and execute tests for the Install recipe
- ✓ Verify the test validates the recipe

LAB



Test the Remaining Recipes

- Write a pending example
- Find the ChefSpec implementation
- Verify that the new example passes
- Mutate the recipe to generate a failure
- Restore the code in the recipe
- Verify that all examples pass

❖ Repeat this series of steps for the configuration recipe and service recipe

Update the ChefSpec Platform

~/apache/spec/unit/recipes/service_spec.rb

```
require 'spec_helper'

describe 'apache::install' do
  context 'When all attributes are default, on an CentOS 6.7' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'centos', version: '6.7')
      runner.converge(described_recipe)
    end

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

Write the Tests to Verify the Service

~/apache/spec/unit/recipes/service_spec.rb

```
# ... START OF THE SPEC FILE ...

it 'starts the necessary service' do
  expect(chef_run).to start_service('httpd')
end

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

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/service_spec.rb
```

```
...
```

```
Finished in 0.93685 seconds (files took 4.28 seconds to load)
```

```
3 examples, 0 failures
```

LAB

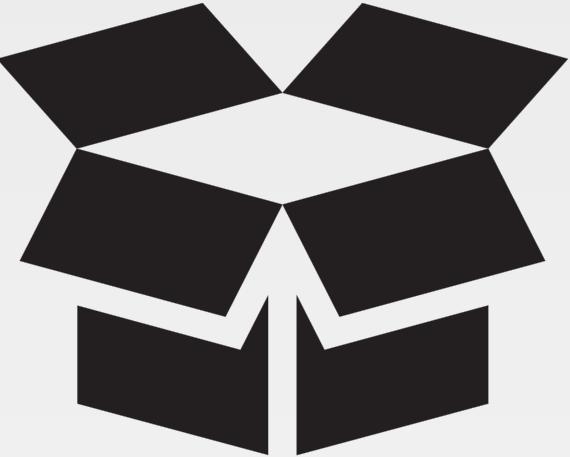


Test the Remaining Recipes

- ✓ Write a pending example
- ✓ Find the ChefSpec implementation
- ✓ Verify that the new example passes
- ✓ Mutate the recipe to generate a failure
- ✓ Restore the code in the recipe
- ✓ Verify that all examples pass

❖ Repeat this series of steps for the configuration recipe and service recipe

CONCEPT



rspec

When you run `rspec` without any paths it will automatically find and execute all the "`_spec.rb`" files within the '`spec`' directory.

Execute All the Tests in the Spec Directory



```
> chef exec rspec
```

```
.....
```

```
Finished in 2.08 seconds (files took 4.29 seconds to load)
```

```
8 examples, 0 failures
```

DISCUSSION

Discussion

What value does it bring to validate that the resources take the appropriate action?

DISCUSSION



Q&A

What questions can we answer for you?

Morning

Introduction

Why Write Tests? Why is that Hard?

Writing a Test First

Refactoring Cookbooks with Tests

Afternoon

Faster Feedback with Unit Testing

Testing Resources in Recipes

Refactoring to Attributes

Refactoring to Multiple Platforms



CHEF™

Testing While Refactoring to Attributes

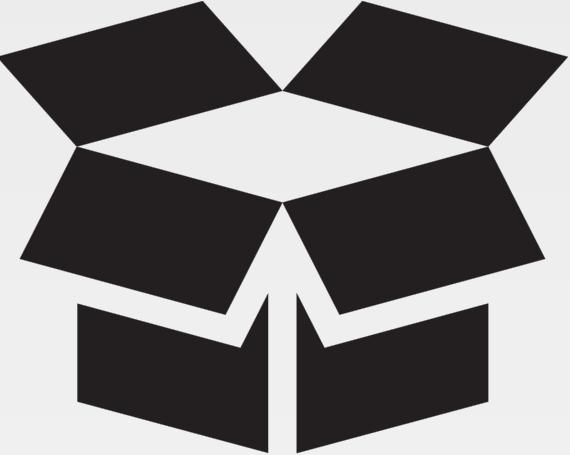


Objectives

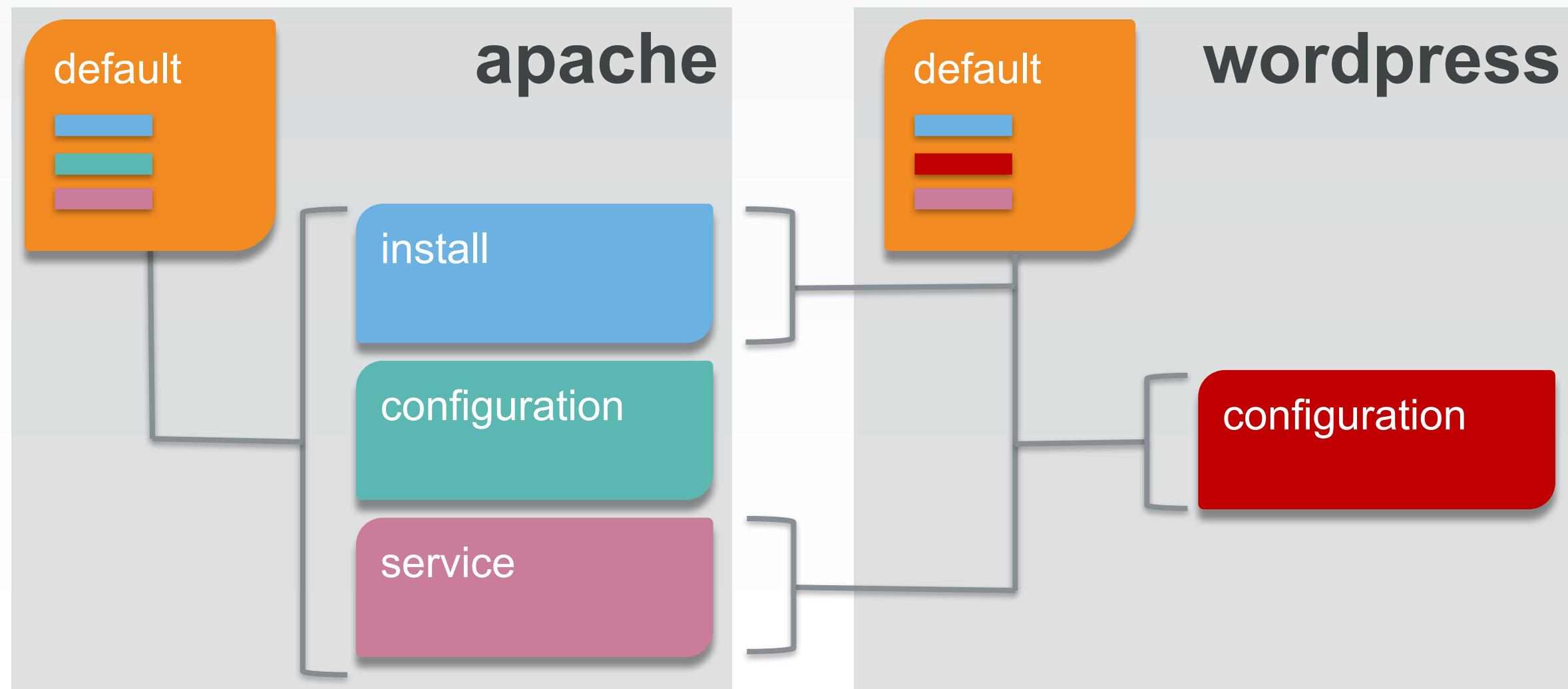
After completing this module, you should be able to:

- Refactor resources to use attributes
- Use Pry to explore the current state of execution
- Make changes to your recipes with confidence

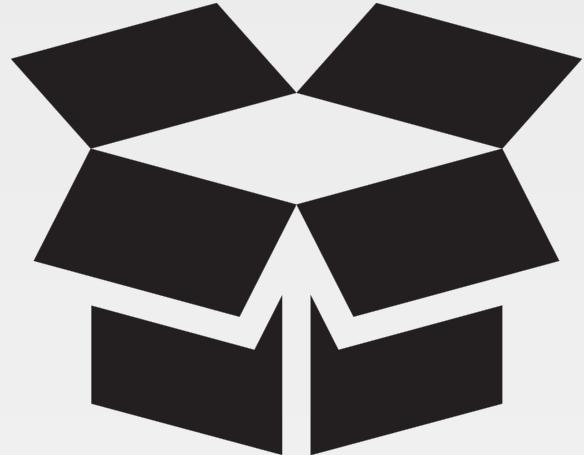
CONCEPT



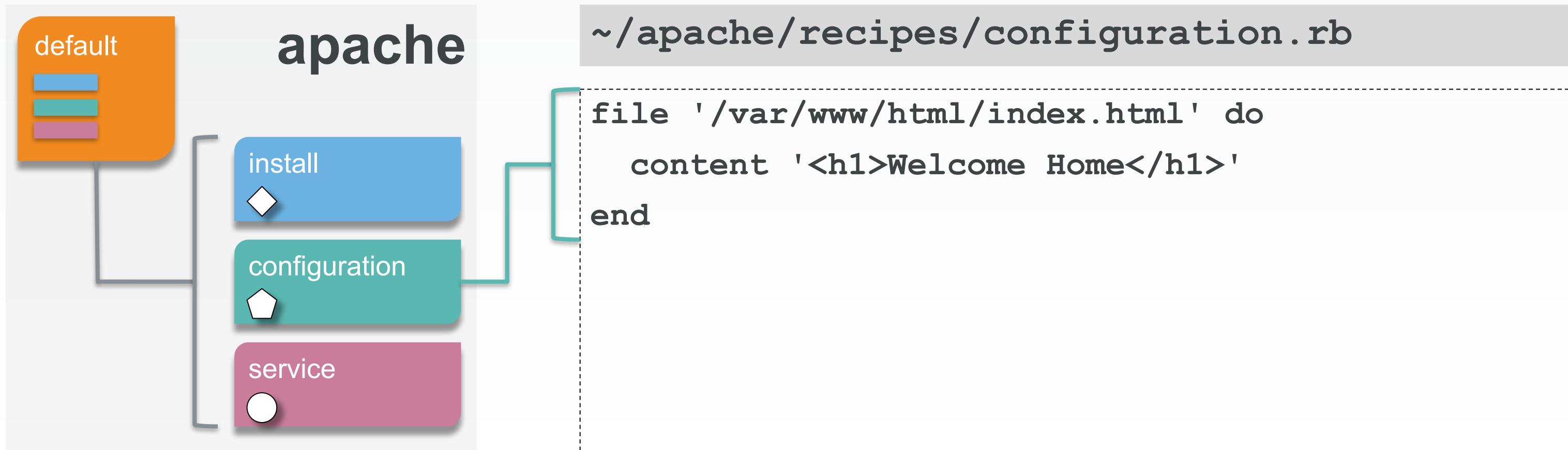
Modular Cookbook Recipes



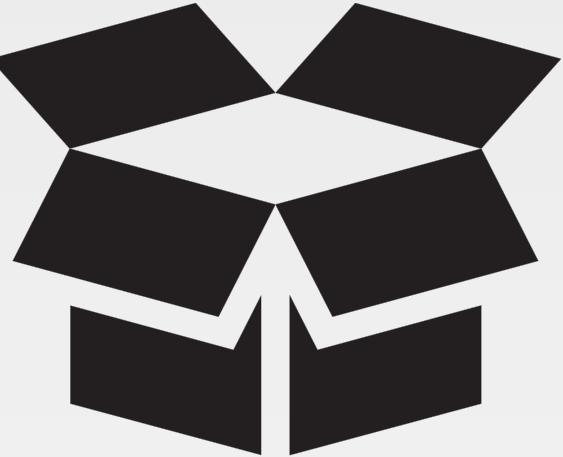
CONCEPT



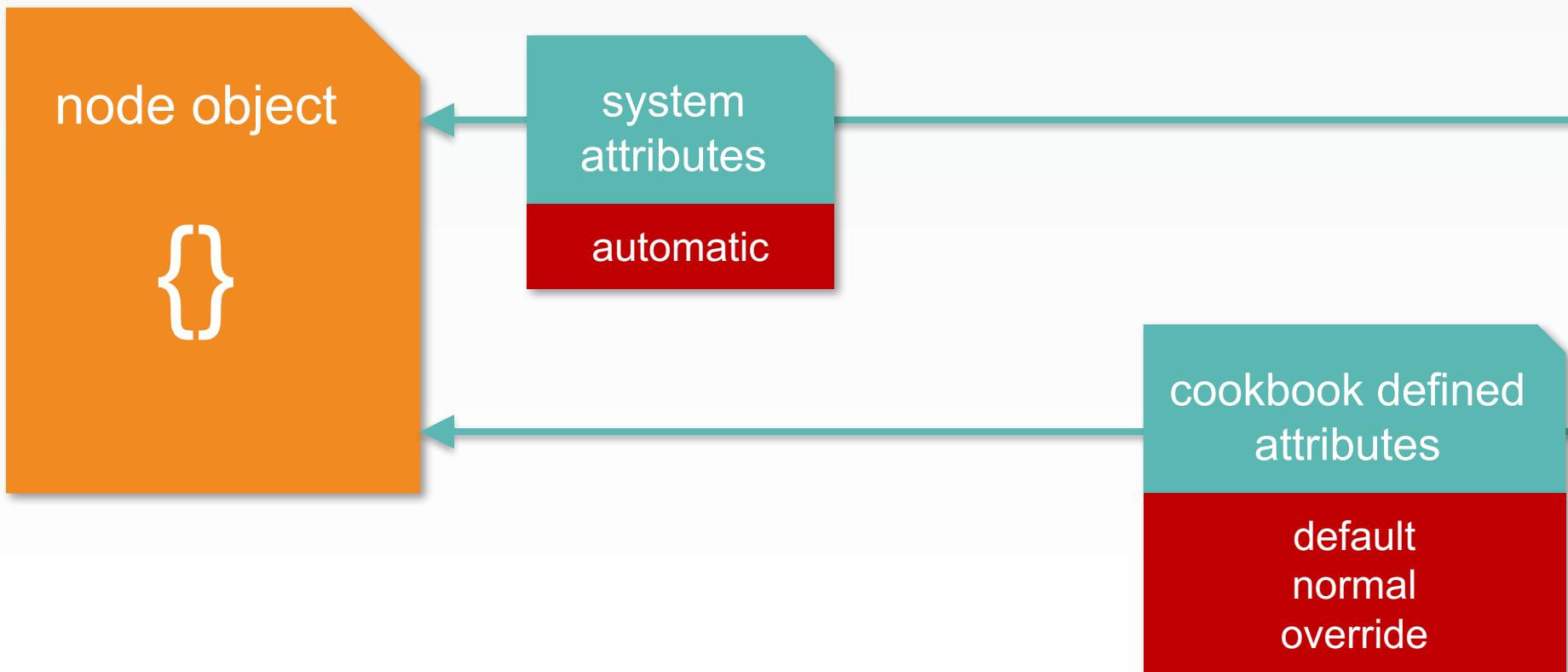
Modular Cookbook Recipes



CONCEPT



The Node Object



Apply the Run List

Build Node (ohai)

Synchronize Cookbooks

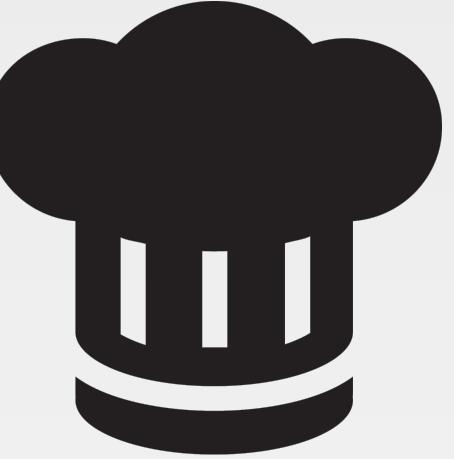
Load Cookbooks

Build Resource Collection

Converge

<https://docs.chef.io/attributes.html - attribute-precedence>

EXERCISE



Refactor to Use Attributes

Time to remove all the hard-coded values and make them attributes.

Objective:

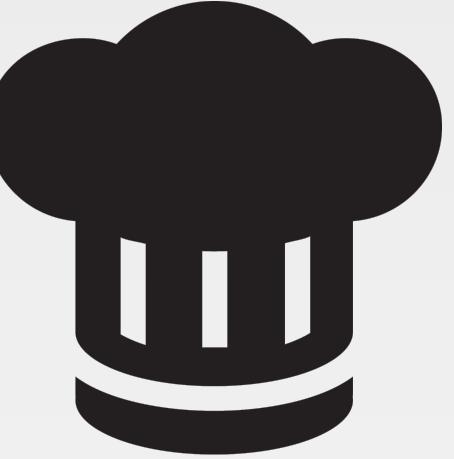
- Refactor the Install recipe to use a Node attribute
- Execute the tests and verify the tests fail
- Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

Replace the Value with a Node Attribute

~/apache/recipes/install.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: install  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
package node['apache']['package_name']
```

EXERCISE



Refactor to Use Attributes

A change means a chance for us to run the tests!

Objective:

- Refactor the Install recipe to use a Node attribute
- Execute the tests and verify the tests fail
- Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

Execute the Tests to See it Fail



```
> chef exec rspec
```

```
FFFFFF...
```

```
Failures:
```

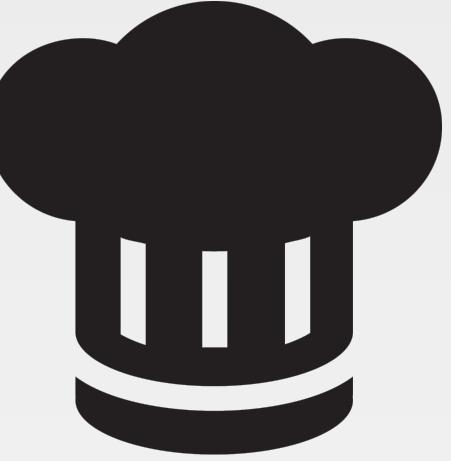
```
1) apache::default When all attributes are default, on an CentOS  
6.7 converges successfully
```

```
Failure/Error: expect { chef_run }.to_not raise_error
```

```
expected no Exception, got #<NoMethodError: undefined  
method `[]' for nil:NilClass> with backtrace:
```

```
# /tmp/d20171026-15641-  
1adgkog/cookbooks/apache/recipes/install.rb:6:in `from_file'
```

EXERCISE



Refactor to Use Attributes

We definitely broke it! Now, let's fix it.

Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

Ask Chef How to Generate an Attributes File



```
> chef generate attribute --help
```

Usage: **chef generate attribute** [path/to/cookbook] NAME [options]

-C, --copyright COPYRIGHT	Name of the copyright hol...
-m, --email EMAIL	Email address of the auth...
-a, --generator-arg KEY=VALUE	Use to set arbitrary ...
-I, --license LICENSE	all_rights, apache2, mit,...
-g GENERATOR_COOKBOOK_PATH, --generator-cookbook	Use GENERATOR_COOKBOOK_PA...

Use Chef to Generate a Default Attributes File



```
> chef generate attribute default
```

Compiling Cookbooks...

Recipe: code_generator::attribute

- * directory[/home/chef/apache/attributes] action create
 - create new directory /home/chef/apache/attributes
- * template[/home/chef/apache/attributes/default.rb] action create

- create new file /home/chef/apache/attributes/default.rb
 - update content in file /home/chef/apache/attributes/default.rb from none to e3b0c4

(diff output suppressed by config)

View the Attributes File Generated



```
> tree attributes
```

```
attributes
```

```
└── default.rb
```

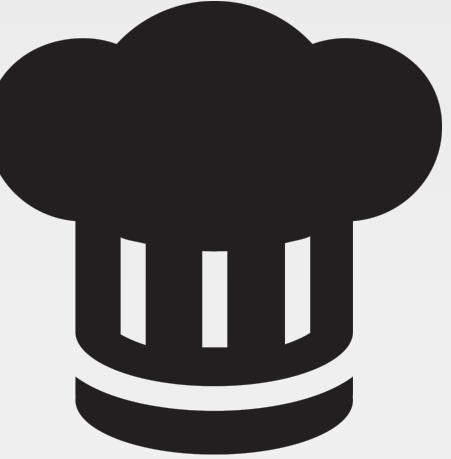
```
0 directories, 1 file
```

Add the Default Node Attribute

~/apache/attributes/default.rb

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

EXERCISE



Refactor to Use Attributes

The work is done. Let's hope it's the right work. Run the tests!

Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- ✓ Create the attributes file and add the Node attribute
- Execute the tests and verify the tests pass

Execute the Tests to See it Pass



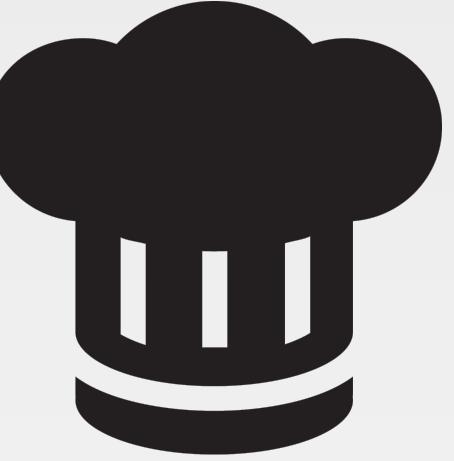
```
> chef exec rspec
```

```
.....
```

```
Finished in 2.28 seconds (files took 4.28 seconds to load)
```

```
9 examples, 0 failures
```

EXERCISE



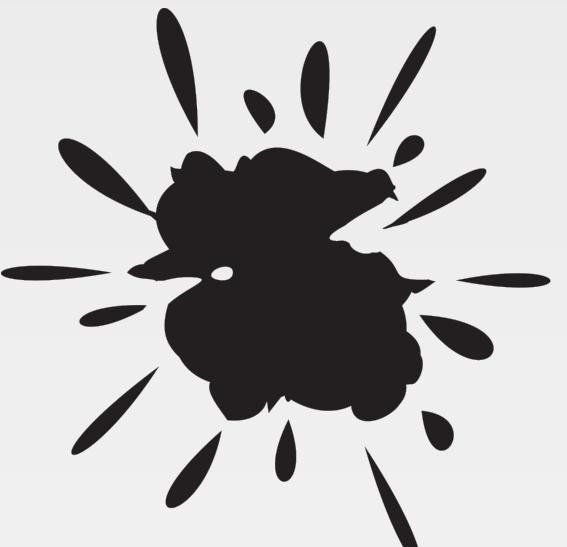
Refactor to Use Attributes

We made a change and we know it works!

Objective:

- ✓ Refactor the Install recipe to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- ✓ Create the attributes file and add the Node attribute
- ✓ Execute the tests and verify the tests pass

PROBLEM



What if We Made a Typo?

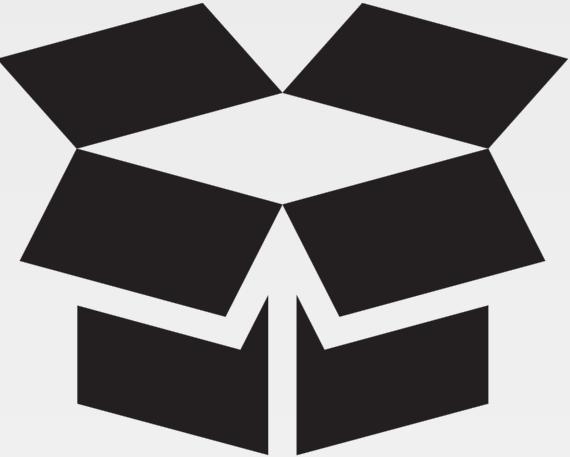
While implementing the node attribute what if made a mistake?

Typos Like This One Will Waste Time

~/apache/attributes/default.rb

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

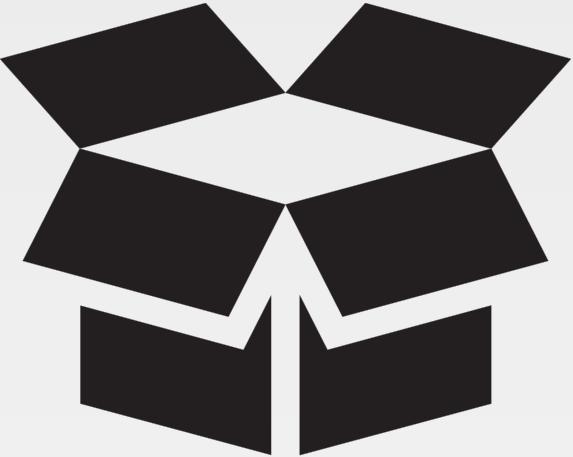
CONCEPT



Mental Model vs Actual Model

Faster feedback helps us build a greater mental model of the actual execution model. Tests that we define help strengthen it. However, tests are not very interactive as they are more like experiments. What we want is the ability to pause execution and look around.

CONCEPT

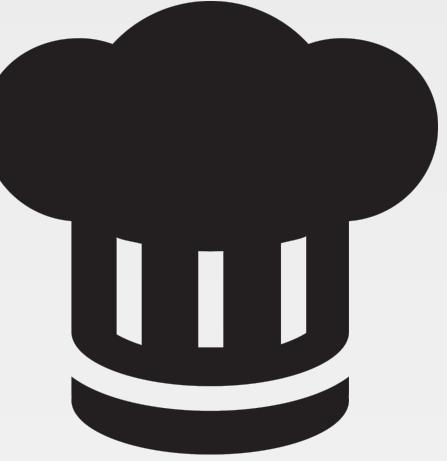


Pry a Debugger

Pry is a Ruby debugger that allows you to define break points. These breakpoints allow you to pause operation and interact with the current process being able to interrogate the current state of the system.

<http://pryrepl.org>

EXERCISE



Setup a Break Point

Time to make trouble for ourselves.

Objective:

- Mutate the code and add the breakpoint
- Execute the tests to cause the breakpoint to trigger
- Remove the breakpoint and restore the code

Create a Typo in the Defined Attribute

~/apache/attributes/default.rb

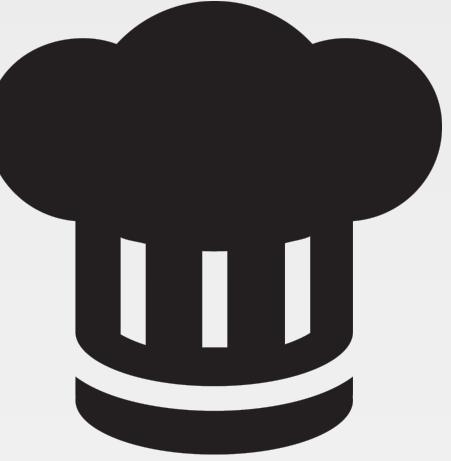
```
default['apche']['package_name'] = 'httpd'
```

Add a Break Point in the Recipe

~/apache/recipes/install.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: install  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
require 'pry'  
binding.pry  
  
package node['apache']['package_name']
```

EXERCISE



Setup a Break Point

Time to pry into the code and see what it is going on.

Objective:

- Mutate the code and add the breakpoint
- Execute the tests to cause the breakpoint to trigger
- Remove the breakpoint and restore the code

Execute the Test to Initiate Pry



```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
From: /tmp/d20171026-17430-19i8bee/cookbooks/apache/recipes/install.rb @ line
7 Chef::Mixin::FromFile#from_file:

2: # Cookbook Name:: apache
3: # Recipe:: install
4: #
5: # Copyright (c) 2017 The Authors, All Rights Reserved.
6: require 'pry'
=> 7: binding.pry
8:
9: package node['apache'] ['package_name']
# ... CONTINUES ON THE NEXT SLIDE ...
```

Pry Provides an Interactive Prompt



```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
# ... CONTINUED FROM THE PREVIOUS SLIDE ...

5: # Copyright (c) 2017 The Authors, All Rights Reserved.

6: require 'pry'

=> 7: binding.pry

8:

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

```
[1] pry(<Chef::Recipe>)>
```

Ask Pry for Help



```
[1] pry(#<Chef::Recipe>) > help
```

Help

`help` Show a list of commands or information about a specific command.

Context

`cd` Move into a new context (object or scope).

`find-method` Recursively search for a method within a class/module or the curr...

`ls` Show the list of vars and methods in the current scope.

`pry-backtrace` Show the backtrace for the pry session.

`raise-up` Raise an exception out of the current pry instance.

`reset` Reset the repl to a clean state.

To escape the help menu, type in q

Execute Any Code As You Would in a Recipe



```
[2] pry(#<Chef::Recipe>) > node['apache']
```

```
=> nil
```

Explore the Different Node Attributes



```
[3] pry(#<Chef::Recipe>) > node['apache']
```

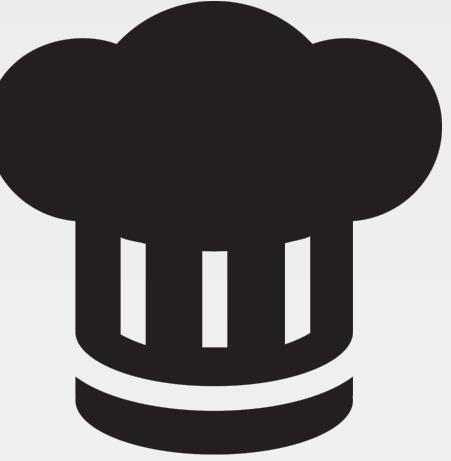
```
=> {"package_name"=>"httpd"}
```

Halt the Execution of the Test Immediately



```
[4] pry(#<Chef::Recipe>) > exit!
```

EXERCISE



Setup a Break Point

Time to pry into the code and see what it is going on.

Objective:

- Mutate the code and add the breakpoint
- Execute the tests to cause the breakpoint to trigger
- Remove the breakpoint and restore the code

Remove the Break Point from the Recipe

~/apache/recipes/install.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: install  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
require 'pry'  
binding.pry  
  
package node['apache']['package_name']
```

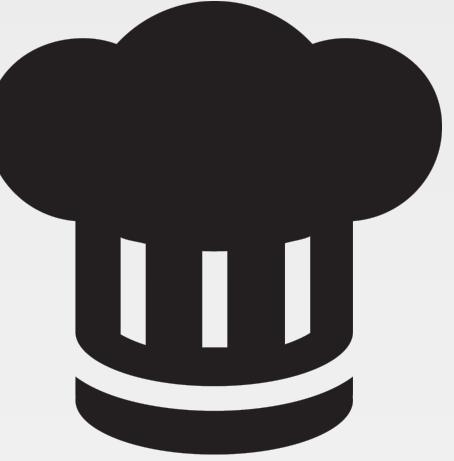
Fix the Change in the Attributes

~/apache/attributes/default.rb

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

+

EXERCISE



Setup a Break Point

Time to pry into the code and see what it is going on.

Objective:

- ✓ Mutate the code and add the breakpoint
- ✓ Execute the tests to cause the breakpoint to trigger
- ✓ Remove the breakpoint and restore the code

LAB



Refactor Remaining Resources

- Refactor the resource to use a Node attribute
- Execute the tests and verify the tests fail
- Add the new Node attribute
- Execute the tests and verify the tests pass

BONUS: Use pry to verify that the attribute has been set.

❖ Repeat this series of steps for the configuration recipe and service recipe

Update the Recipe to use the Node Attribute

~/apache/recipes/service.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: service  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
service node['apache'] ['service_name'] do +  
  action [:enable, :start]  
end
```

Execute the Tests to See it Fail



```
> chef exec rspec spec/unit/recipes/service_spec.rb
```

FFF

Failures:

1) apache::service When all attributes are default, on an CentOS 6.7
converges successfully

Failure/Error: expect { chef_run }.to_not raise_error

expected no Exception, got #<NoMethodError: undefined method `[]' for
nil:NilClass> with backtrace:

/tmp/d20171027-27872-
9rctn8/cookbooks/apache/recipes/service.rb:6:in `from_file'

Add the Default Node Attribute

~/apache/attributes/default.rb

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

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/service_spec.rb
```

```
...
```

```
Finished in 1.06 seconds (files took 4.33 seconds to load)  
3 examples, 0 failures
```

LAB



Refactor Remaining Resources

- ✓ Refactor the resource to use a Node attribute
- ✓ Execute the tests and verify the tests fail
- ✓ Add the new Node attribute
- ✓ Execute the tests and verify the tests pass

BONUS: Use pry to verify that the attribute has been set.

❖ Repeat this series of steps for the configuration recipe and service recipe

DISCUSSION



Discussion

What are the benefits of providing the package name and service name as node attributes?

What value does Pry provide to you as a Cookbook Developer?

DISCUSSION



Q&A

What questions can we answer for you?

Morning

Introduction

Why Write Tests? Why is that Hard?

Writing a Test First

Refactoring Cookbooks with Tests

Afternoon

Faster Feedback with Unit Testing

Testing Resources in Recipes

Refactoring to Attributes

Refactoring to Multiple Platforms



CHEF™

Testing While Refactoring to Multiple Platforms

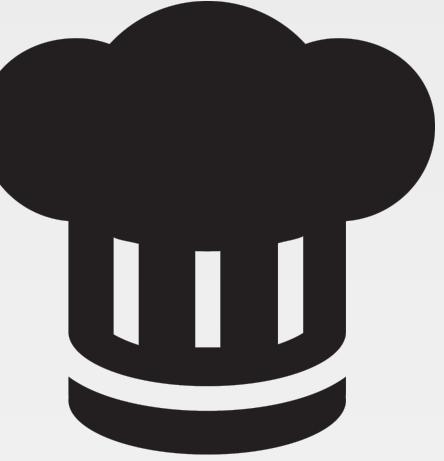


Objectives

After completing this module, you should be able to:

- Define expectations for multiple platforms
- Implement a cookbook that supports multiple platforms

EXERCISE



Node Platform in ChefSpec

What platform is the node when running a ChefSpec test?

How might you find out what is the platform?

Objective:

- Insert a break point, execute the tests, and determine the node's platform
- Remove the break point and transcend documentation

Then you will bridge the gap!



Add a Break Point to the Default Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
require 'pry'  
binding.pry  
include_recipe 'apache::install'  
include_recipe 'apache::configuration'  
include_recipe 'apache::service'
```

+

Execute the Tests to Initiate Pry



```
> chef exec rspec
```

```
From: /tmp/d20171027-28748-19ibu2o/cookbooks/apache/recipes/default.rb @ line
7 Chef::Mixin::FromFile#from_file:

 2: # Cookbook Name:: apache
 3: # Recipe:: default
 4: #
 5: # Copyright (c) 2017 The Authors, All Rights Reserved.
 6: require 'pry'
=> 7: binding.pry
 8: include_recipe 'apache::install'
 9: include_recipe 'apache::service'
```

Query the Node Object's Platform



```
[1] pry(#<Chef::Recipe>) > node['platform']
```

```
"chefspec"
```

REFERENCE



Fauxhai

ChefSpec by default uses a 'chefspec' platform. You can specify a platform from a list of platforms that are stored in a gem named 'Fauxhai'.

The gem contains static node objects for most major platforms and versions.

<https://github.com/customink/fauxhai/tree/master/lib/fauxhai/platforms>

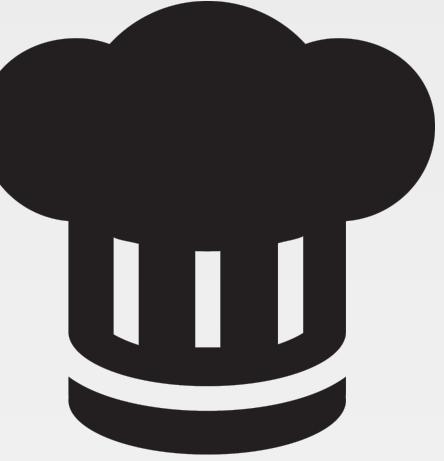
<https://github.com/customink/fauxhai>

Immediately Exit the Execution



```
[2] pry(#<Chef::Recipe>) > exit!
```

EXERCISE



Node Platform in ChefSpec

What platform is the node when running a ChefSpec test?

How might you find out what is the platform?

Objective:

- ✓ Insert a break point, execute the tests, and determine the node's platform
- ❑ Remove the break point and transcend documentation

A tidy life is a healthy life.

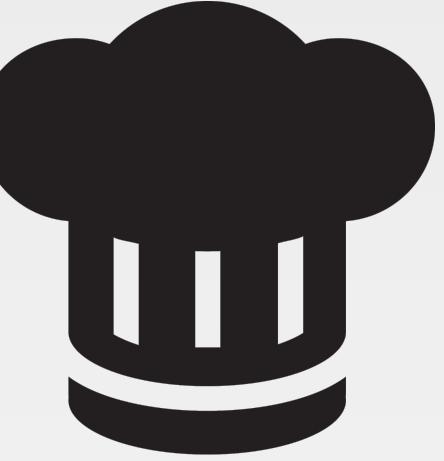


Remove the Break Point from the Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
require 'pry'  
binding.pry  
include_recipe 'apache::install'  
include_recipe 'apache::service'
```

EXERCISE



Node Platform in ChefSpec

What platform is the node when running a ChefSpec test?

How might you find out what is the platform?

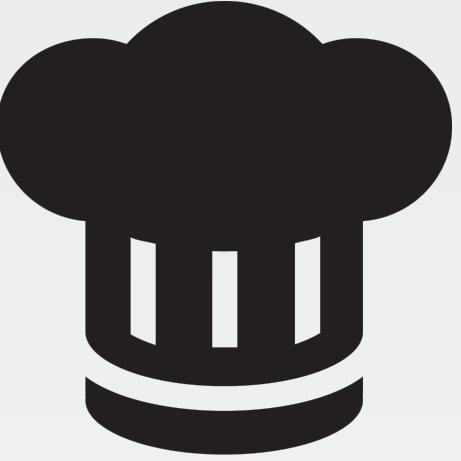
Objective:

- ✓ Insert a break point, execute the tests, and determine the node's platform
- ✓ Remove the break point and transcend documentation

Now I am ready to be
shaved.



EXERCISE



Support for CentOS & Ubuntu

The best of both worlds!

Objective:

- Write a test that verifies the Install recipe chooses the correct package on CentOS & Ubuntu
- Execute the tests and verify the tests fail
- Update the attribute to provide support for CentOS & Ubuntu
- Execute the tests and verify the tests pass

Update the Context to be Platform Specific



~/spec/unit/recipes/install_spec.rb

```
describe 'apache::install' do
  context 'When all attributes are default, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'centos', version: '6.7')
      runner.converge(described_recipe)
    end

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

    it 'installs the appropriate package' do
      expect(chef_run).to install_package('httpd')
    end
  end
# ... SPECIFICATION CONTINUES ON THE NEXT SLIDE ...
```

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
..
```

```
Finished in 1.35 seconds (files took 4.51 seconds to load)
2 examples, 0 failures
```

Add a Second Context for Another Platform



~/spec/unit/recipes/install_spec.rb

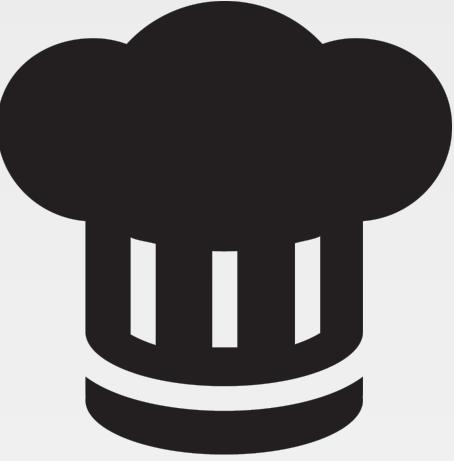
```
# ... CONTINUED FROM THE PREVIOUS SLIDE ...

context 'When all attributes are default, on Ubuntu' do
  let(:chef_run) do
    runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '14.04')
    runner.converge(described_recipe)
  end

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

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

EXERCISE



Support for CentOS & Ubuntu

Seems like a lot of duplication but its worth it for the test coverage.

Objective:

- ✓ Write a test that verifies the Install recipe chooses the correct package on CentOS & Ubuntu
- Execute the tests and verify the tests fail
- Update the attribute to provide support for CentOS & Ubuntu
- Execute the tests and verify the tests pass

Execute the Tests to See it Fail



```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

... F

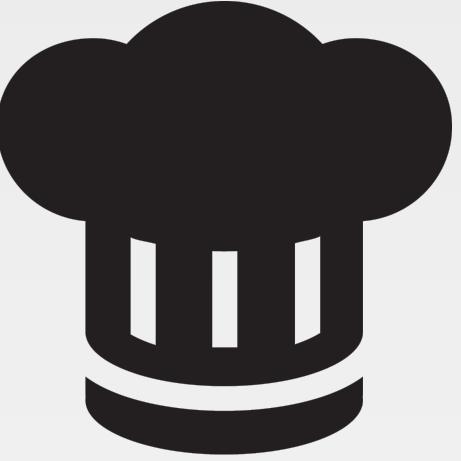
Failures:

1) apache::install When all attributes are default, on Ubuntu installs the appropriate 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]

EXERCISE



Support for CentOS & Ubuntu

Failure means we have work to do!

Objective:

- ✓ Write a test that verifies the Install recipe chooses the correct package on CentOS & Ubuntu
- ✓ Execute the tests and verify the tests fail
- Update the attribute to provide support for CentOS & Ubuntu
- Execute the tests and verify the tests pass

CONCEPT



Switching on Node Platform

To control the flow of execution we need to employ some Ruby conditional statements. Conditional statements allow us to alter this control flow. Because we have access to the power of Ruby we have many choices.

https://docs.chef.io/release/12-1/dsl_recipe.html

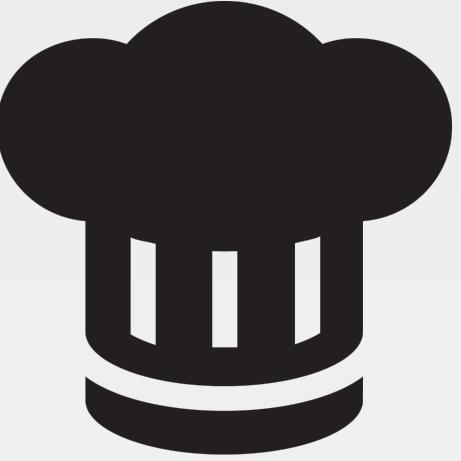
Update the Attributes to Support Platforms

~/apache/attributes/default.rb

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

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

EXERCISE



Support for CentOS & Ubuntu

This should do it!

Objective:

- ✓ Write a test that verifies the Install recipe chooses the correct package on CentOS & Ubuntu
- ✓ Execute the tests and verify the tests fail
- ✓ Update the attribute to provide support for CentOS & Ubuntu
- Execute the tests and verify the tests pass

Execute the Tests to See it Pass



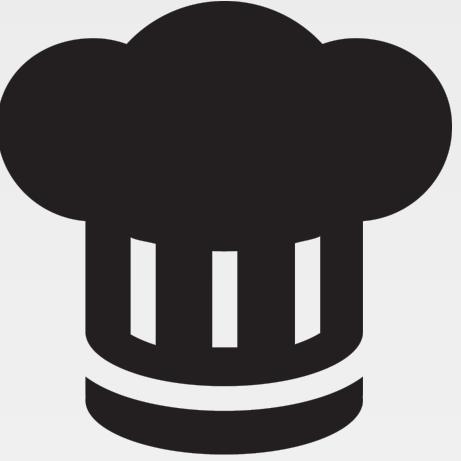
```
> chef exec rspec spec/unit/recipes/install_spec.rb
```

```
....
```

```
Finished in 1.35 seconds (files took 4.51 seconds to load)
```

```
4 examples, 0 failures
```

EXERCISE



Support for CentOS & Ubuntu

Woot! Multi-platform support for the installation!

Objective:

- ✓ Write a test that verifies the Install recipe chooses the correct package on CentOS & Ubuntu
- ✓ Execute the tests and verify the tests fail
- ✓ Update the attribute to provide support for CentOS & Ubuntu
- ✓ Execute the tests and verify the tests pass

LAB



Support for CentOS & Ubuntu

- ❑ Write a test that verifies the Service recipe chooses the service named 'httpd' on CentOS and 'apache2' on Ubuntu
- ❑ Execute the tests and verify the tests **fail**
- ❑ Update the attribute to choose the service name 'httpd' on CentOS and 'apache2' on Ubuntu
- ❑ Execute the tests and verify the tests **pass**

Update the Context to be Platform Specific

~/spec/unit/recipes/service_spec.rb

```
describe 'apache::service' do
  context 'When all attributes are default, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'centos', version: '6.7')
      runner.converge(described_recipe)
    end
    # ... it converges successfully ...

    it 'starts the appropriate service' do
      expect(chef_run).to start_service('httpd')
    end

    it 'enables the appropriate service' do
      expect(chef_run).to enable_service('httpd')
    end
  end
  # ... SPECIFICATION CONTINUES ON THE NEXT SLIDE ...
end
```

Add a Second Context for Another Platform



~/spec/unit/recipes/service_spec.rb

```
# ... CONTINUED FROM THE PREVIOUS SLIDE ...

context 'When all attributes are default, on Ubuntu' do
  let(:chef_run) do
    runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '14.04')
    runner.converge(described_recipe)
  end
  # ... it converges successfully ...

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

Execute the Tests to See it Fail



```
> chef exec rspec spec/unit/recipes/service_spec.rb
```

```
....FF
```

Failures:

1) apache::service When all attributes are default, on Ubuntu starts the appropriate service

Failure/Error: expect(chef_run).to start_service('apache2')
expected "service[apache2]" with action :start to be in
Chef run. Other service resources:

service[httpd]

Update the Attribute to Support Platforms

~/apache/attributes/default.rb

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

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/service_spec.rb
```

```
.....
```

```
Finished in 1.84 seconds (files took 4.22 seconds to load)
```

```
6 examples, 0 failures
```

LAB



Support for CentOS & Ubuntu

- ✓ Write a test that verifies the Service recipe chooses the service named 'httpd' on CentOS and 'apache2' on Ubuntu
- ✓ Execute the tests and verify the tests **fail**
- ✓ Update the attribute to choose the service name 'httpd' on CentOS and 'apache2' on Ubuntu
- ✓ Execute the tests and verify the tests **pass**

LAB



Support for CentOS & Ubuntu

- ❑ Write a test that verifies the file recipe chooses the same path (name)
'/var/www/html/index.html' on CentOS and on Ubuntu
- ❑ Execute the tests that verify the tests **pass**
- ❑ Update the attribute to choose the same path on CentOS and on Ubuntu
- ❑ Execute the tests that verify the tests **pass**
- ❑ Get nervous! Mutate the attributes file!
- ❑ Undo the entire attributes change and verify the tests **pass**

This is where it all comes together.



Update the Context to be Platform Specific

~/spec/unit/recipes/configuration_spec.rb

```
describe 'apache::configuration' do
  context 'When all attributes are default, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(platform: 'centos', version: '6.7')
      runner.converge(described_recipe)
    end
    # ... it converges successfully ...
  end
  it 'creates a default index html page' do
    expect(chef_run).to create_file('/var/www/html/index.html')
  end
end
```

.... SPECIFICATION CONTINUES ON THE NEXT SLIDE

Add a Second Context for Another Platform

~/spec/unit/recipes/configuration_spec.rb

```
# ... CONTINUED FROM THE PREVIOUS SLIDE ...

context 'When all attributes are default, on Ubuntu' do
  let(:chef_run) do
    runner = ChefSpec::ServerRunner.new(platform: 'ubuntu', version: '14.04')
    runner.converge(described_recipe)
  end
  # ... it converges successfully ...

  it 'creates a default index html page' do
    expect(chef_run).to create_file('/var/www/html/index.html')
  end
end
end
```

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/configuration_spec.rb
```

```
....
```

```
Finished in 1.84 seconds (files took 4.22 seconds to load)
```

```
4 examples, 0 failures
```

Update the Attribute to Support Platforms

~/apache/attributes/default.rb

```
case node['platform']
when 'ubuntu'

  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
  default['apache']['default_index_html'] = '/var/www/html/index.html'

else

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

end

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

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/configuration_spec.rb
```

```
....
```

```
Finished in 1.84 seconds (files took 4.22 seconds to load)
```

```
4 examples, 0 failures
```

Update the Attribute to Support Platforms

~/apache/attributes/default.rb

```
case node['platform']
when 'ubuntu'

  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
  default['apache']['default_index_html'] = '/var/www/html/index.html2'

else

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

end
```

Execute the Tests to See it Pass



```
> chef exec rspec spec/unit/recipes/configuration_spec.rb
```

```
...F
```

```
Finished in 1.84 seconds (files took 4.22 seconds to load)
```

```
4 examples, 1 failures
```

Update the Attribute to Support Platforms

~/apache/attributes/default.rb

```
case node['platform']
when 'ubuntu'

  default['apache']['package_name'] = 'apache2'
  default['apache']['service_name'] = 'apache2'
  default['apache']['default_index_html'] = '/var/www/html/index.html'

else

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

end

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

LAB



Support for CentOS & Ubuntu

- ✓ Write a test that verifies the file recipe chooses the same path (name)
'/var/www/html/index.html' on CentOS and on Ubuntu
- ✓ Execute the tests that verify the tests **pass**
- ✓ Update the attribute to choose the same path on CentOS and on Ubuntu
- ✓ Execute the tests that verify the tests **pass**
- ✓ Get nervous! Mutate the attributes file!
- ✓ Undo the entire attributes change and verify the tests **pass**

There is only one more
thing to do.



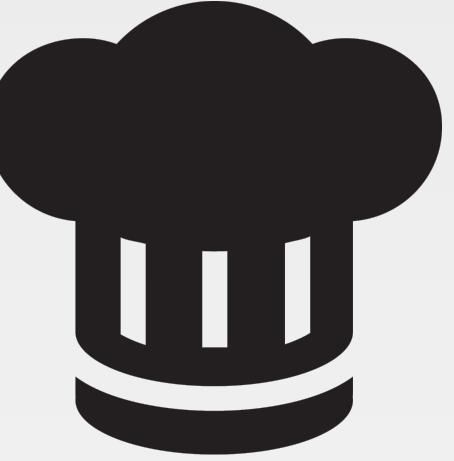
PROBLEM



What About an Integration Test

Remember that ChefSpec and Fauxhai are fake in-memory representations of a chef-client run. They are not equivalent to running the recipe on the specified platform.

EXERCISE



Integration Test with Ubuntu

This is where it all started.

Objective:

- Update the Kitchen Configuration to test on Ubuntu
- Execute the integration tests and verify that they pass

Add a New Platform to the Kitchen Configuration

~/apache/.kitchen.yml

```
---
driver:
  name: docker

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: centos-6.7
  - name: ubuntu-14.04
```

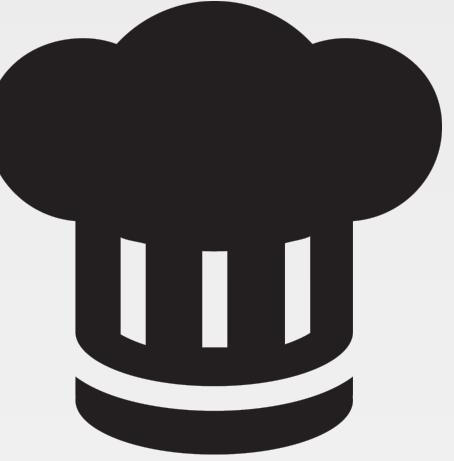
Verify the New Instance is Present



```
> kitchen list
```

Instance	Driver	Provisioner	Verifier	Transport	Last Action
default-centos-67	Docker	ChefZero	InSpec	Ssh	Set Up
default-ubuntu-1404	Docker	ChefZero	InSpec	Ssh	<Not Created>

EXERCISE



Integration Test with Ubuntu

Fingers crossed

Objective:

- ✓ Update the Kitchen Configuration to test on Ubuntu
- ❑ Execute the integration tests and verify that they pass

Execute the Tests for All Platforms



```
> kitchen test
```

```
-----> Starting Kitchen (v1.11.1)
-----> Cleaning up any prior instances of <default-centos-67>
-----> Destroying <default-centos-67>...
      Finished destroying <default-centos-67> (0m0.00s) .
-----> Testing <default-centos-67>
-----> Creating <default-centos-67>
      ...
      ...
```

EXERCISE



Integration Test with Ubuntu

Now I'm sure the cookbook works on two platforms and it would be easy to add a third ... or fourth.

Objective:

- ✓ Update the Kitchen Configuration to test on Ubuntu
- ✓ Execute the integration tests and verify that they pass

Your work has only begun



DISCUSSION



Discussion

What are the benefits and drawbacks of defining unit tests for multiple platforms?

What are the benefits and drawbacks of defining integration tests for multiple platforms?

When testing multiple platforms would you start with integration tests or unit tests?

DISCUSSION



Q&A

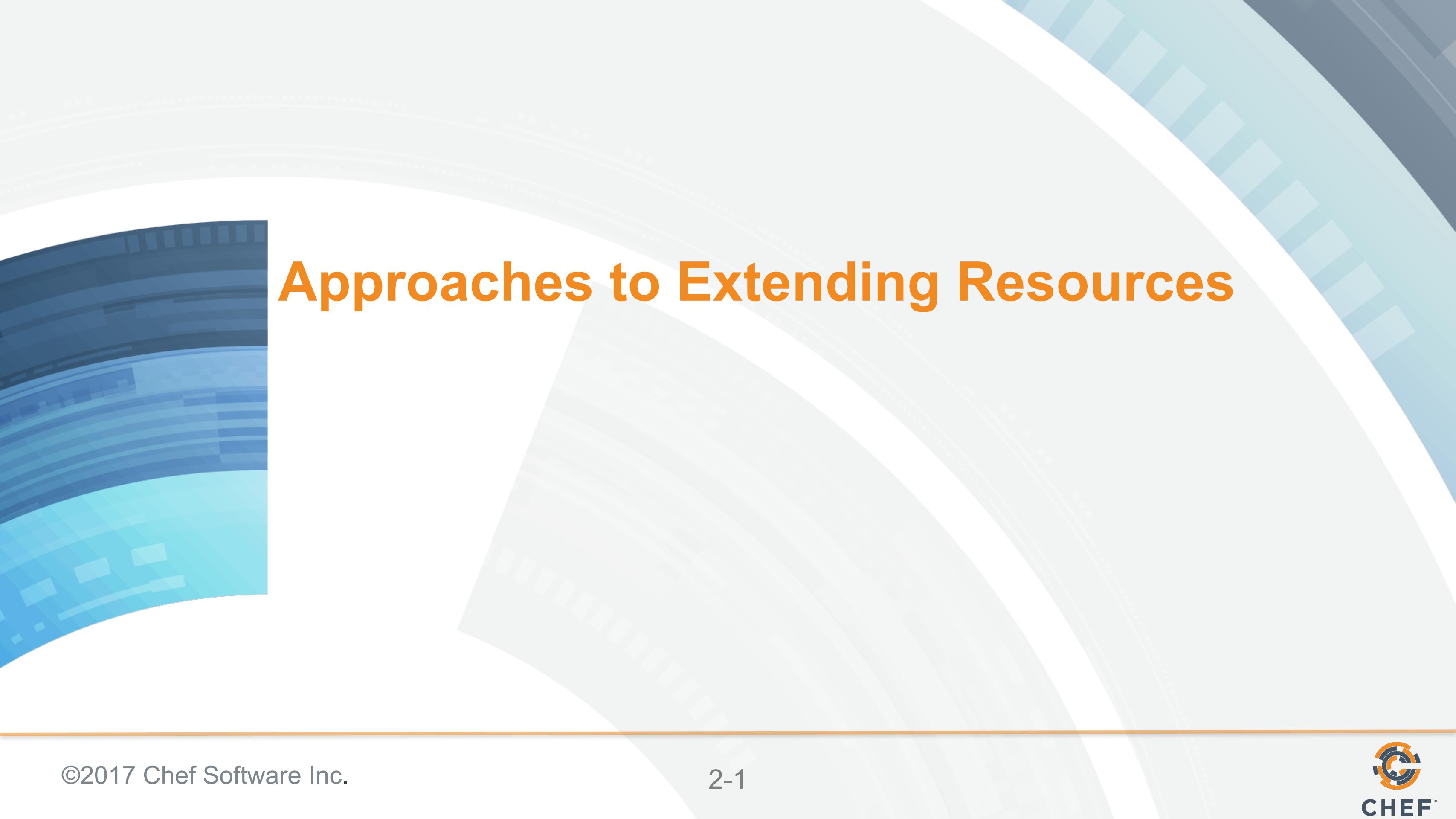
What questions can we answer for you?



You did it!



CHEF™



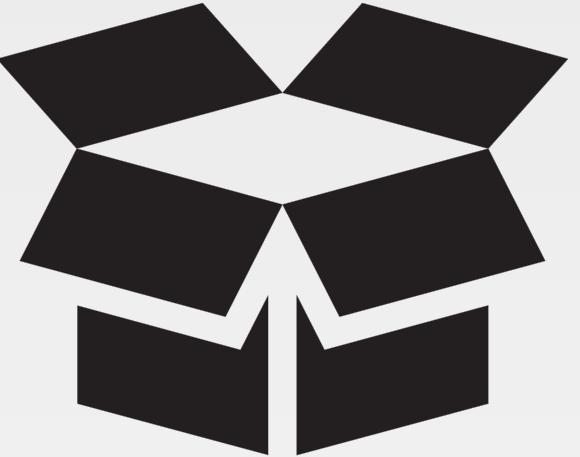
Approaches to Extending Resources

Objectives

After completing this module, you should be able to:

- Describe the difference between:
 - Custom Resources
 - Definitions
 - Heavy-Weight Resource-Providers
 - Light-Weight Resource-Providers

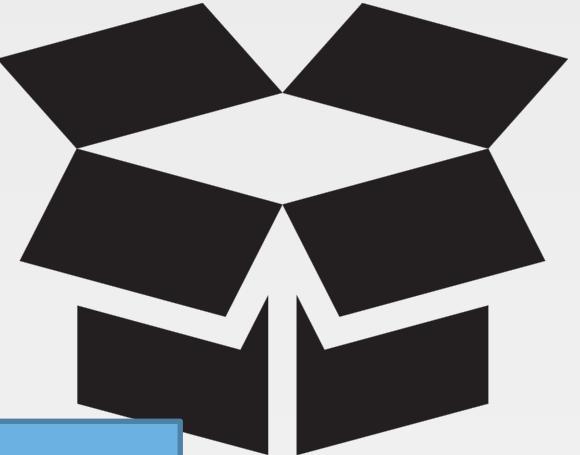
CONCEPT



Approaches to Extending Resources

- 1 Pure Ruby (Heavy-Weight Resource-Providers / HWRP)
- 2 Definitions
- 3 Light-Weight Resource-Providers (LWRP)
- 4 Custom Resources

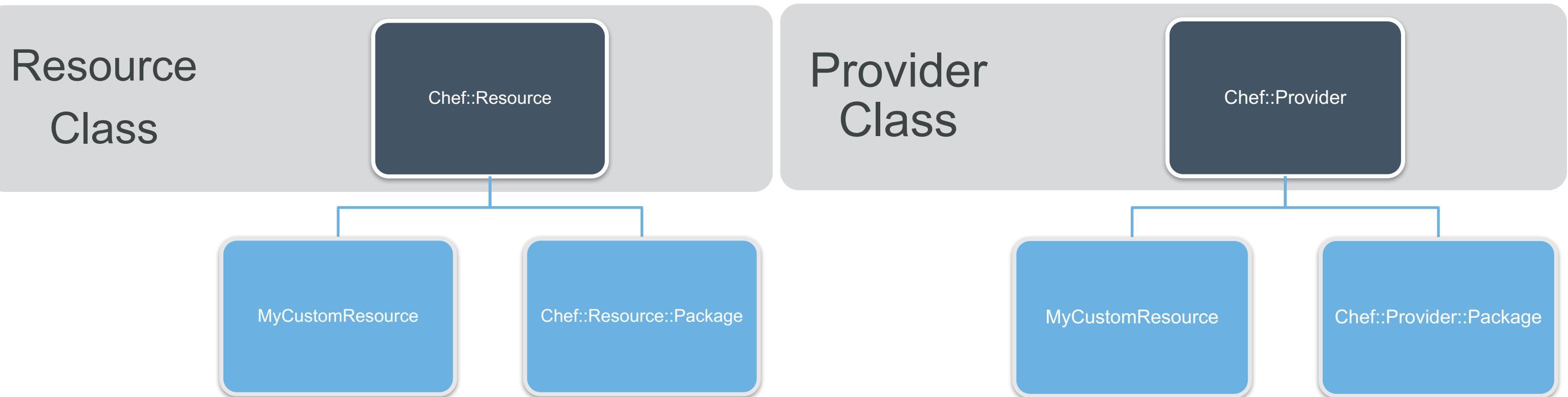
CONCEPT



1 Pure Ruby (Heavy-Weight Resource-Providers / HWRP)

- Description
- File and Folder Structure
- Implementation Language & Usage
- Benefits & Drawbacks

DESCRIPTION



Describes how the resource appears within the recipe
(e.g. resource name, properties, supported actions)

Describes how the resource behaves when it takes a
supported action on each supported platform

my_cookbook

`libraries/`

- `[my_custom_resource]_resource.rb`
- `[my_custom_resource]_provider.rb`

They are stored within the `libraries` folder in separate files for the resource and the provider. The file names are snake case representations of the class name stored within the file.

IMPLEMENTATION LANGUAGE - RESOURCE

`libraries/apache_vhost_resource.rb`

```
class Chef
  class Resource
    class ApacheVhost < Chef::Resource
      def initialize(name, run_context=nil)
        super
        @resource_name = :apache_vhost          # Defining the resource name
        @provider = Chef::Provider::ApacheVhost # Specifying which Provider to use
        @action = :create                      # Setting the default action
        @allowed_actions = [:create, :remove]   # Setting the list of actions
        # ... SETUP ANY DEFAULT VALUES HERE ...
      end

      def site_name(arg=nil)
        set_or_return(:site_name, arg, :kind_of => String)
      end
    end
  end
end
```

IMPLEMENTATION LANGUAGE - PROVIDER

`libraries/apache_vhost_provider.rb`

```
class Chef
  class Provider
    class ApacheVhost < Chef::Provider
      def load_current_resource
        @current_resource ||= Chef::Resource::ApacheVhost.new(new_resource.name)

        @current_resource.site_name(new_resource.site_name)
        # ... remaining properties defined in the resource
        @current_resource
      end

      def action_create
        # ... code that creates the resource on all supported platforms ...
      end
    end
  end
end
```

recipes/default.rb

```
apache_vhost 'welcome' do
  action :delete
end

apache_vhost 'users'

apache_vhost 'admins' do
  site_port 8080
end
```

BENEFITS & DRAWBACKS

- Available in some of the earliest versions of Chef
- Allows for extremely flexible and powerful resource implementations
- Requires knowledge of Ruby
- Requires knowledge of Object-Oriented Programming techniques

2 Definitions

DESCRIPTION

`recipes/admins_site.rb`

```
apache_vhost 'admins' do
  site_name 'admins'
end
```

`recipes/users_site.rb`

```
apache_vhost 'users' do
  site_name 'users'
end
```

`recipes/dogs_site.rb`

`...`

`definitions/apache_vhost.rb`

```
define :apache_vhost site_name: 'default' do
  directory ...
  template ...
  file ...
end
```

my_cookbook

```
definitions/
  • [my_definition_name].rb
```

They are stored within the definitions folder and often the name of the definition defines of the file.

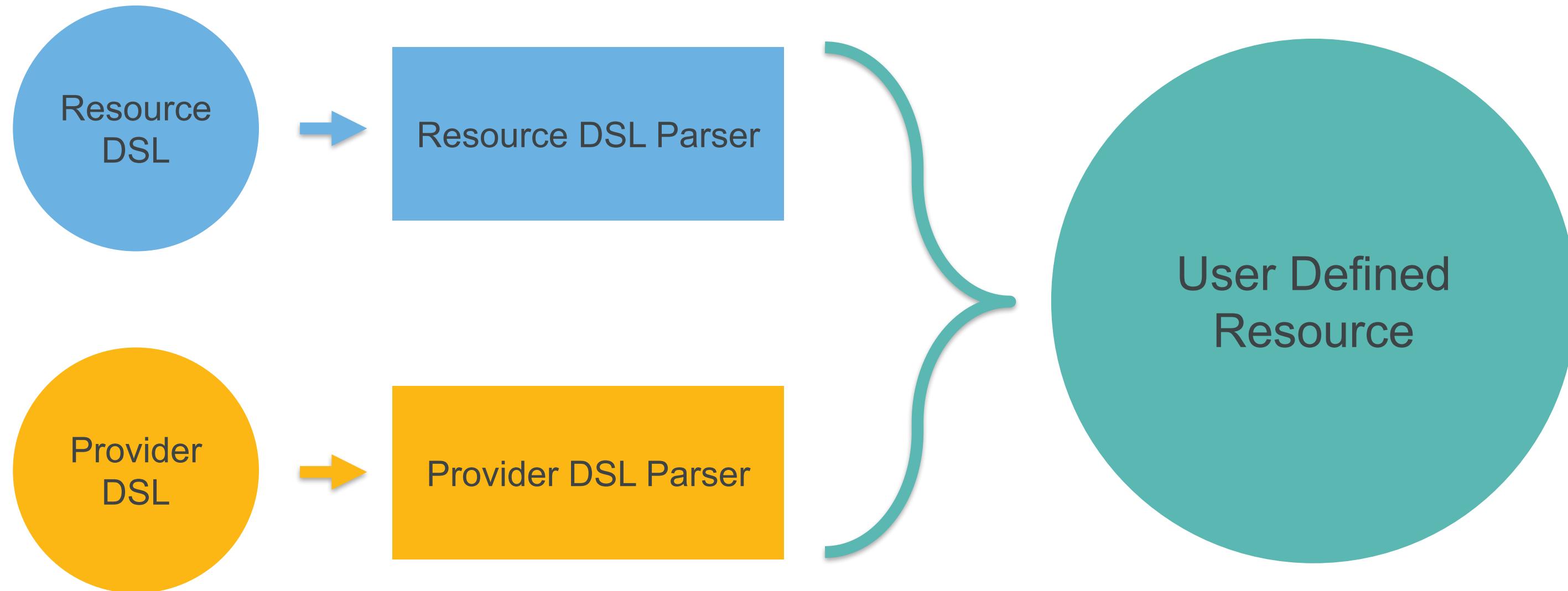
definitions/apache_vhost.rb

```
define :apache_vhost site_name: 'default', site_port: 80 do
  directory "/srv/apache/#{params[:site_name]}/html" do
    recursive true
    mode '0755'
  end

  templates "/srv/apache/#{params[:site_name]}/html" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/apache/#{params[:site_name]}/html", port: params[:site_port])
    mode '0755'
    notifies :restart, 'service[httpd]'
  end

  # ... remaining resources ...
end
```

- Available in some of the earliest versions of Chef
- Allows for code re-use within recipes
- Definition usage could be mistaken for a true resource
- Definitions do not support notifications (subscribes and notifies)



my_cookbook

```
resources/
  .. [my_resource_name].rb
providers/
  .. [my_resource_name].rb
```

An LWRP is defined in two separate files that share the same name. The resource definition is defined in the resources directory of the cookbook; the provider definition in the providers directory.

The cookbook name is combined with the file name to create the name of the resource.

IMPLEMENTATION LANGUAGE - RESOURCE

`resources/vhost.rb`

```
actions :create, :delete

default_action :create

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

IMPLEMENTATION LANGUAGE - PROVIDER

`providers/vhost.rb`

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

  templates "/srv/apache/#{new_resource.site_name}/html" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/apache/#{new_resource.site_name}/html",
              port: new_resource.site_port)
    mode '0755'
    notifies :restart, 'service[httpd]'
  end

  # ... remaining resources ...
end
```

recipes/default.rb

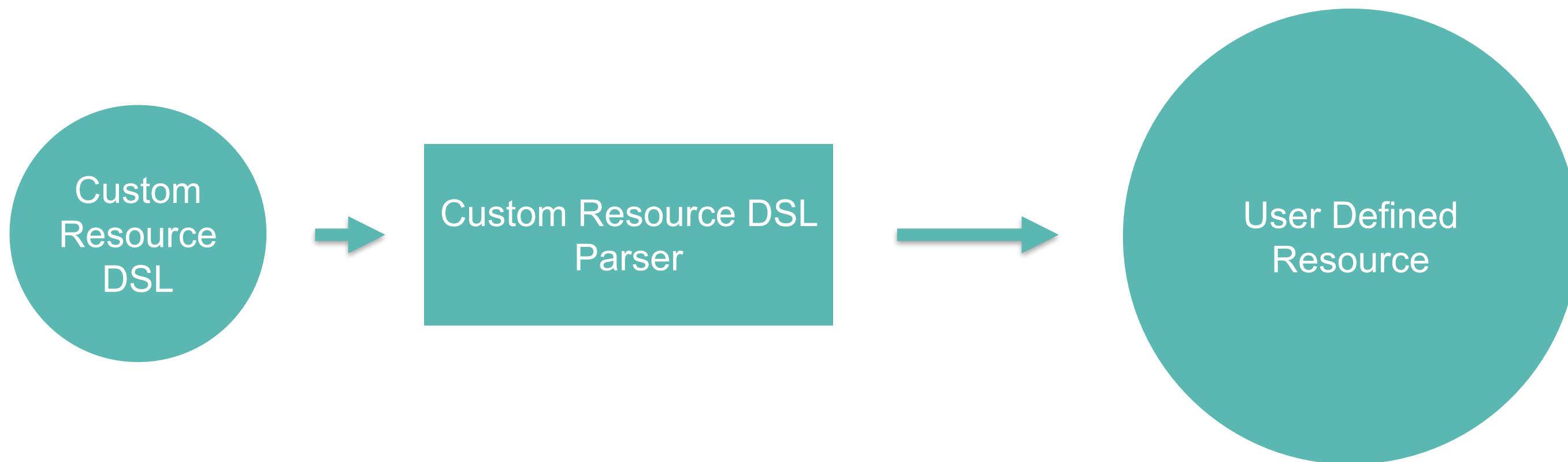
```
apache_vhost 'welcome' do
  action :delete
end

apache_vhost 'users'

apache_vhost 'admins' do
  site_port 8080
end
```

BENEFITS & DRAWBACKS

- Available in 0.7.12 version of Chef
- Allows for a real resource definition without understanding Ruby (vs. HWRP)
- Resource and provider implementation require learning a new DSL
- Complete resource definition is spread across two files



my_cookbook

```
resources/
  • [my_resource_name].rb
```

A custom resource is defined in a single file within the resources directory.

resources/vhost.rb

```
resource_name :apache_vhost

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

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

  # ... remaining resources ...
end

# ... remaining actions ...
```

recipes/default.rb

```
apache_vhost 'welcome' do
  action :delete
end

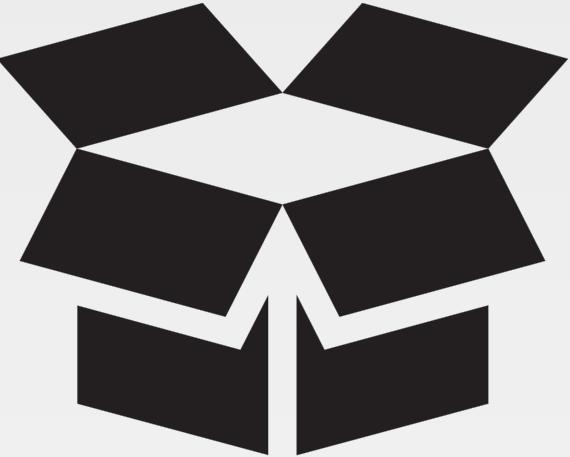
apache_vhost 'users'

apache_vhost 'admins' do
  site_port 8080
end
```

BENEFITS & DRAWBACKS

- Available in 12.5.0 version of Chef
- Allows for a real resource definition without understanding Ruby (vs. HWRP)
- Complete resource definition is defined in a single file (vs. LWRP)
- Custom resource implementation require learning a new DSL

CONCEPT



Approaches to Extending Resources

- 1 Pure Ruby (Heavy-Weight Resource-Providers / HWRP)
- 2 Definitions
- 3 Light-Weight Resource-Providers (LWRP)
- 4 Custom Resources

DISCUSSION



Discussion

Which approaches require you to define your solution in two separate files?

What are the limitations of choosing the Definitions approach?

What are some differences between LWRP and Custom Resources?

Given a Chef version prior to 12.5.0, which approach would you choose?

DISCUSSION



Q&A

What questions can we answer for you?



CHEF™

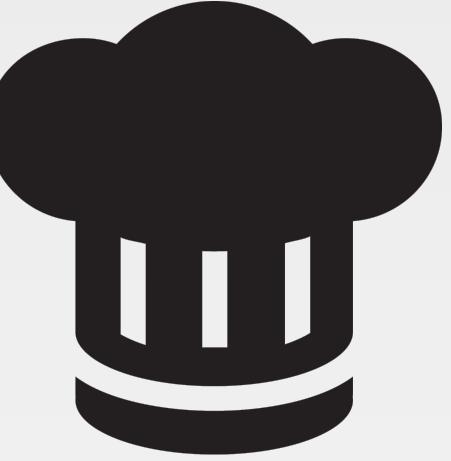
Why Use Custom Resources

Objectives

After completing this module, you should be able to:

- Determine when a Custom Resource would be beneficial for clarity and reusability

EXERCISE



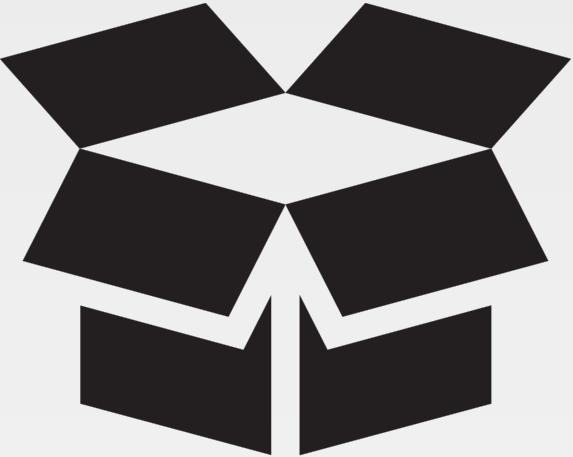
Evaluation Before Pursuit

Just because I can does not mean I should. It is important to implement solutions that are arguably better software design.

Objective:

- Define the judgment criteria
- Evaluate a code sample

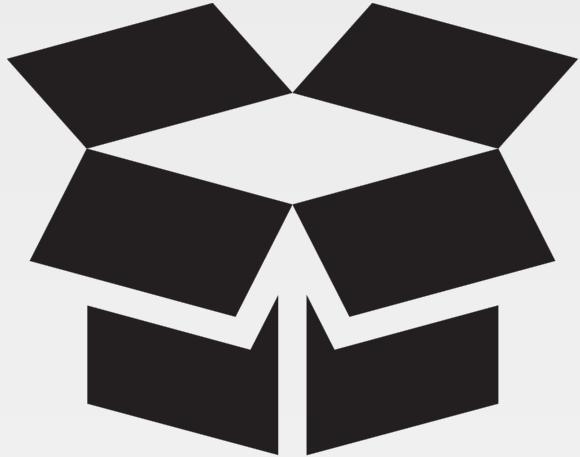
CONCEPT



Software Quality Standards

When defining resources within our recipes we are writing software. Software has a number of quality characteristics that have already been defined. ISO/IEC 9126 is an international standard for evaluation of software quality.

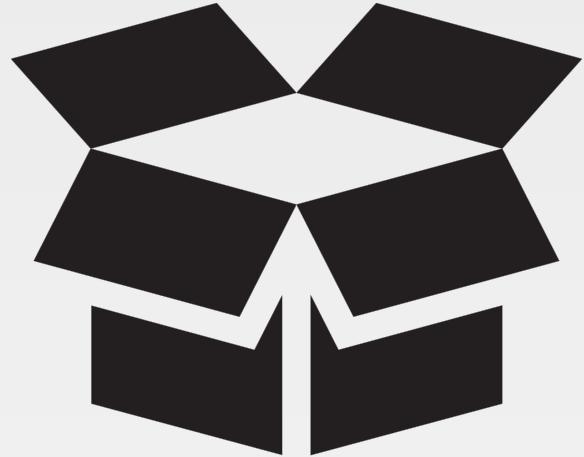
CONCEPT



Software Quality Standards

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

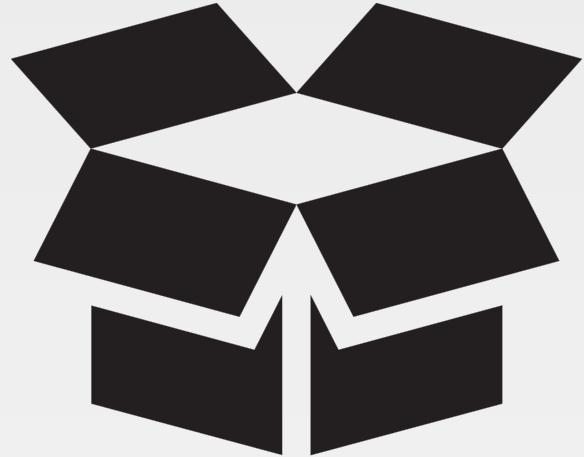
CONCEPT



Software Quality Standards

- **Functionality**
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Does the code accomplish what it is designed to accomplish?

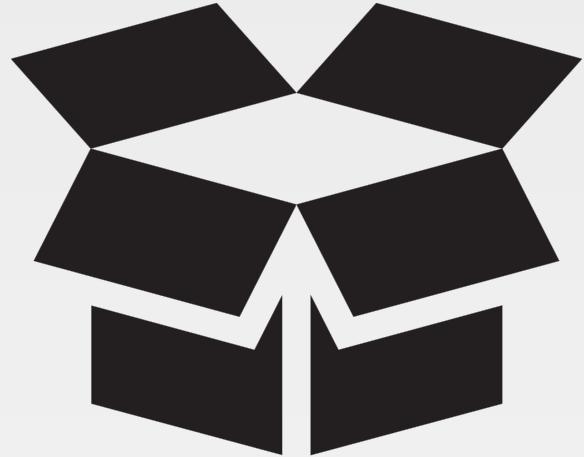


Software Quality Standards

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Is the solution able to withstand fault and recover from a failure?

CONCEPT

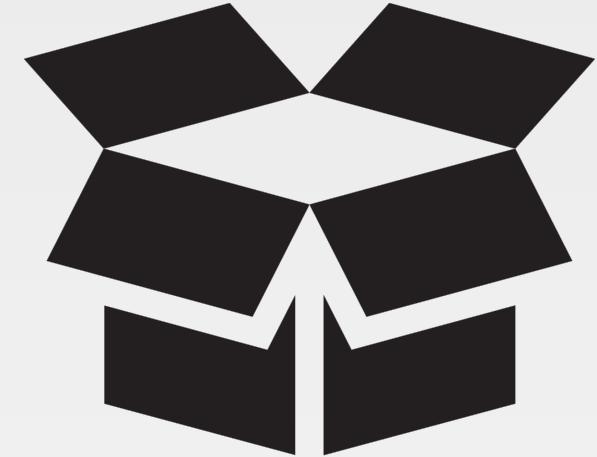


Software Quality Standards

- Functionality
- Reliability
- **Usability**
- Efficiency
- Maintainability
- Portability

Is the code easy to understand?
Is it easy to learn?

CONCEPT

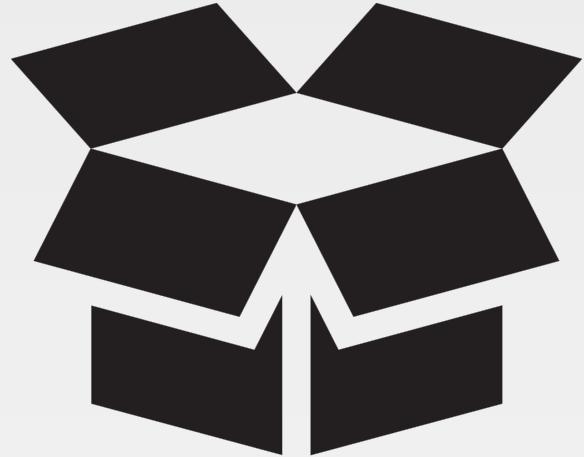


Software Quality Standards

- Functionality
- Reliability
- Usability
- **Efficiency**
- Maintainability
- Portability

Does the code consume too many physical resources when it executes (e.g. CPU, memory)?

CONCEPT

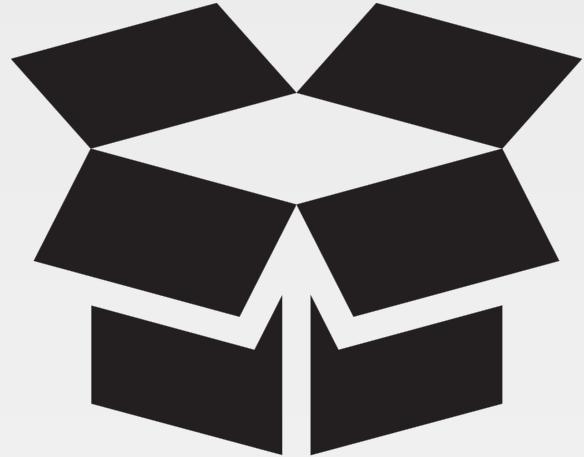


Software Quality Standards

- Functionality
- Reliability
- Usability
- Efficiency
- **Maintainability**
- Portability

Are you able to easily adapt the solution? Is it testable?

CONCEPT

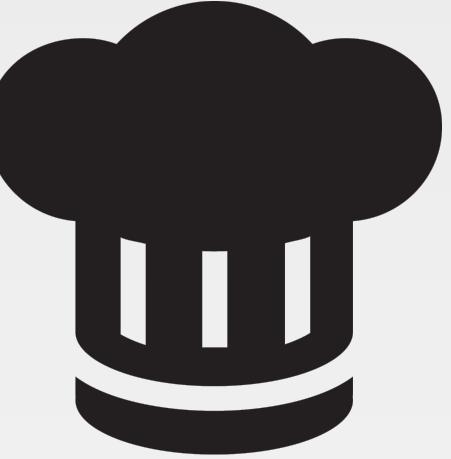


Software Quality Standards

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Can the software adapt to changes in its environment? Or changes to its requirements?

EXERCISE



Examine the Code Sample

With the criteria defined we can now examine code samples...

Objective:

- Define the judgment criteria
- Evaluate a code sample

Resource Implementation v Custom Resource

```
directory '/srv/apache/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'

variables(document_root:'/srv/apache/admins/html',
port: 8080)
  notifies :restart, 'service[httpd]'
end

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

```
apache_vhost 'admins' do
  site_port 8080
end
```

Functionality | Reliability | Usability | Efficiency | Maintainability | Portability

Does the code accomplish what it is designed to accomplish?

Resource Implementation v Custom Resource

```
directory '/srv/apache/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'

variables(document_root:'/srv/apache/admins/html',
port: 8080)
  notifies :restart, 'service[httpd]'
end

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

```
apache_vhost 'admins' do
  site_port 8080
end
```

Functionality | Reliability | Usability | Efficiency | Maintainability | Portability

Is the solution able to withstand fault and recover from a failure?

Resource Implementation v Custom Resource

```
directory '/srv/apache/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'

variables(document_root:'/srv/apache/admins/html',
port: 8080)
  notifies :restart, 'service[httpd]'
end

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

```
apache_vhost 'admins' do
  site_port 8080
end
```

Functionality | Reliability | **Usability** | Efficiency | Maintainability | Portability

Is the code easy to understand? Is it easy to learn?

Resource Implementation v Custom Resource

```
directory '/srv/apache/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'

variables(document_root: '/srv/apache/admins/html',
port: 8080)
  notifies :restart, 'service[httpd]'
end

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

```
apache_vhost 'admins' do
  site_port 8080
end
```

Functionality | Reliability | Usability | **Efficiency** | Maintainability | Portability

Does the code consume too many physical resources when it executes (e.g. CPU, memory)?

Resource Implementation v Custom Resource

```
directory '/srv/apache/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'

variables(document_root: '/srv/apache/admins/html',
port: 8080)
  notifies :restart, 'service[httpd]'
end

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

```
apache_vhost 'admins' do
  site_port 8080
end
```

Functionality | Reliability | Usability | Efficiency | **Maintainability** | Portability

Are you able to easily adapt the solution? Is it testable?

Resource Implementation v Custom Resource

```
directory '/srv/apache/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'

variables(document_root: '/srv/apache/admins/html',
port: 8080)
  notifies :restart, 'service[httpd]'
end

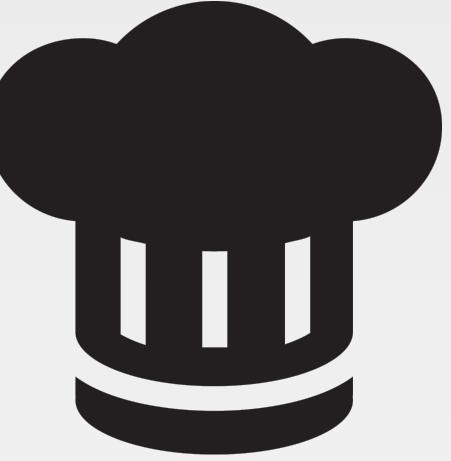
file '/srv/apache/admins/html/index.html' do
  content '<h1>Welcome admins!</h1>'
end
```

```
apache_vhost 'admins' do
  site_port 8080
end
```

Functionality | Reliability | Usability | Efficiency | Maintainability | **Portability**

Can the software adapt to changes in its environment? Or changes to its requirements?

EXERCISE



Evaluation Before Pursuit

There are many ways to critically evaluate code ... if these do not suit you or your team find the ones that do; talk about them and share them.

Objective:

- ✓ Define the judgment criteria
- ✓ Evaluate a code sample

DISCUSSION

Discussion

What value does reviewing code for functionality, reliability, usability, efficiency, maintainability, portability bring?

DISCUSSION



Q&A

What questions can we answer for you?



CHEF™

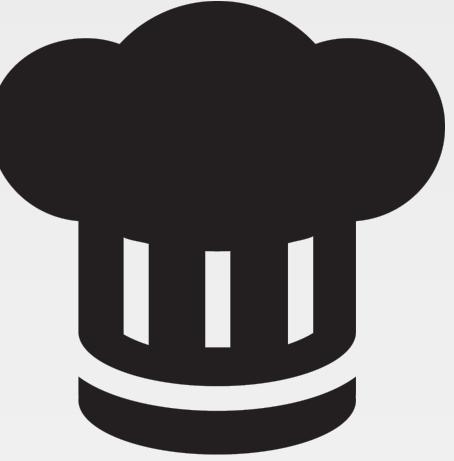
Creating a Custom Resource

Objectives

After completing this module, you should be able to:

- Create a custom resource file
- Define a custom resource action
- Extract Chef resources into a custom resource action implementation
- Create custom resource properties

EXERCISE



Review the Cookbook

We need to make sure everything works before we get started.

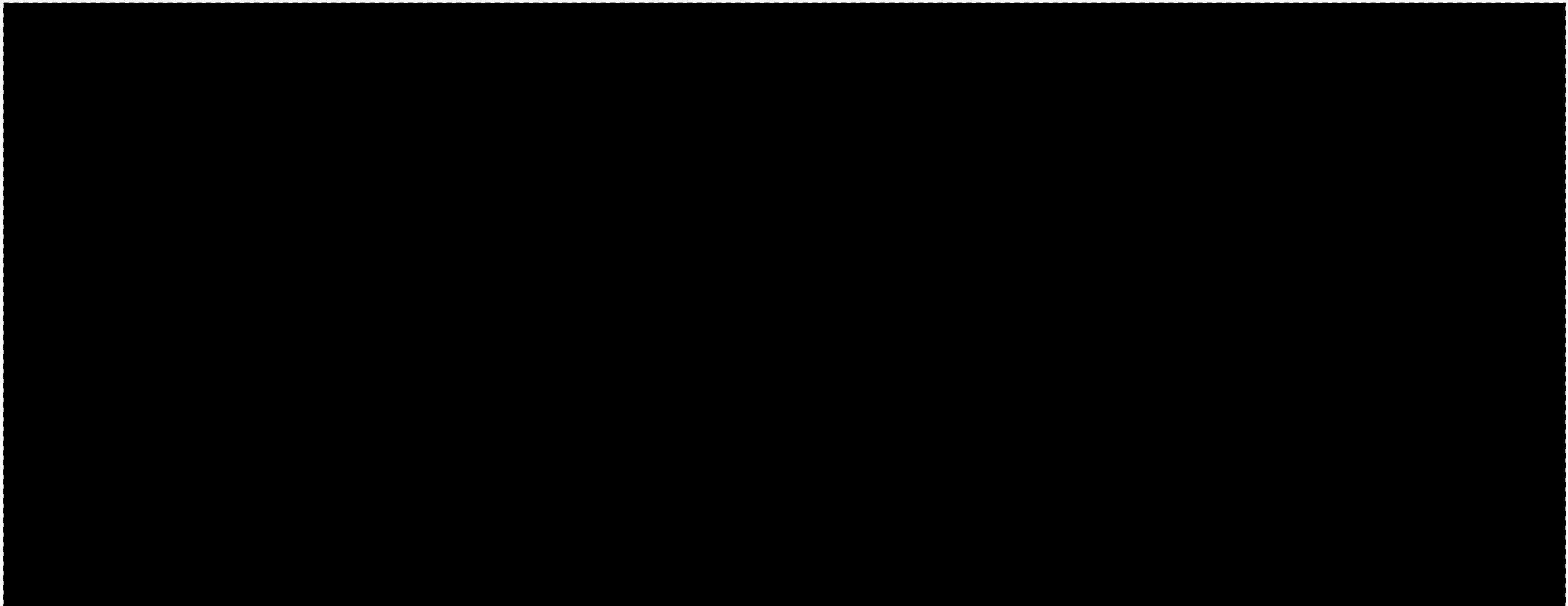
Objective:

- Checkout different branch of cookbook
- Review and execute the integration tests
- Review and execute the unit tests
- Review the default recipe

Move in the apache cookbook



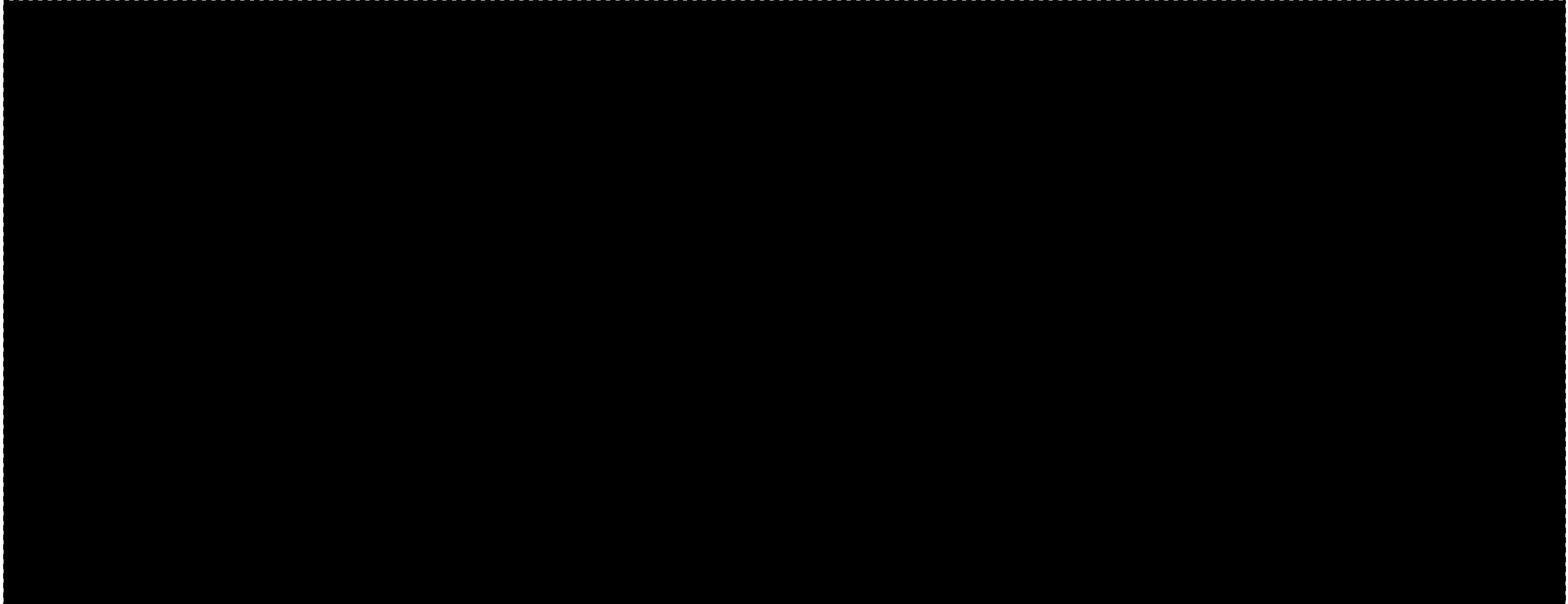
```
> cd ~/apache
```



Add the Changed Files to Staging



```
> git add .
```



Commit the Staged Changes



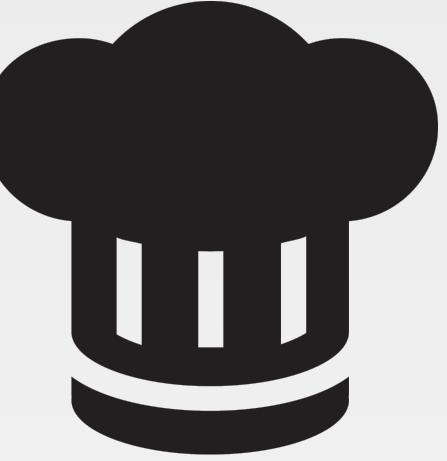
```
> git commit -m "Saving TDD work"
```

Change Git Branches



```
> git checkout extending-cookbook
```

EXERCISE



Review the Cookbook

We need to make sure everything works before we get started.

Objective:

- Checkout different branch of cookbook
- Review and execute the integration tests
- Review and execute the unit tests
- Review the default recipe

Reviewing the Existing Integration Tests

~/apache/test/smoke/default/default_test.rb

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

end

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

end
```

Executing the Existing Integration Tests



```
> kitchen verify
```

```
Target: ssh://vagrant@127.0.0.1:2222
```

```
✓ Command curl http://localhost stdout should match /Welcome  
home/
```

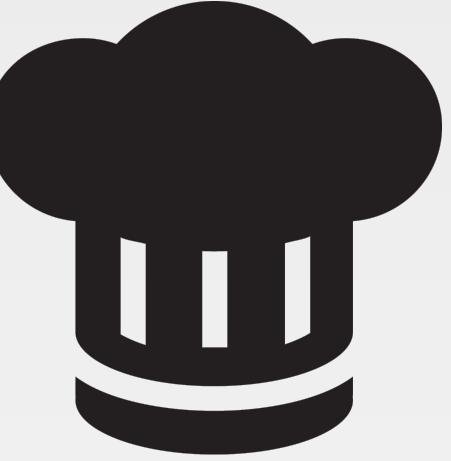
```
✓ Command curl http://localhost:8080 stdout should match  
/Welcome admins/
```

```
Summary: 2 successful, 0 failures, 0 skipped
```

```
Finished verifying <default-centos-67> (0m0.74s) .
```

```
-----> Kitchen is finished. (0m7.37s)
```

EXERCISE



Review the Cookbook

We need to make sure everything works before we get started.

Objective:

- Checkout different branch of cookbook
- Review and execute the integration tests
- Review and execute the unit tests
- Review the default recipe

Reviewing the Existing Unit Tests

~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an unspecified platform' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new
      runner.converge(described_recipe)
    end

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

# ... CONTINUES ON THE NEXT SLIDE ...
```

Reviewing the Existing Unit Tests

~/apache/spec/unit/recipes/default_spec.rb

```
# ... CONTINUES FROM THE PREVIOUS SLIDE ...
```

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

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

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

```
# ... CONTINUES ON THE NEXT SLIDE ...
```

Reviewing the Existing Unit Tests

~/apache/spec/unit/recipes/default_spec.rb

```
# ... CONTINUES FROM THE PREVIOUS SLIDE ...
```

```
describe 'for the default site' do
  it 'writes out a new home page' do
    expect(chef_run).to render_file('/var/www/html/index.html').with_content('<h1>Welcome
home!</h1>')
  end
end
```

```
# ... CONTINUES ON THE NEXT SLIDE ...
```

Reviewing the Existing Unit Tests

~/apache/spec/unit/recipes/default_spec.rb

```
# ... CONTINUES FROM THE PREVIOUS SLIDE ...

describe 'for the admin site' do
  it 'creates the directory' do
    expect(chef_run).to create_directory('/srv/httpd/admins/html')
  end

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

  it 'creates a new home page' do
    expect(chef_run).to render_file('/srv/httpd/admins/html/index.html').with('<h1>')
  end
# ... CONTINUES ON THE NEXT SLIDE ...
```

Reviewing the Existing Unit Tests

~/apache/spec/unit/recipes/default_spec.rb

```
# ... CONTINUES FROM THE PREVIOUS SLIDE ...
```

```
end # describe admin site
end # context
end # describe 'apache::default'
```

Executing the Existing Unit Tests

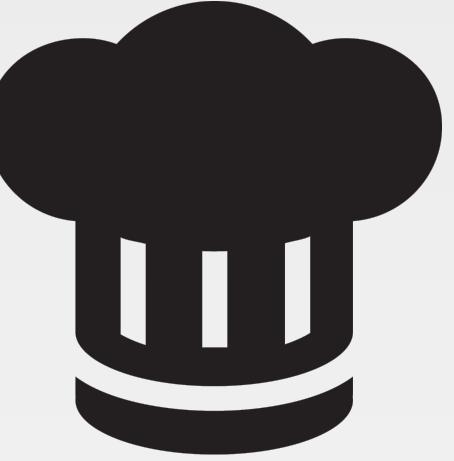


```
> chef exec rspec
```

```
.....
```

```
Finished in 0.92866 seconds (files took 2.76 seconds to load)  
8 examples, 0 failures
```

EXERCISE



Review the Cookbook

We need to make sure everything works before we get started.

Objective:

- ✓ Checkout different branch of cookbook
- ✓ Review and execute the integration tests
- ✓ Review and execute the unit tests
- Review the default recipe

Reviewing the Default Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
package 'httpd'  
  
file '/var/www/html/index.html' do  
  content '<h1>Welcome home!</h1>'  
end  
  
# ... CONTINUES ON THE NEXT SLIDE ...
```

Reviewing the Default Recipe

~/apache/recipes/default.rb

```
# ... CONTINUES FROM THE PREVIOUS SLIDE ...

directory '/srv/httpd/admins/html' do
  recursive true
  mode '0755'
end

template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'
  variables(document_root: '/srv/httpd/admins/html', port: 8080)
  notifies :restart, 'service[httpd]'
end

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

# ... CONTINUES ON THE NEXT SLIDE ...
```

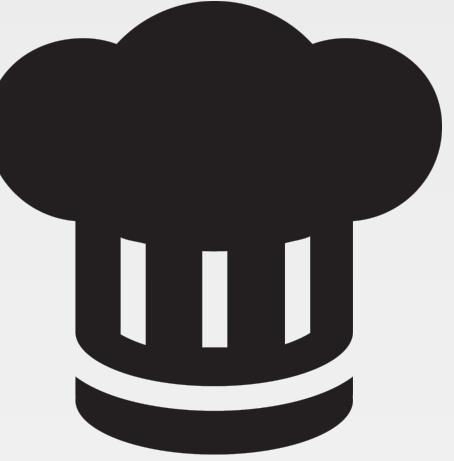
Reviewing the Default Recipe

~/apache/recipes/default.rb

```
# ... CONTINUES FROM THE PREVIOUS SLIDE ...
```

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

EXERCISE



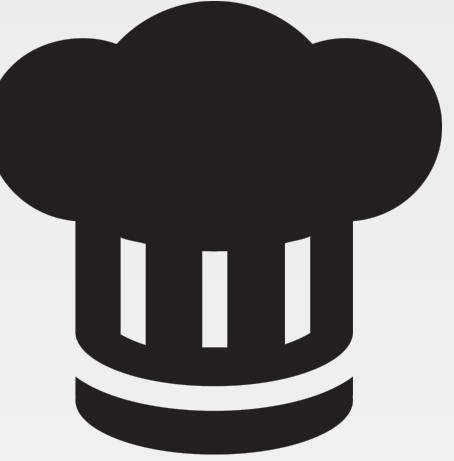
Review the Cookbook

We need to make sure everything works before we get started.

Objective:

- ✓ Checkout different branch of cookbook
- ✓ Review and execute the integration tests
- ✓ Review and execute the unit tests
- ✓ Review the default recipe

EXERCISE



Creating a Custom Resource

This will make our recipe much cleaner.

Objective:

- Create a custom resource that create the admin site
- Allow the custom resource to have configurable properties

Generating a Custom Resource



```
> chef generate lwrp vhost
```

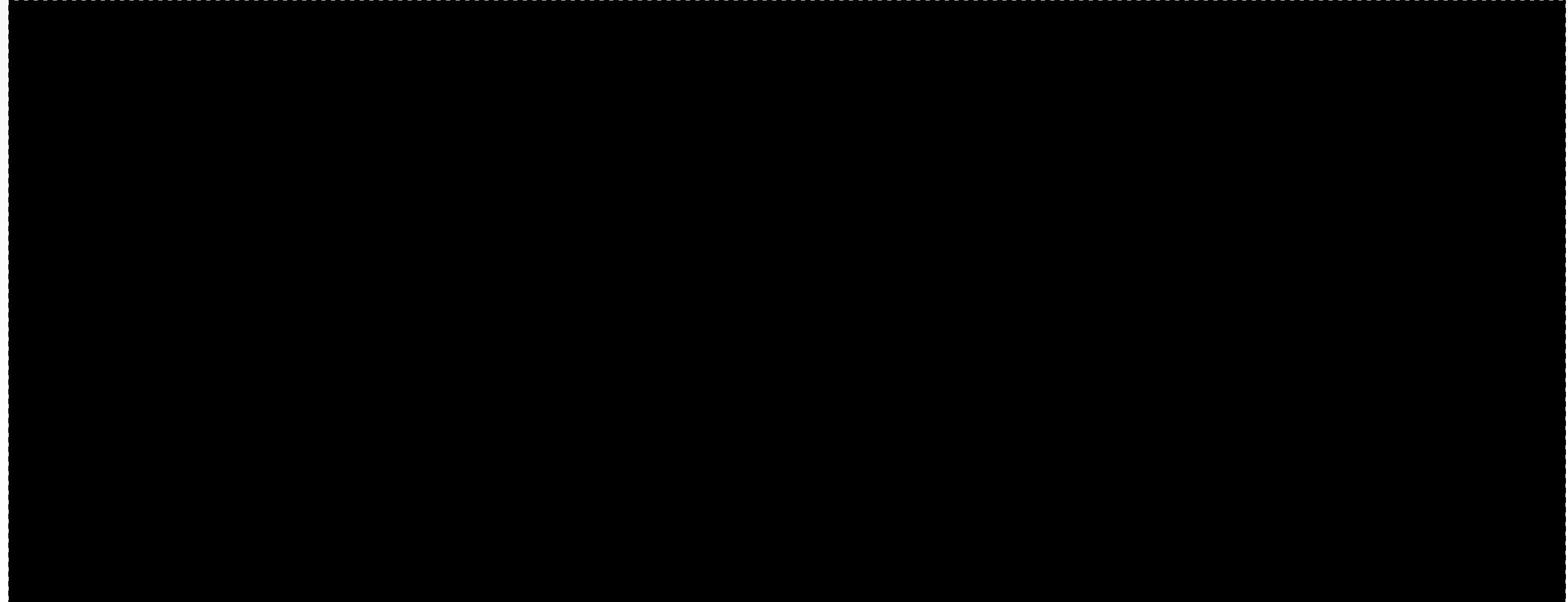
```
Compiling Cookbooks...

Recipe: code_generator::lwrp
  * directory[/home/chef/apache/resources] action create
    - create new directory /home/chef/apache/resources
  * template[/home/chef/apache/resources/vhost.rb] action create
    - create new file ~/httpd/resources/vhost.rb
    - update content in file /home/chef/apache/resources/vhost.rb from none to e3b0c4
      (diff output suppressed by config)
  * directory[/home/chef/apache/providers] action create
```

Removing an un-needed Directory



```
> rm -rf providers
```



Defining the Create Action

~/apache/resources/vhost.rb

```
action :create do  
  
end
```

Implementing the Create Action

~/apache/resources/vhost.rb

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

  template '/etc/httpd/conf.d/admins.conf' do
    source 'conf.erb'
    mode '0644'
    variables(document_root: '/srv/httpd/admins/html', port: 8080)
    notifies :restart, 'service[httpd]'
  end

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

Refactoring the Default Recipe

~/apache/recipes/default.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: default  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
package 'httpd'  
  
file '/var/www/html/index.html' do  
  content '<h1>Welcome home!</h1>'  
end  
  
directory '/srv/httpd/admins/html' do  
  recursive true  
  mode '0755'  
end  
  
template '/etc/httpd/conf.d/admins.conf' do
```

Refactoring the Default Recipe

~/apache/recipes/default.rb

```
template '/etc/httpd/conf.d/admins.conf' do
  source 'conf.erb'
  mode '0644'
  variables(
    document_root: '/srv/httpd/admins/html',
    port: 8080
  )
  notifies :restart, 'service[httpd]'
end

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

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

Adding the New Custom Resource



~/apache/recipes/default.rb

```
package 'httpd'

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

apache_vhost 'admins' do
  action :create
end

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

Executing the Unit Tests



```
> chef exec rspec
```

```
Finished in 0.9673 seconds (files took 2.72 seconds to load)  
8 examples, 3 failures
```

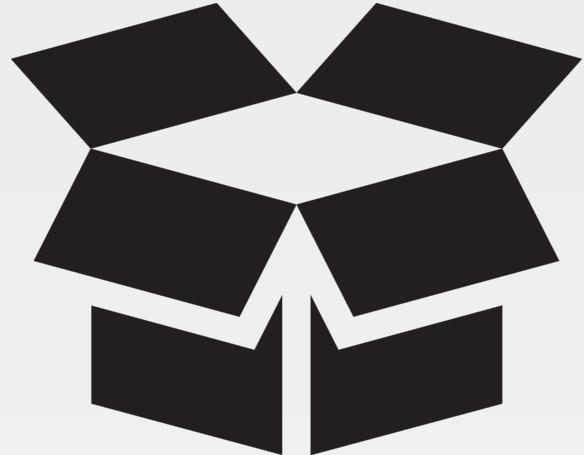
Failed examples:

```
rspec ./spec/unit/recipes/default_spec.rb:39 # apache::default When all attributes are  
default, on an unspecified platform for the admin site creates the directory
```

```
rspec ./spec/unit/recipes/default_spec.rb:43 # apache::default When all attributes are  
default, on an unspecified platform for the admin site creates the configuration
```

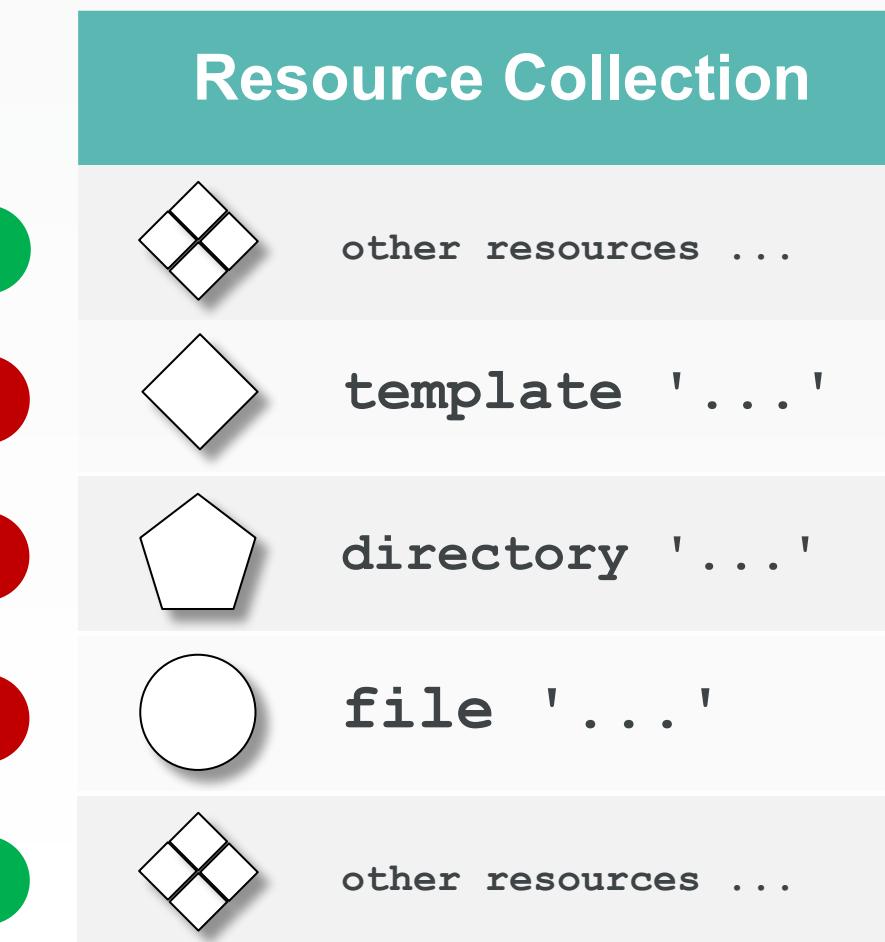
```
rspec ./spec/unit/recipes/default_spec.rb:47 # apache::default When all attributes are  
default, on an unspecified platform for the admin site creates a new home page
```

CONCEPT



Resource Collection

> rspec ➔

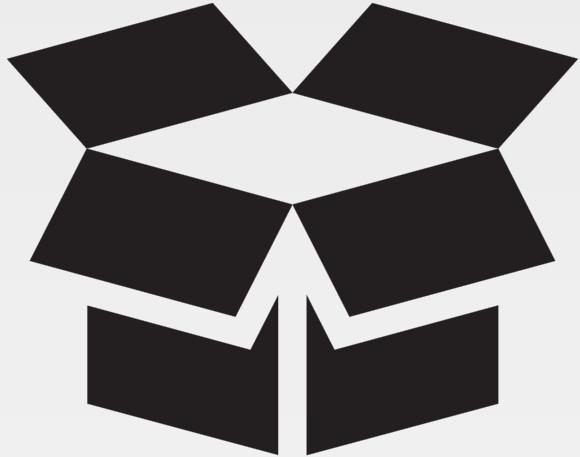


This resource is missing from the resource collection.

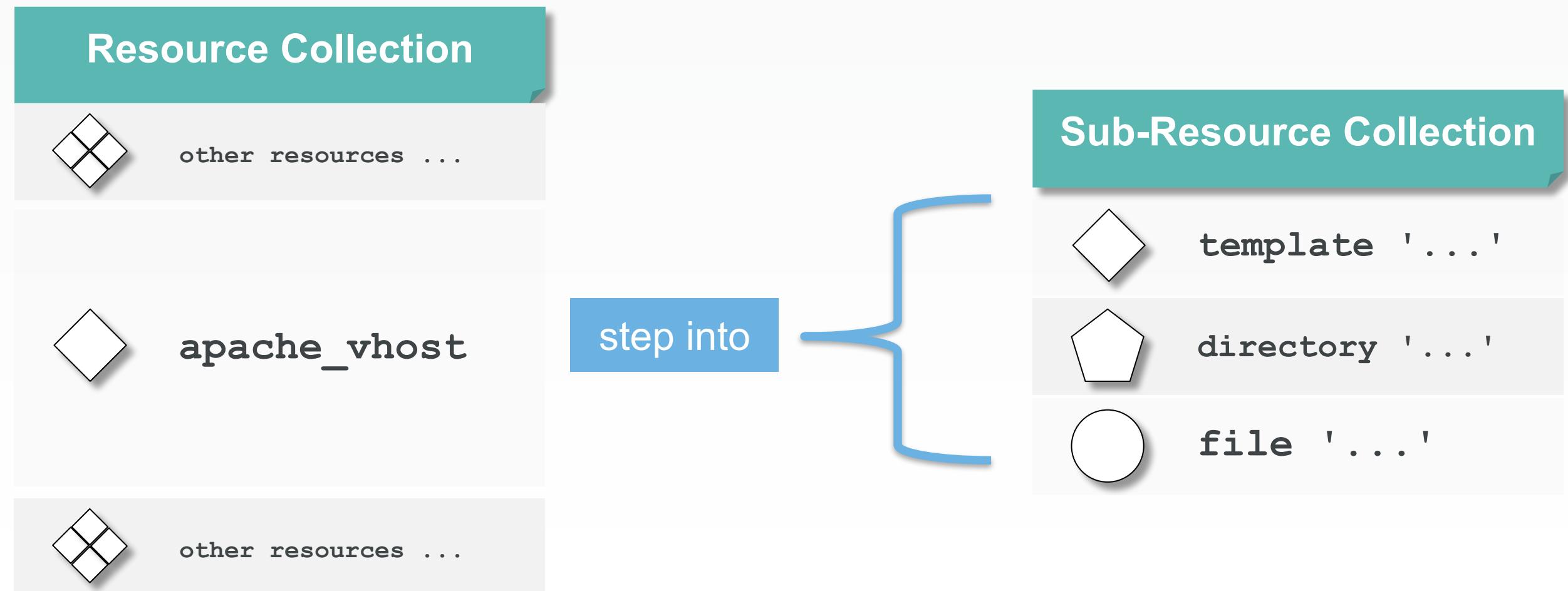
This resource is missing from the resource collection.

This resource is missing from the resource collection.

CONCEPT



A Sub-Resource Collection



Updating the Unit Tests to Verify Custom Resource



~/apache/spec/unit/recipes/default_spec.rb

```
require 'spec_helper'

describe 'apache::default' do
  context 'When all attributes are default, on an unspecified platform' do
    let(:chef_run) do
      runner = ChefSpec::ServerRunner.new(step_into: ['apache_vhost'])
      runner.converge(described_recipe)
    end

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

# ... CONTINUES ON THE NEXT SLIDE ...
```

Executing the Unit Tests



```
> chef exec rspec
```

```
.....
```

```
Finished in 0.98788 seconds (files took 2.95 seconds to load)
```

```
8 examples, 0 failures
```

Converging the Test Instance



```
> kitchen converge
```

```
(up to date)
(up to date)
(up to date)
(up to date)
* service[apache] action start (up to date)
```

```
Running handlers:
```

```
Running handlers complete
```

```
Chef Client finished, 0/8 resources updated in 05 seconds
```

```
Finished converging <default-centos-67> (0m8.81s).
```

```
-----> Kitchen is finished. (0m9.80s)
```

Verifying the Test Instance



```
> kitchen verify
```

```
----> Starting Kitchen (v1.11.1)
----> Setting up <default-centos-67>...
      Finished setting up <default-centos-67> (0m0.00s).
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

```
Target: ssh://vagrant@127.0.0.1:2222
```

- ✓ Command curl http://localhost stdout should match /Welcome home/
- ✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

```
Summary: 2 successful, 0 failures, 0 skipped
      Finished verifying <default-centos-67> (0m0.70s).
----> Kitchen is finished. (0m1.82s)
```

Run a Complete Test on the Test Instance



```
> kitchen test
```

```
Target: ssh://vagrant@127.0.0.1:2222
```

- ✓ Command curl http://localhost stdout should match /Welcome home/
- ✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

```
Summary: 2 successful, 0 failures, 0 skipped
```

```
    Finished verifying <default-centos-67> (0m0.70s) .
```

```
----> Destroying <default-centos-67>...
```

```
    ==> default: Forcing shutdown of VM... .
```

```
    ==> default: Destroying VM and associated drives... .
```

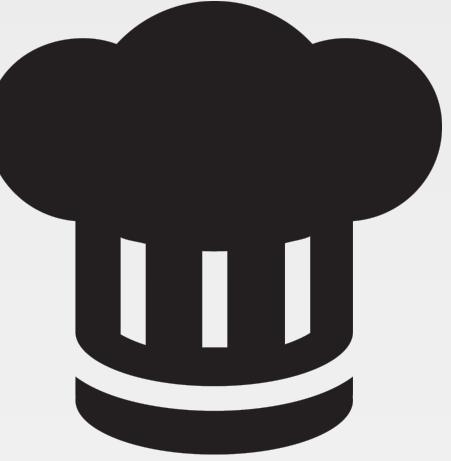
```
    Vagrant instance <default-centos-67> destroyed.
```

```
    Finished destroying <default-centos-67> (0m4.09s) .
```

```
    Finished testing <default-centos-67> (2m21.40s) .
```

```
----> Kitchen is finished. (2m22.49s)
```

EXERCISE



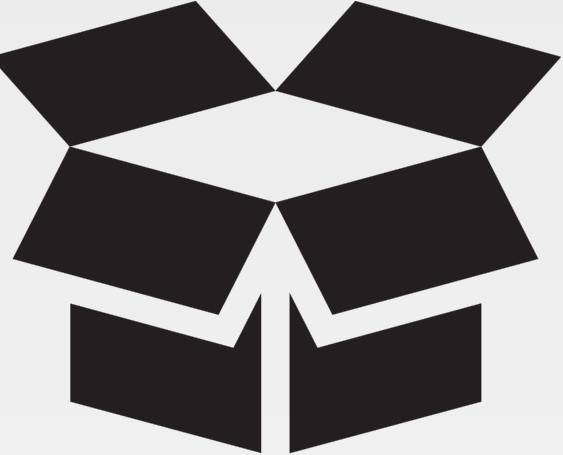
Creating a Custom Resource

This will make our recipe much cleaner.

Objective:

- ✓ Create a custom resource that create the admin site
- ❑ Allow the custom resource to have configurable properties

CONCEPT



Resource Properties

A property is defined with the following syntax.

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

```
property :site_name, String
```

1

2

property method

1

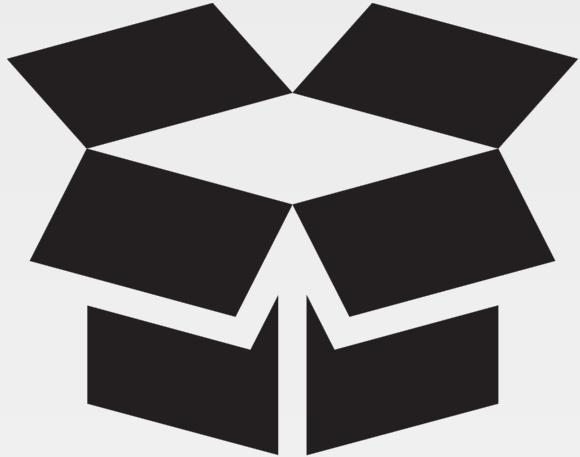
name (symbol)

2

type

Optional Parameters

CONCEPT



Resource Properties

Properties are defined in the resource definition and then available within definition of the resource within the recipe.

~/apache/resources/vhost.rb

```
property :site_name, String
```

~/apache/recipes/default.rb

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

Defining a Property to Manage the site_name

~/apache/resources/vhost.rb

```
property :site_name, String

action :create do

  directory "/srv/httpd/#{site_name}/html" do
    recursive true
    mode '0755'
  end

  template "/etc/httpd/conf.d/#{site_name}.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/httpd/#{site_name}/html", port: 8080)
    notifies :restart, 'service[httpd]'
  end

# ... CONTINUES ON THE NEXT SLIDE ...

```

Updating the Action to use the Property

~/apache/resource/vhost.rb

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

  template "/etc/httpd/conf.d/#{site_name}.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/httpd/#{site_name}/html", port: 8080)
    notifies :restart, "service[httpd]"
  end

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

Updating the Resource to use the Property

~/apache/recipes/default.rb

```
package 'httpd'

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

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

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

Executing the Unit Tests



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 7.58 seconds to load)
```

```
8 examples, 0 failures
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

```
Target: ssh://vagrant@127.0.0.1:2222
```

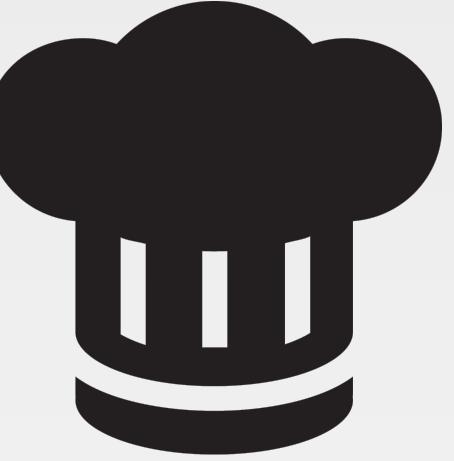
- ✓ Command curl http://localhost stdout should match /Welcome home/
- ✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

```
Summary: 2 successful, 0 failures, 0 skipped
```

```
      Finished verifying <default-centos-67> (0m0.77s) .
```

```
----> Kitchen is finished. (2m42.37s)
```

EXERCISE



Creating a Custom Resource

That's much better.

Objective:

- ✓ Create a custom resource that creates the admin site
- ✓ Allow the custom resource to have configurable properties

LAB



apache_vhost - site_port Property

- ❑ Create a custom resource property named **site_port** that is a *Fixnum*
- ❑ Within the apache_vhost's create action replace the hard-coded value 8080 with the **site_port** property
- ❑ Within the default recipe set the apache_vhost resource named 'admins' to have a site_port 8080

Defining a Property to Manage the site_port

~/apache/resource/vhost.rb

```
property :site_name, String
property :site_port, Fixnum

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

  template "/etc/httpd/conf.d/#{site_name}.conf" do
    source 'conf.erb'
    mode '0644'
    variables(document_root: "/srv/httpd/#{site_name}/html", port: site_port)
    notifies :restart, 'service[httpd]'
  end
end

# ... REMAINDER OF CUSTOM RESOURCE ...
```

Updating the Resource to use the Property

~/apache/recipes/default.rb

```
package 'httpd'

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

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

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

Executing the Unit Tests



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 7.58 seconds to load)
```

```
8 examples, 0 failures
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

```
Target: ssh://vagrant@127.0.0.1:2222
```

- ✓ Command curl http://localhost stdout should match /Welcome home/
- ✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

```
Summary: 2 successful, 0 failures, 0 skipped
```

```
Finished verifying <default-centos-67> (0m0.77s).
```

```
----> Kitchen is finished. (2m42.37s)
```

LAB



apache_vhost - site_port Property

- ✓ Create a custom resource property named **site_port** that is a *Fixnum*
- ✓ Within the apache_vhost's create action replace the hard-coded value 8080 with the **site_port** property
- ✓ Within the default recipe set the apache_vhost resource named 'admins' to have a site_port 8080

PROBLEM



Remove the Welcome Site

When apache installs itself it defines a default site on port 80. Up until this point we have relied on this site. We now want to remove that initial welcome site but we want to do that with a new action we will define on the custom resource we are defining.

LAB



apache_vhost Remove Action

- ❑ Define the 'remove' action for apache_vhost that defines the policy:

A directory named "/srv/httpd/#{{site_name}}/html" is deleted

A file named "/etc/httpd/conf.d/#{{site_name}}.conf" is deleted

- ❑ Update the default recipe's policy to include a new resource:

An apache_vhost resource named 'welcome' is removed

Defining the Resource's Remove Action

~/apache/resources/vhost.rb

```
# ... CREATE ACTION ...

action :remove do
  directory "/srv/httpd/#{site_name}/html" do
    action :delete
  end

  file "/etc/httpd/conf.d/#{site_name}.conf" do
    action :delete
  end
end
```

Adding the Resource with Remove Action to the Recipe



~/apache/recipes/default.rb

```
package 'httpd'

apache_vhost 'welcome' do
  site_name 'welcome'
  action :remove
end

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

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

LAB



apache_vhost Remove Action

- ✓ Define the 'remove' action for apache_vhost that defines the policy:

A directory named "/srv/httpd/#{{site_name}}/html" is deleted

A file named "/etc/httpd/conf.d/#{{site_name}}.conf" is deleted

- ✓ Update the default recipe's policy to include a new resource:

An apache_vhost resource named 'welcome' is removed

LAB



apache_vhost Remove Action

- ❑ Update the default recipe's policy:
 - Remove the file resource named '/var/www/html/index.html'
 - Add an apache_vhost resource named 'users' is created with the site_port 80
- ❑ Update the ChefSpec tests to stop expecting the file resource and start expecting the new resources found within the apache_vhost resource named 'users'
- ❑ Update the InSpec tests to expect the default site to "Welcome users!"

Removing the Resource from the Recipe

~/apache/recipes/default.rb

```
package 'httpd'

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

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

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

Adding the Resource to create the users site

~/apache/recipes/default.rb

```
apache_vhost 'welcome' do
  site_name 'welcome'
  action :remove
end

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

apache_vhost 'admins' do
  notifies :restart, 'service[httpd]'
end
```

Removing the Un-needed Unit Test Expectation

~/apache/spec/unit/recipes/default_spec.rb

```
# ... EXAMPLES DEFINED ABOVE ...

describe 'for the default site' do
  it 'writes out a new home page' do
    expect(chef_run).not_to
render_file('/var/www/html/index.html').with_content('<h1>Welcome home!</h1>')
  end
end

# ... EXAMPLES DEFINED BELOW ...
```

Adding Expectations for the users Site



~/apache/spec/unit/recipes/default_spec.rb

```
# ... EXAMPLES DEFINED ABOVE ...

describe 'for the users site' do
  it 'creates the directory' do
    expect(chef_run).to create_directory('/srv/httpd/users/html')
  end

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

  it 'creates a new home page' do
    expect(chef_run).to
render_file('/srv/httpd/users/html/index.html').with_content('<h1>Welcome users!</h1>')
  end

# ... EXAMPLES DEFINED ABOVE ...
```

Executing the Unit Test Suite



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 2.56 seconds to load)  
10 examples, 0 failures
```

Updating the Expectation for the users Site

~/apache/test/smoke/default/default_test.rb

```
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
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing

Target: ssh://vagrant@127.0.0.1:2222

✓ Command curl http://localhost stdout should match /Welcome home/
✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

Summary: 2 successful, 0 failures, 0 skipped
Finished verifying <default-centos-67> (0m0.77s).
----> Kitchen is finished. (2m42.37s)
```



apache_vhost Remove Action

- ✓ Update the default recipe's policy:
 - Remove the file resource named '/var/www/html/index.html'
 - Add an apache_vhost resource named 'users' is created with the site_port 80
- ✓ Update the ChefSpec tests to stop expecting the file resource and start expecting the new resources found within the apache_vhost resource named 'users'
- ✓ Update the InSpec tests to expect the default site to "Welcome users!"

DISCUSSION



Discussion

What are the benefits of using a custom resource to manage the virtual hosts? What are the drawbacks of using a custom resource?

What does the resource collection look like when using a custom resource?

DISCUSSION



Q&A

What questions can we answer for you?



CHEF™

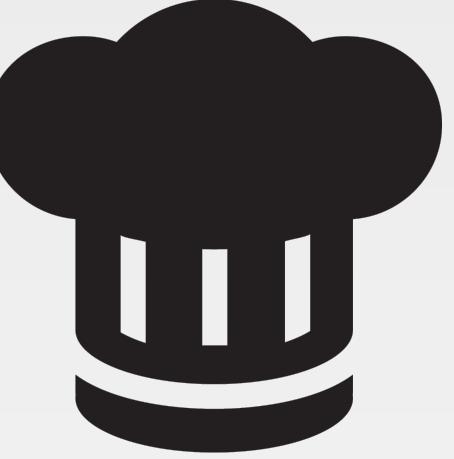
Refining a Custom Resource

Objectives

After completing this module, you should be able to:

- Set a custom resource's name to a property
- Set a default value for a custom resource property
- Define notifications correctly when creating custom resources

EXERCISE



Refactoring the Resources

Objective:

- 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

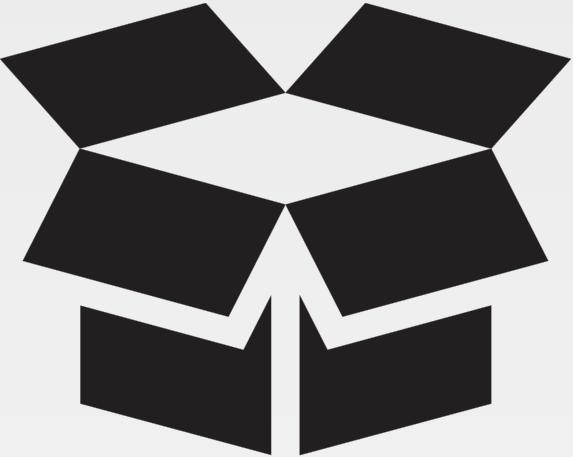
PROBLEM



Selecting a Default Action

Resources often have a default action. The default action is often the action that is the least surprising. For most resources it is an additive/restorative action that moves the system into the desired state.

CONCEPT



The First Action is the Default

The first action within the custom resource definition is the default one. But you can explicitly define the default action within the resource definition.

Defining a Default Action for the Resource

~/apache/resources/vhost.rb

```
property :site_name, String
property :site_port, Fixnum

default_action :create

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

# ... REMAINING RESOURCE DEFINITION ...
```

Removing the Default Action for Resources

~/apache/recipes/default.rb

```
# ... INITIAL RECIPE DEFINITION ...

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

apache_vhost 'admins' do
  site_port 8080
  site_name 'admins'
```

Removing the Default Action for Resources

~/apache/recipes/default.rb

```
# ... RECIPE DEFINITION ...

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

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

Executing the Unit Test Suite



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 2.56 seconds to load)  
10 examples, 0 failures
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

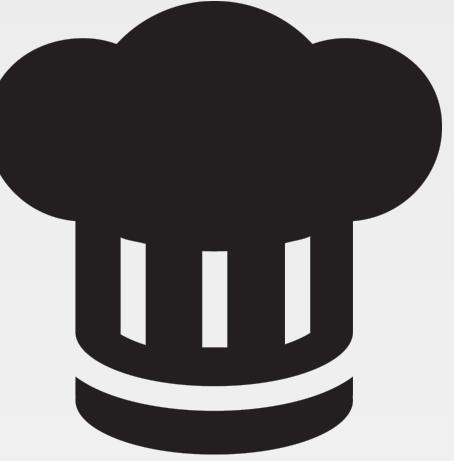
```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing

Target: ssh://vagrant@127.0.0.1:2222

✓ Command curl http://localhost stdout should match /Welcome home/
✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

Summary: 2 successful, 0 failures, 0 skipped
Finished verifying <default-centos-67> (0m0.77s).
----> Kitchen is finished. (2m42.37s)
```

EXERCISE



Refactoring the Resources

Objective:

- ✓ 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

PROBLEM



Clarity in the Custom Resource

There is some duplication in the declaration of the resource. The name of the resource and the `site_name`. We want to default to use the name specified in the resource as the `site_name`. This is similar to other resources.

Tying the Resource Name to the Property

~/apache/resources/vhost.rb

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

default_action :create

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

# ... REMAINING RESOURCE DEFINITION ...
```

Removing the site_name Property

~/apache/recipes/default.rb

```
package 'apache'

apache_vhost 'welcome' do
  site_name 'welcome'
  action :remove
end

apache_vhost 'admins' do
  site_port 80
  site_name 'users'
end
```

Removing the site_name Property

~/apache/recipes/default.rb

```
# ... INITIAL RECIPE DEFINITION ...

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

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

Executing the Unit Test Suite



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 2.56 seconds to load)  
10 examples, 0 failures
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

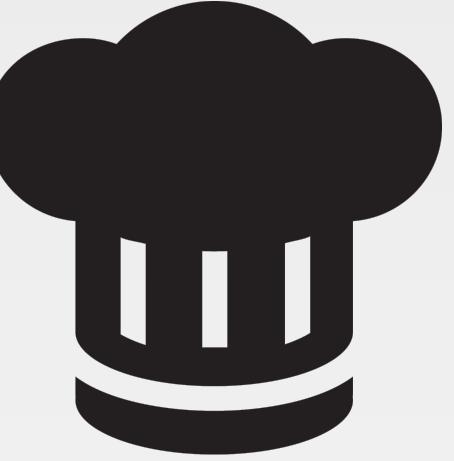
```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing

Target: ssh://vagrant@127.0.0.1:2222

✓ Command curl http://localhost stdout should match /Welcome home/
✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

Summary: 2 successful, 0 failures, 0 skipped
Finished verifying <default-centos-67> (0m0.77s).
----> Kitchen is finished. (2m42.37s)
```

EXERCISE

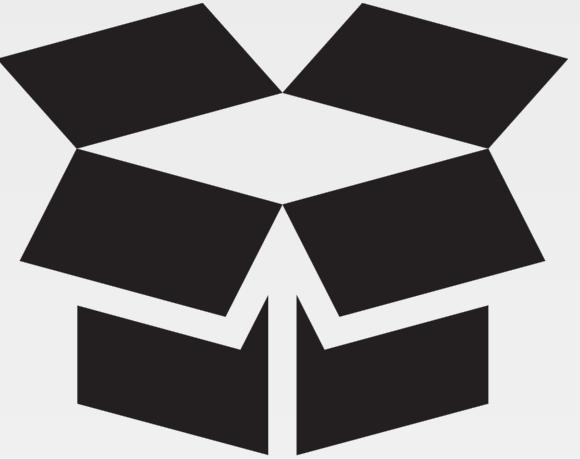


Refactoring the Resources

Objective:

- ✓ 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

CONCEPT



Setting Default Values for Properties

Properties can also have default values. When deploying a website the default port that one would expect to see that site is on port 80.

Setting a Default Value for the Property

~/apache/resources/vhost.rb

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

default_action :create

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

# ... REMAINING RESOURCE DEFINITION ...
```

Removing the site_port Property

~/apache/recipes/default.rb

```
package 'apache'

apache_vhost 'welcome' do
  site_name 'welcome'
  action :remove
end

apache_vhost 'users' do
  site_port 80
end
```

Executing the Unit Test Suite



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 2.56 seconds to load)  
10 examples, 0 failures
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

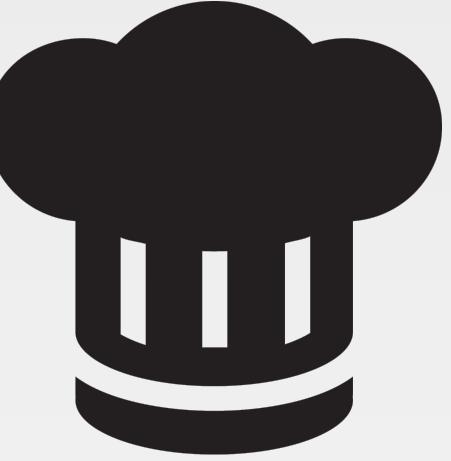
```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing

Target: ssh://vagrant@127.0.0.1:2222

✓ Command curl http://localhost stdout should match /Welcome home/
✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

Summary: 2 successful, 0 failures, 0 skipped
Finished verifying <default-centos-67> (0m0.77s).
----> Kitchen is finished. (2m42.37s)
```

EXERCISE

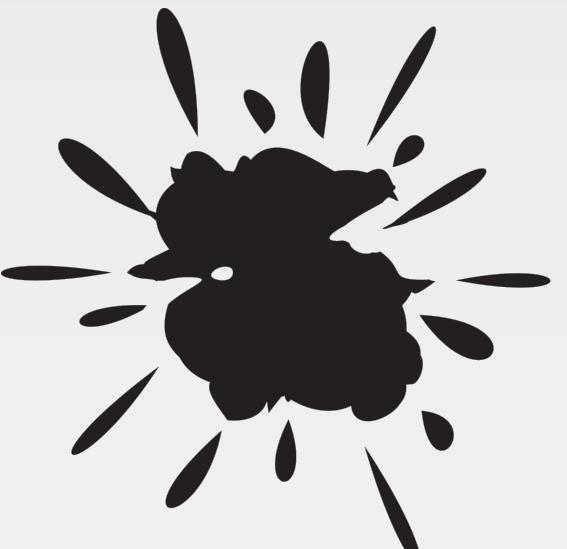


Refactoring the Resources

Objective:

- ✓ 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

PROBLEM



Resource Notifications

Defining a notification in a resource within a custom resource creates a fragile relationship. One that we want to address by removing any notifications to outside resources.

Removing the Notification from the Resource

~/apache/resources/vhost.rb

```
# ... INITIAL RESOURCE DEFINITION ...

template "/etc/httpd/conf.d/#{site_name}.conf" do
  source 'conf.erb'
  mode '0644'
  variables(:document_root => "/srv/apache/#{site_name}/html",
:port => site_port)
  notifies :restart, 'service[httpd]'
end

# ... REMAINDER OF CUSTOM RESOURCE ...
```

Adding the Notification to the Resources

~/apache/recipes/default.rb

```
package 'apache'

apache_vhost 'welcome' do
  notifies :restart, 'service[httpd]'
  action :remove
end

apache_vhost 'users' do
  notifies :restart, 'service[httpd]'
end
```

Adding the Notification to the Resources

~/apache/recipes/default.rb

```
# ... INITIAL RECIPE DEFINITION ...

apache_vhost 'admins' do
  notifies :restart, 'service[httpd]'
  site_port 8080
end

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

Executing the Unit Test Suite



```
> chef exec rspec
```

```
.....
```

```
Finished in 1.22 seconds (files took 2.56 seconds to load)  
10 examples, 0 failures
```

Converging and Verifying the Test Instance



```
> kitchen converge && kitchen verify
```

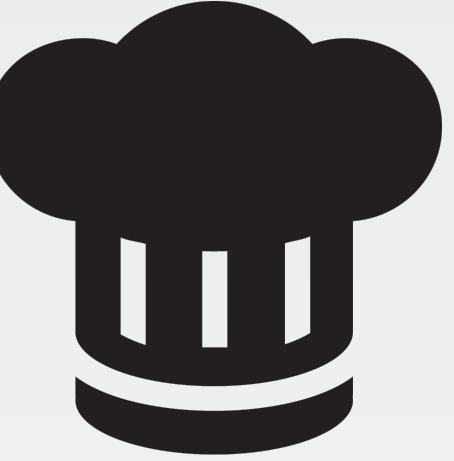
```
----> Verifying <default-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing

Target: ssh://vagrant@127.0.0.1:2222

✓ Command curl http://localhost stdout should match /Welcome home/
✓ Command curl http://localhost:8080 stdout should match /Welcome admins/

Summary: 2 successful, 0 failures, 0 skipped
Finished verifying <default-centos-67> (0m0.77s).
----> Kitchen is finished. (2m42.37s)
```

EXERCISE



Refactoring the Resources

Objective:

- ✓ 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

DISCUSSION



Q&A

What questions can we answer for you?



CHEF™



Ohai

Objectives

After completing this module, you should be able to:

- Execute the Ohai command-line tool to return an attribute
- Describe when Ohai is loaded in the chef-client run
- Describe when new attributes for the node are stored
- Describe precedence of attributes collected by Ohai

CONCEPT

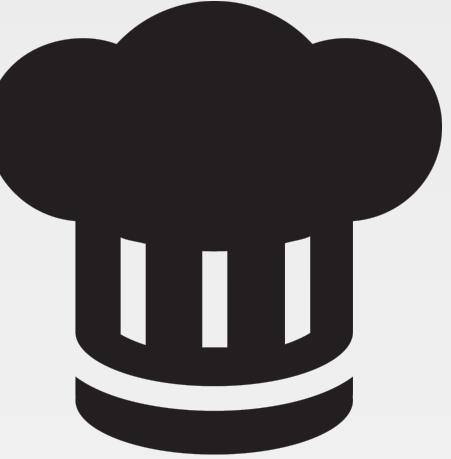


Ohai

Ohai is a tool that is used to detect attributes on a node, and then provide these attributes to the chef-client at the start of every chef-client run. The types of attributes Ohai collects include (but are not limited to):

- Platform details
- Network usage
- Memory usage
- CPU data

EXERCISE



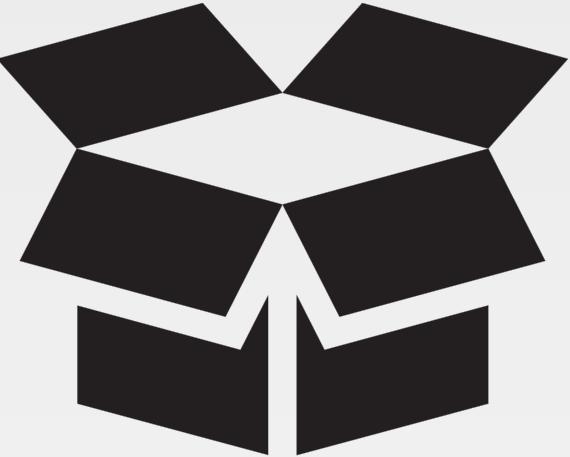
Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- Execute Ohai to retrieve details about the node
- View Ohai's execution within a chef-client run
- Describe attributes precedence

CONCEPT



All About The System

Ohai queries the operating system with a number of commands, similar to the ones demonstrated.

The data is presented in JSON (JavaScript Object Notation).

Running Ohai to Show All Attributes



```
> ohai
```

```
{
  "kernel": {
    "name": "Linux",
    "release": "2.6.32-431.1.2.0.1.el6.x86_64",
    "version": "#1 SMP Fri Dec 13 13:06:13 UTC 2013",
    "machine": "x86_64",
    "os": "GNU/Linux",
    "modules": {
      "veth": {
        "size": "5040",
        "refcount": "0"
      },
      "ipt_addrtype": {
        "size": "5040"
      }
    }
  }
}
```

Running Ohai to Show the IP Address



```
> ohai ipaddress
```

```
[  
  "172.31.57.153"  
]
```

Running Ohai to Show the Hostname



```
> ohai hostname
```

```
[  
  "ip-172-31-57-153"  
]
```

Running Ohai to Show the Memory



```
> ohai memory
```

```
{  
  "swap": {  
    "cached": "0kB",  
    "total": "0kB",  
    "free": "0kB"  
  },  
  "total": "604308kB",  
  "free": "297940kB",  
  "buffers": "24824kB",  
  "cached": "198264kB",  
}
```

Running Ohai to Show the Total Memory



```
> ohai memory/total
```

```
[  
  "604308kB"  
]
```

Running Ohai to Show the CPU



```
> ohai cpu
```

```
{
  "0": {
    "vendor_id": "GenuineIntel",
    "family": "6",
    "model": "45",
    "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",
    "stepping": "7",
    "mhz": "1795.673",
    "cache_size": "20480 KB",
    "physical_id": "34"
  }
}
```

Running Ohai to Show the First CPU



```
> ohai cpu/0
```

```
{  
  "vendor_id": "GenuineIntel",  
  "family": "6",  
  "model": "45",  
  "model_name": "Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz",  
  "stepping": "7",  
  "mhz": "1795.673",  
  "cache_size": "20480 KB",  
  "physical_id": "34",  
  "core_id": "0",  
  "cores": "1"
```

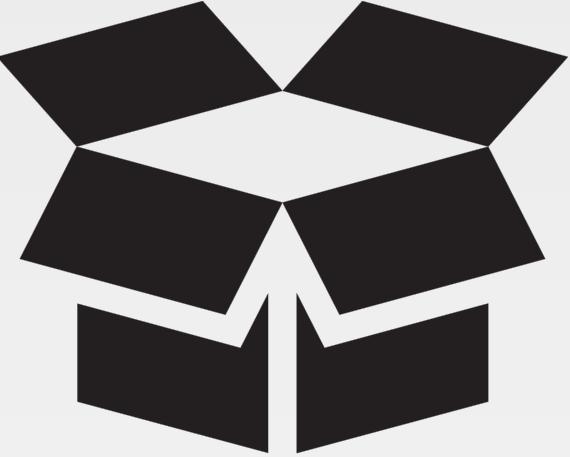
Running Ohai to Show the First CPU Mhz



```
> ohai cpu/0/mhz
```

```
[  
  "1795.673"  
]
```

CONCEPT

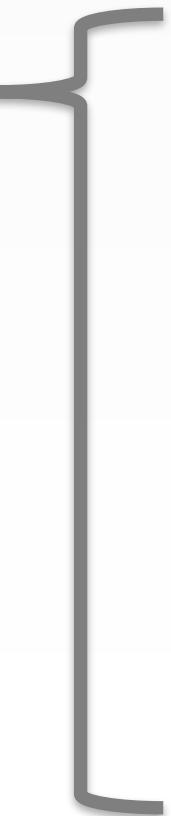


Ohai is Composed of Plugins

Ohai is packaged with a core set of plugins that are automatically loaded when executing Ohai.

These plugins provide the attributes we see in the JSON output (e.g. `ipaddress`, `hostname`, `memory`, `cpu`).

Ohai



Example Plugins

Network Addresses

`ipaddress`, `ip6address`, `macaddress`

Hostname

`hostname`, `domain`, `fqdn`, `machinename`

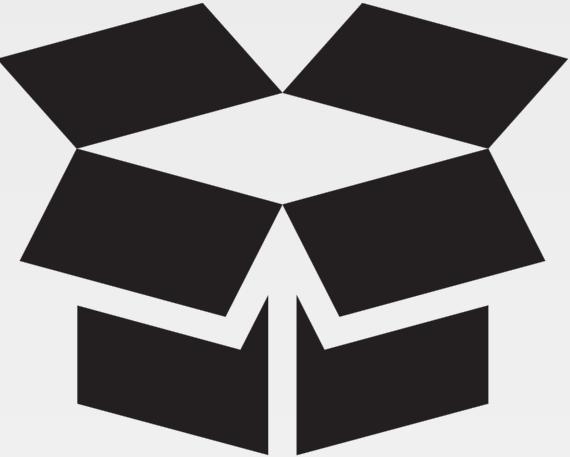
Memory

`memory`, `memory/swap`

CPU

`cpu`

CONCEPT



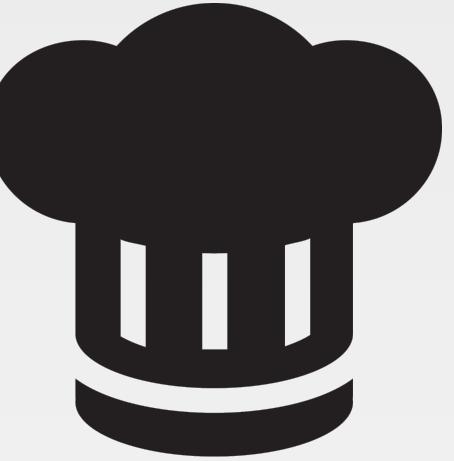
Custom Ohai Plugins

It is possible to define your own plugins and have Ohai load those plugins.

```
> ohai -d PATH_TO_CUSTOM_PLUGINS
```

We will explore creating an Ohai plugin and loading it with Ohai from the command-line in the 'Creating Ohai Plugins' module.

EXERCISE



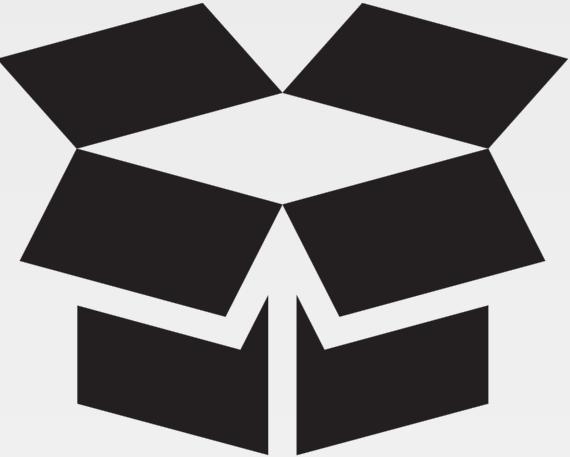
Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- Execute Ohai to retrieve details about the node
- View Ohai's execution within a chef-client run
- Describe attributes precedence

CONCEPT

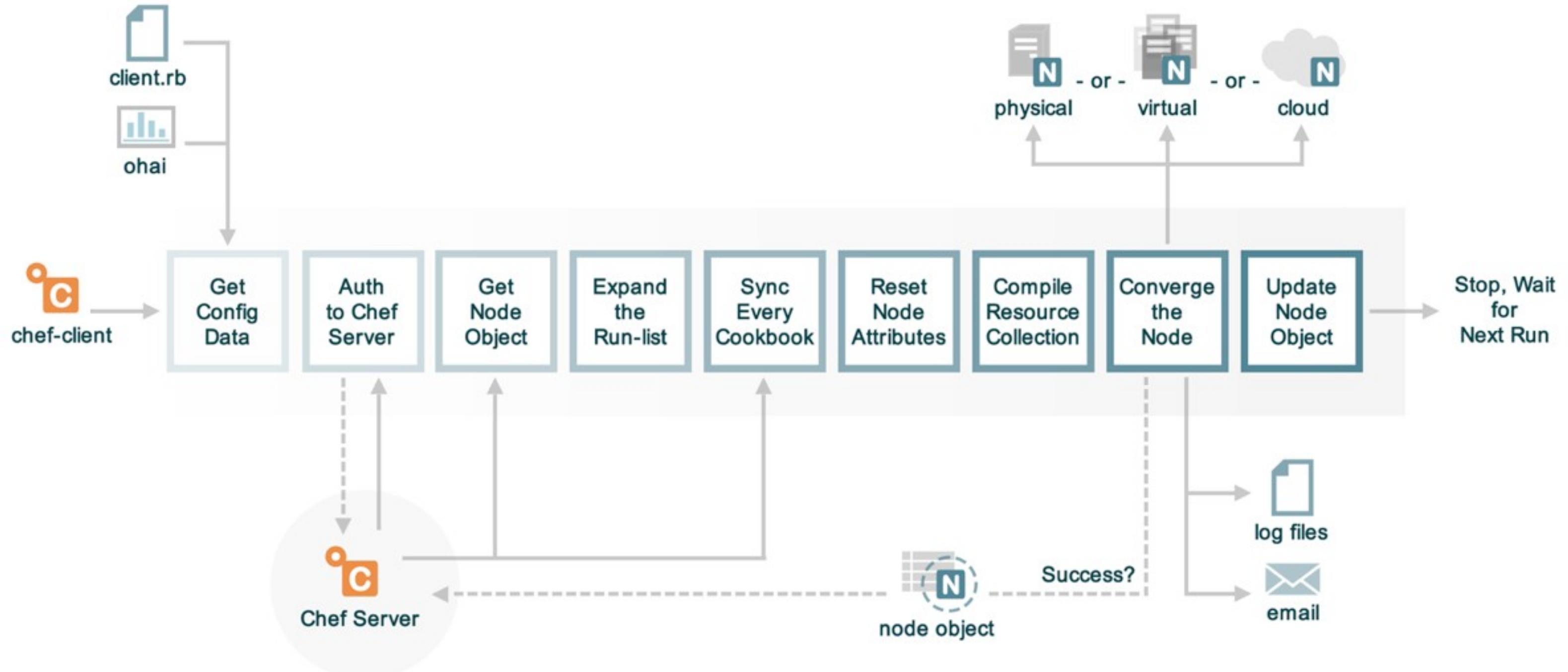


chef-client

chef-client automatically executes ohai and stores the data about the node in an object we can use within the recipes. When the chef-client run completes successfully the details about the node are sent to the Chef Server.

<http://docs.chef.io/ohai.html>

The Anatomy of a chef-client Run



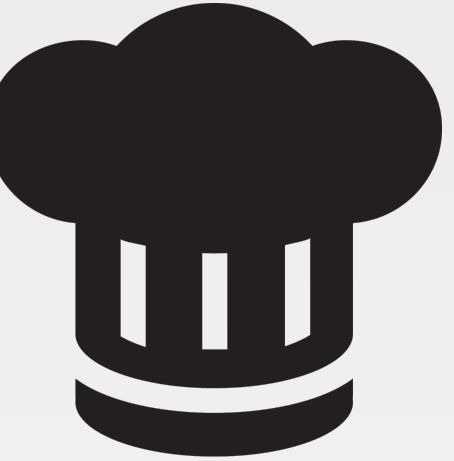
Running Ohai in Code



```
> chef exec pry
```

```
[1] pry(main)> require 'ohai'  
=> true  
  
[2] pry(main)> ohai = Ohai::System.new  
=> #<Ohai::System:0x007fc62fadc490 @plugin_pat...@safe_run=true>>  
  
[3] pry(main)> ohai.all_plugins('ipaddress')  
[4] pry(main)> ohai.all_plugins('hostname')  
[5] pry(main)> ohai.all_plugins('memory')  
[6] pry(main)> ohai.all_plugins('cpu')  
[7] pry(main)> ohai.all_plugins  
[8] pry(main)> exit
```

EXERCISE



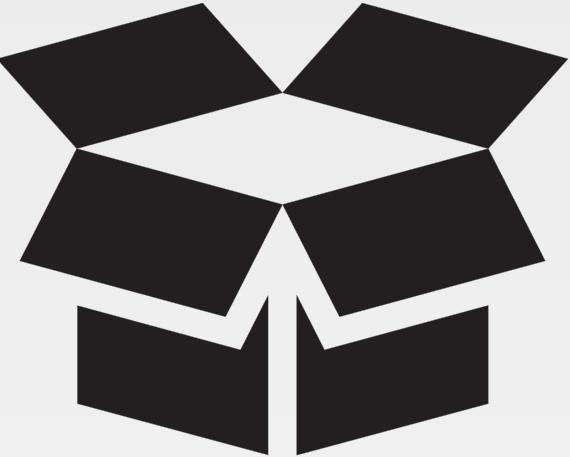
Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- ✓ Execute Ohai to retrieve details about the node
- ✓ View Ohai's execution within a chef-client run
- Describe attributes precedence

CONCEPT



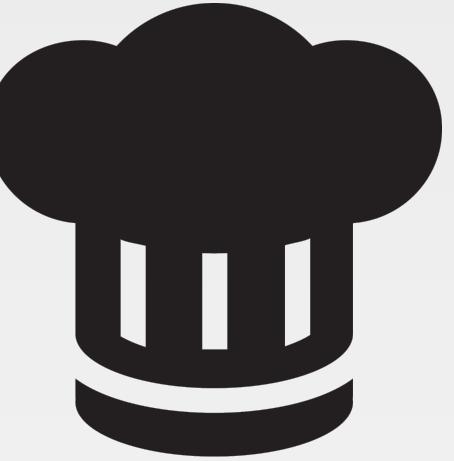
Node Attributes

A node object maintains the attributes collected from Ohai, from the previous node object (as returned by the Chef Server), from the environments, roles, and the cookbooks defined in the run list.

Viewing Attribute Precedence as a Table

LOCATION	Attribute Files	Node / Recipe	Environment	Role	
LEVEL	default	1	2	3	4
force_default	5	6			
normal	7	8			
override	9	10	12	11	
force_override	13	14			
automatic			15		
			Ohai		

EXERCISE



Exploring Ohai

To understand Ohai we must explore it in isolation, understand where it fits in the ecosystem, and how the data it provides is stored.

Objective:

- ✓ Execute Ohai to retrieve details about the node
- ✓ View Ohai's execution within a chef-client run
- ✓ Describe attributes precedence

DISCUSSION



Discussion

When might you execute ohai from the command-line to gather data about the system?

Why might it be important to collect details about the system, through Ohai, early in the chef-client run?

What kind of data should be collected and stored within Ohai? What kind of data should it not collect?

DISCUSSION



Q&A

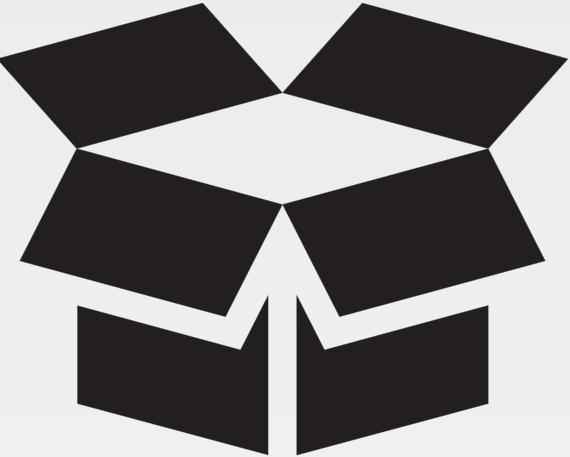
What questions can we answer for you?



CHEF™

Ohai Plugins

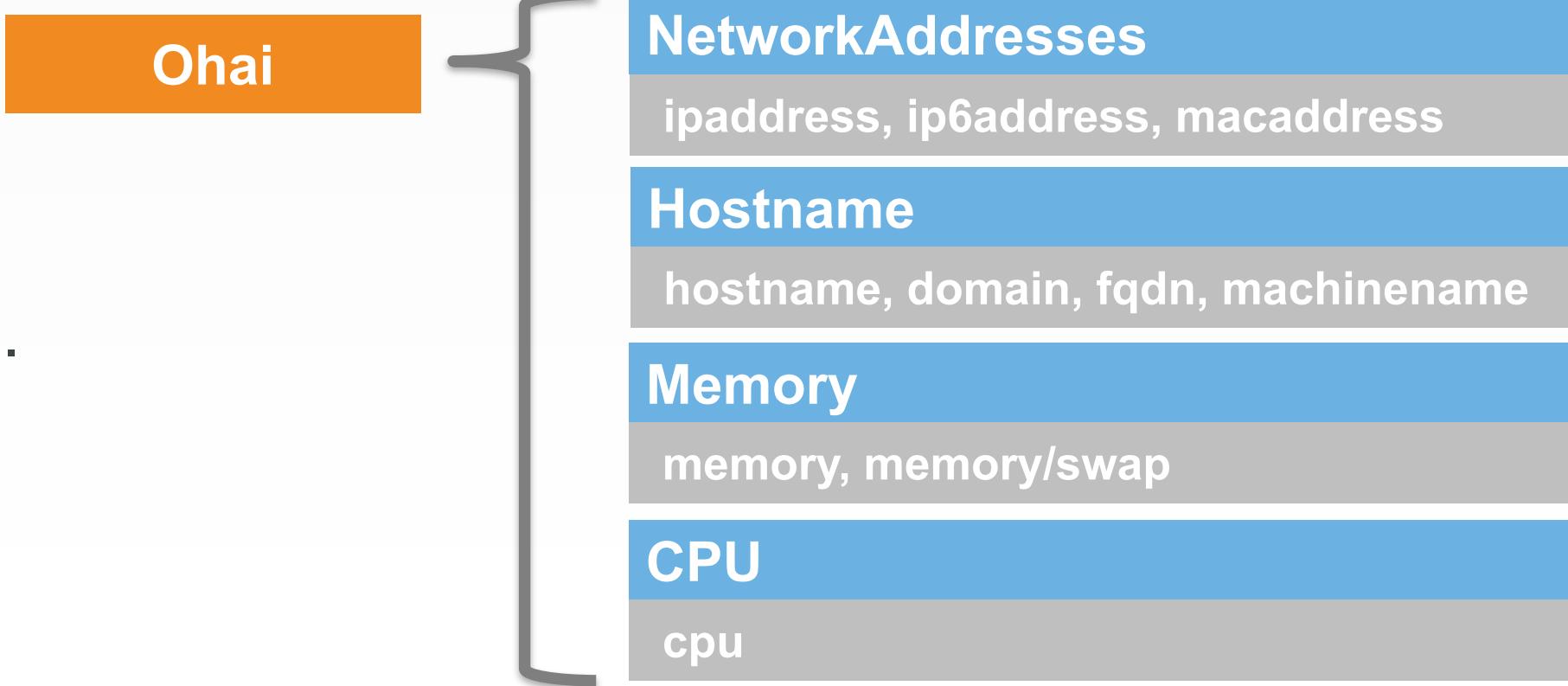
CONCEPT



Ohai is Composed of Plugins

Ohai is packaged with a core set of plugins that are automatically loaded when executing Ohai.

These plugins provide the attributes we see in the JSON output (e.g. `ipaddress`, `hostname`, `memory`, `cpu`).

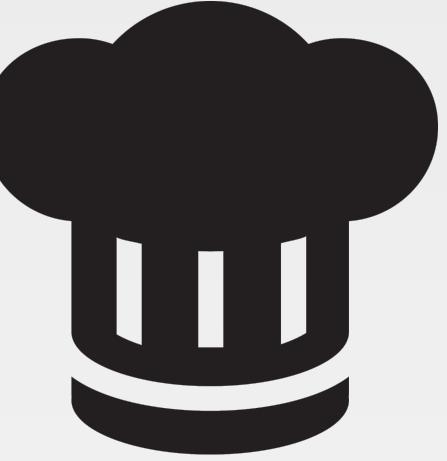


Objectives

After completing this module, you should be able to:

- Find Ohai's core plugins
- Express what a plugin provides, depends on, and how it collects its data

EXERCISE



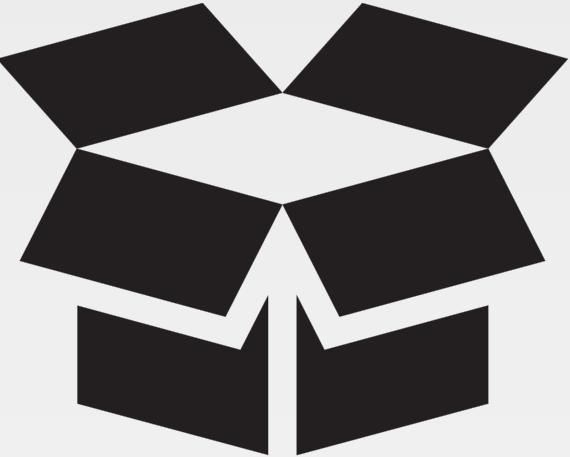
Reviewing the Ohai Gem

Ohai is a Rubygem. First we need to learn about how a gem is structured.

Objective:

- Review the basic structure of the Ohai gem
- Review the 'language' plugin
- Review the 'python' plugin

CONCEPT



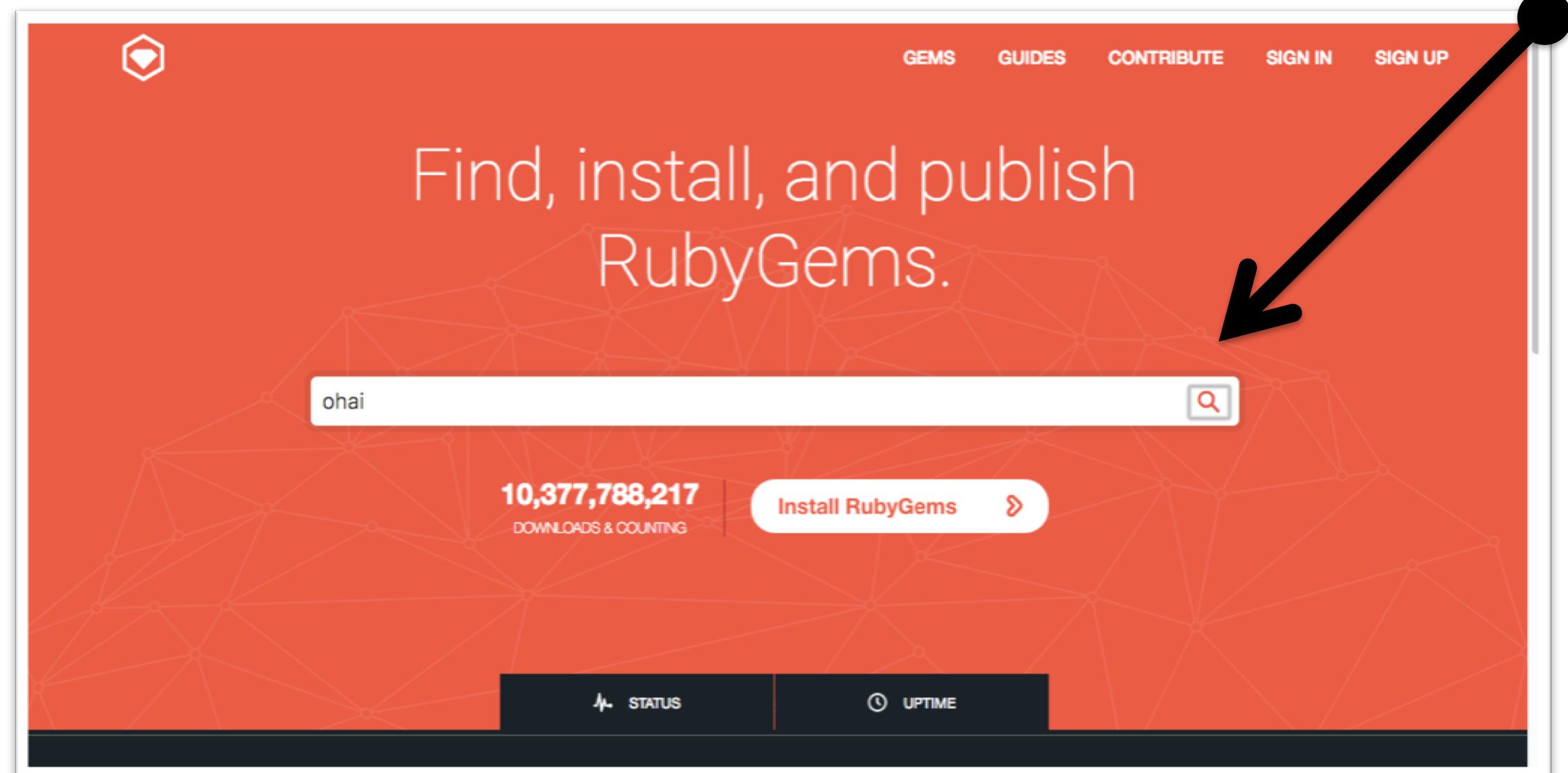
Ohai is Ruby Gem

Ruby gems are the ways in which Ruby developers share the code that they develop with others. A Ruby gem is really a packaging structure similar to that of a Chef cookbook.

Searching for a Gem on Rubygems

Steps

1. Visit <https://rubygems.org>
2. Within the search field enter: **ohai**
3. Press enter or click the magnified glass at the right-side of the search box.
4. Click the 'Source Code' link
5. Click on 'Clone or download' and then copy the git URL.



Searching for a Gem on Rubygems

Steps

1. Visit <https://rubygems.org>
2. Within the search field enter: **ohai**
3. Press enter or click the magnified glass at the right-side of the search box.
4. Click the 'Source Code' link
5. Click on 'Clone or download' and then copy the git URL.

The screenshot shows the Rubygems website with the search term 'ohai' entered. The results page for 'ohai 8.21.0' is displayed. The sidebar on the right contains links for 'Homepage', 'Source Code', 'Documentation', 'Bug Tracker', 'Download', 'Badge', 'Subscribe', 'RSS', and 'Report Abuse'. A blue arrow points from the 'Source Code' link in the sidebar down to the 'Clone or download' button in the main content area.

ohai 8.21.0

Ohai profiles your system and emits JSON

VERSIONS:

- 8.21.0 - October 18, 2016 (501 KB)
- 8.20.0 - September 7, 2016 (498 KB)
- 8.19.2 - August 16, 2016 (496 KB)
- 8.19.1 - August 12, 2016 (502 KB)
- 8.19.0 - August 11, 2016 (496 KB)

Show all versions (100 total) →

RUNTIME DEPENDENCIES:

- chef-config < 13, >= 12.5.0.alpha.1
- ffi ~> 1.9
- ffi-yajl ~> 2.2
- ipaddress >= 0
- mixlib-cli >= 0
- mixlib-config ~> 2.0
- mixlib-log < 2.0, >= 1.7.1
- mixlib-shellout ~> 2.0
- plist ~> 3.1
- systemu ~> 2.6.4
- wmi-lite ~> 1.0

DEVELOPMENT DEPENDENCIES:

- github_changelog_generator = 1.13.1
- rack ~> 1.0
- rake < 12.0.0, >= 10.1.0
- rspec-collection_matchers ~> 1.0
- rspec-core ~> 3.0
- rspec-expectations ~> 3.0
- rspec_junit_formatter >= 0
- rspec-mocks ~> 3.0

TOTAL DOWNLOADS
9,008,075

FOR THIS VERSION
14,432

GEMFILE:
gem 'ohai', '~> 8.2.0' ↗

INSTALL:
gem install ohai ↗

LICENSE:
APACHE-2.0

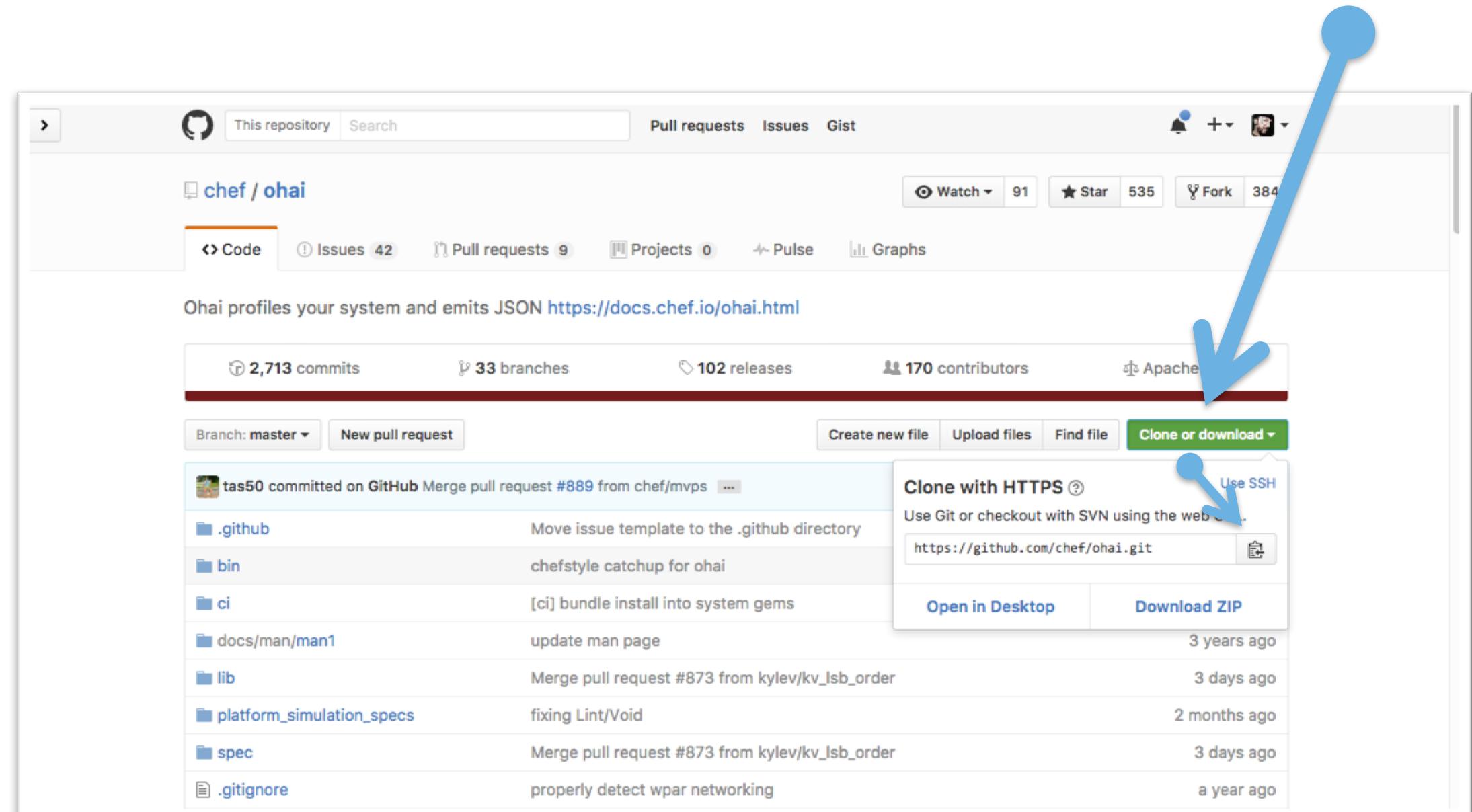
REQUIRED RUBY VERSION
≥ 2.1.0

LINKS:
Homepage
Source Code
Documentation
Bug Tracker
Download
Badge
Subscribe
RSS
Report Abuse

Searching for a Gem on Rubygems

Steps

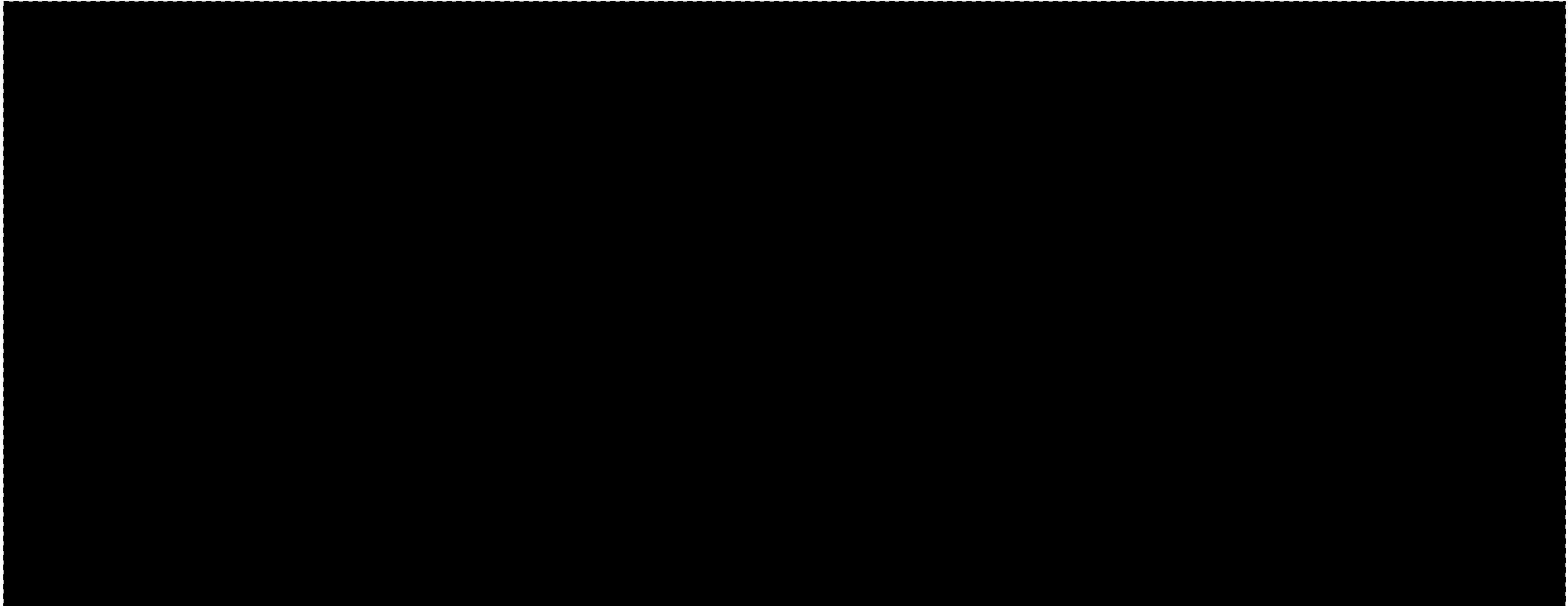
1. Visit <https://rubygems.org>
2. Within the search field enter: **ohai**
3. Press enter or click the magnified glass at the right-side of the search box.
4. Click the 'Source Code' link
5. Click on 'Clone or download' and then copy the git URL.



Returning to the Home Directory



```
> cd ~
```



Cloning the Ohai Library



```
> git clone https://github.com/chef/ohai.git
```

```
Cloning into 'ohai'...
remote: Counting objects: 20571, done.
remote: Compressing objects: 100% (31/31), done.
remote: Total 20571 (delta 7), reused 0 (delta 0), pack-reused 20540
Receiving objects: 100% (20571/20571), 4.46 MiB | 2.13 MiB/s, done.
Resolving deltas: 100% (13408/13408), done.
Checking connectivity... done.
```

Viewing the Contents of the Project



```
> tree ohai
```

```
ohai
├── CHANGELOG.md
├── DOC_CHANGES.md
├── Gemfile
├── LICENSE
├── NOTICE
├── OHAII_MVPS.md
├── README.md
├── RELEASE_NOTES.md
├── Rakefile
├── appveyor.yml
└── bin
```

Viewing the README

~/ohai/README.md

```
# ohai

... PROJECT BADGES ...
```

Description

Ohai detects data about your operating system. It can be used standalone, but its primary purpose is to provide node data to Chef.

Ohai will print out a JSON data blob for all the known data about your system. When used with Chef, that data is reported back via node attributes.

Chef distributes ohai as a RubyGem. This README is for developers who want to modify the Ohai source code. For users who want to write plugins for Ohai, see the docs:

- General documentation: <<https://docs.chef.io/ohai.html>>
- Custom plugin documentation: <https://docs.chef.io/ohai_custom.html>

Viewing the Gem Specification

```
~/ohai/ohai.gemspec
```

```
$:.unshift File.expand_path("../lib", __FILE__)
require "ohai/version"

Gem::Specification.new do |s|
  s.name = "ohai"
  s.version = Ohai::VERSION
  s.platform = Gem::Platform::RUBY
  s.summary = "Ohai profiles your system and emits JSON"
  s.description = s.summary
  s.license = "Apache-2.0"
  s.author = "Adam Jacob"
  s.email = "adam@chef.io"
  s.homepage = "https://docs.chef.io/ohai.html"

  s.required_ruby_version = ">= 2.1.0"

  s.add_dependency "systemu", "~> 2.6.4"
```

Viewing the lib Directory



```
> tree ohai/lib
```

```
ohai/lib
└── ohai
    ├── application.rb
    ├── common
    │   └── dmi.rb
    ├── config.rb
    ...
    └── version.rb
└── ohai.rb
19 directories, 161 files
```

Viewing the ohai.rb file in the lib Directory

~/ohai/lib/ohai.rb

```
#  
# http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
  
require "ohai/version"  
require "ohai/config"  
require "ohai/system"  
require "ohai/exception"
```

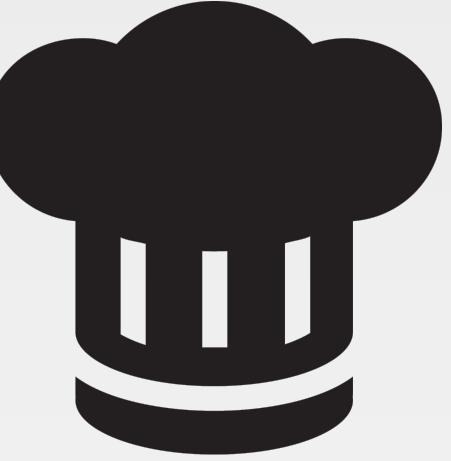
Viewing the plugins directory



```
> tree ohai/lib/ohai/plugins
```

```
ohai/lib/ohai/plugins
├── aix
│   ├── cpu.rb
│   ├── filesystem.rb
│   ├── kernel.rb
│   ├── memory.rb
│   ├── network.rb
│   ├── os.rb
│   ├── platform.rb
│   ├── uptime.rb
│   └── virtualization.rb
└── azure.rb
└── bsd
    └── filesystem.rb
```

EXERCISE



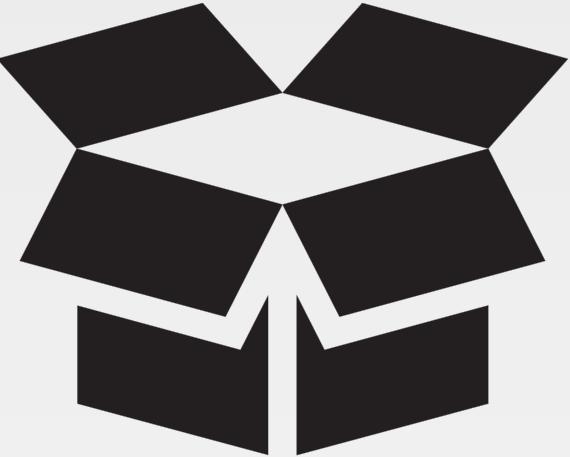
Reviewing the Ohai Gem

Let's take a look at the structure of a plugin.

Objective:

- Review the basic structure of the Ohai gem
- Review the 'language' plugin
- Review the 'python' plugin

CONCEPT



Recent Major Ohai Releases

Ohai 6

Released: April 13, 2011

Chef Version: 0.10.0

docs.chef.io/release/ohai-6

Ohai 7

Released: April 8, 2014

Chef Version: 11.12.0

docs.chef.io/release/ohai-7

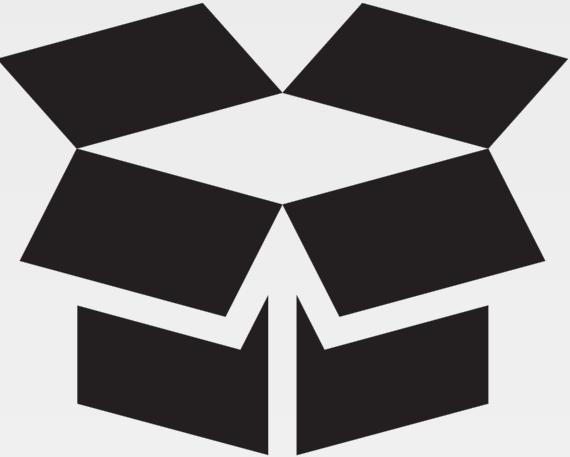
Ohai 8

Released: Dec. 4, 2014

Chef Version: 11.18.0

docs.chef.io/release/ohai-8

CONCEPT



Focus on Ohai 7

Ohai 7 refined the Domain Specific Language (DSL) created in the previous version of Ohai. Ohai 8 continues to use the same language.

Ohai 6

Ohai 7

Ohai 8

Viewing the Languages Plugin

~/ohai/lib/ohai/plugins/languages.rb

```
Ohai.plugin(:Languages) do
  provides "languages"
  collect_data do
    languages Mash.new
  end
end
```

Viewing the Languages Plugin

```
~/ohai/lib/ohai/plugins/languages.rb
```

```
Ohai.plugin(:Languages) do
  provides "languages"
  collect_data do
    languages Mash.new
  end
end
```

Plugin Name

- Ruby Symbol
- First Letter Capitalized

Node attributes provided by the plugin

Code executed on all platforms and stored in the provided attribute(s).

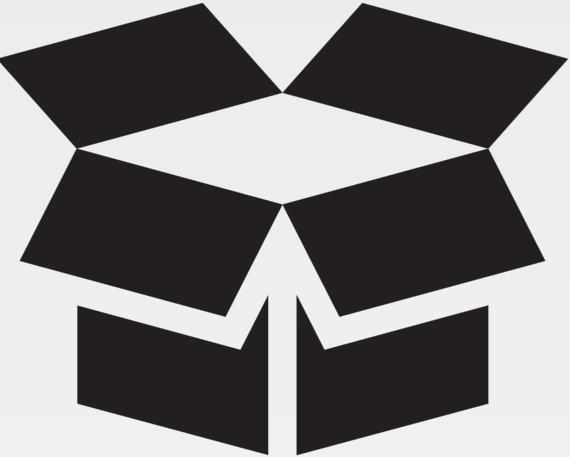
Viewing the Language Plugin

~/ohai/lib/ohai/plugins/languages.rb

```
Ohai.plugin(:Languages) do
  provides "languages"
  collect_data do
    languages Mash.new
  end
end
```

The [Languages](#) plugin provides the node attribute '[languages](#)' which is populated, on all platforms, with a [Mash](#).

CONCEPT



Hash

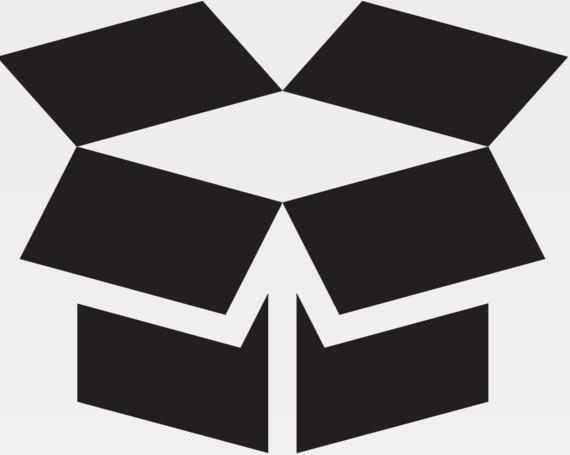
Using a String as a key

```
[1] pry(main)> content = Hash.new  
=> {}  
[2] pry(main)> content['name'] =  
'Chef'  
=> "Chef"  
[3] pry(main)> content['name']  
=> "Chef"  
[4] pry(main)> content[:name]  
=> nil
```

Using a Symbol as a key

```
[1] pry(main)> content = {}  
=> {}  
[2] pry(main)> content[:name] =  
'Chef'  
=> "Chef"  
[3] pry(main)> content[:name]  
=> "Chef"  
[4] pry(main)> content['name']  
=> nil
```

CONCEPT



Mash

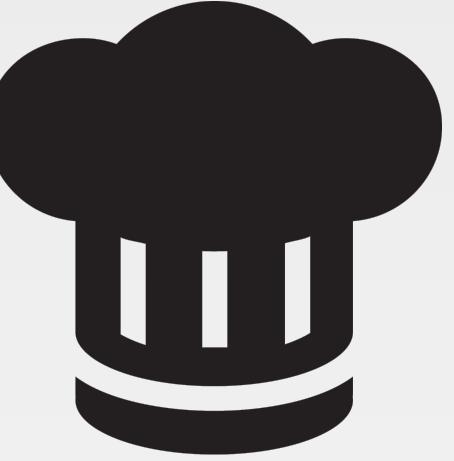
Using a String as a key

```
[1] pry(main)> require 'chef'  
=> true  
  
[2] pry(main)> content = Mash.new  
=> {}  
  
[3] pry(main)> content['name'] = 'Chef'  
=> "Chef"  
  
[4] pry(main)> content['name']  
=> "Chef"  
  
[5] pry(main)> content[:name]  
=> "Chef"
```

Using a Symbol as a key

```
[1] pry(main)> require 'chef'  
=> true  
  
[2] pry(main)> content = Mash.new  
=> {}  
  
[3] pry(main)> content[:name] = 'Chef'  
=> "Chef"  
  
[4] pry(main)> content[:name]  
=> "Chef"  
  
[5] pry(main)> content['name']  
=> "Chef"
```

EXERCISE



Reviewing the Ohai Gem

Now it is time to look at a more complex plugin.

Objective:

- ✓ Review the basic structure of the Ohai gem
- ✓ Review the 'language' plugin
- Review the 'python' plugin

Viewing the Python plugin

~/ohai/lib/ohai/plugins/python.rb

```
Ohai.plugin(:Python) do
  provides "languages/python"
  depends "languages"

  collect_data do
    begin
      so = shell_out("python -c \"import sys; print (sys.version)\"")
      # Sample output:
      # 2.7.11 (default, Dec 26 2015, 17:47:53)
      # [GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)]
      if so.exitstatus == 0
        python = Mash.new
        output = so.stdout.split
        python[:version] = output[0]
        if output.length >= 6
```

Node attributes provided by the plugin

This plugin depends on the attributes in the Languages to be defined

Viewing the Python plugin

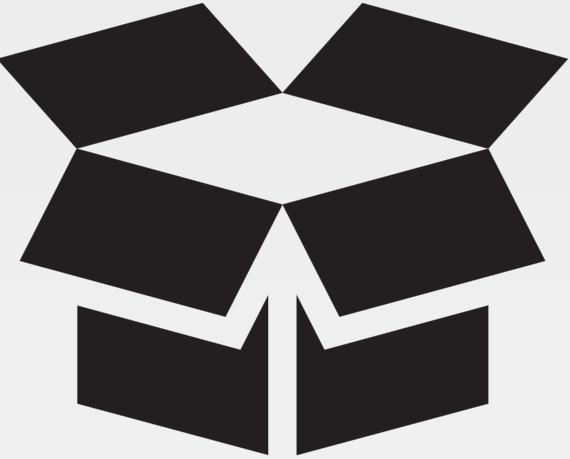
~/ohai/lib/ohai/plugins/python.rb

```
Ohai.plugin(:Python) do
  provides "languages/python"

  depends "languages"

  collect_data do
    begin
      so = shell_out("python -c \"import sys; print (sys.version)\"")
      # Sample output:
      # 2.7.11 (default, Dec 26 2015, 17:47:53)
      # [GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)]
      if so.exitstatus == 0
        python = Mash.new
        output = so.stdout.split
        python[:version] = output[0]
        if output.length >= 6
```

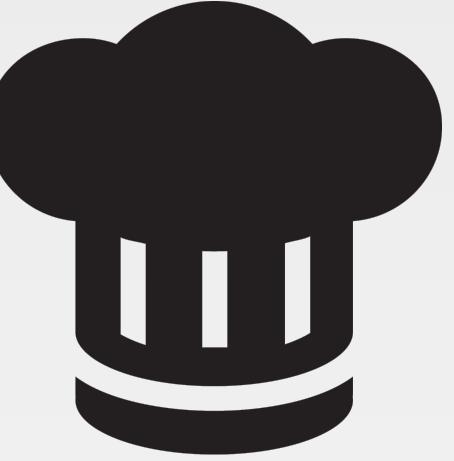
CONCEPT



collect_data for a specific Platform

Plugins can collect data in different ways across different platforms. When defining a `collect_data` block if you do not provide any arguments it is assumed the default and all platforms unless you define a `collect_data` block specific for a platform.

EXERCISE



Reviewing the Ohai Gem

We know where the plugins are located and what they look like. Now it's time to make one.

Objective:

- ✓ Review the basic structure of the Ohai gem
- ✓ Review the 'language' plugin
- ✓ Review the 'python' plugin

DISCUSSION



Discussion

How are the structures of a Rubygem and a Cookbook similar to each other?

What are the requirements when specifying the name of a Ohai plugin?

What is the difference between a Ruby Hash and a Mash?

DISCUSSION



Q&A

What questions can we answer for you?



CHEF™

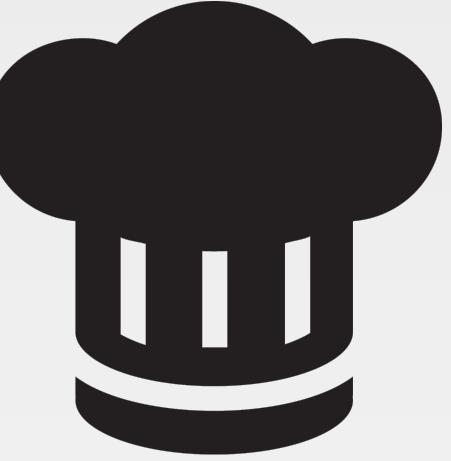
Creating Ohai Plugins

Objectives

After completing this module, you should be able to:

- Create a tested Ohai plugin

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- Define expectations for the Ohai plugin
- Create the Ohai plugin
- Define expectations for the recipe to deliver the Ohai plugin
- Create the recipe that delivers the Ohai plugin
- Define an integration test

Installing the `chefspec-ohai` gem



```
> chef gem install chefspec-ohai
```

```
Successfully installed chefspec-ohai-0.1.0
1 gem installed
```

Adding the Gem to the Spec Helper

~/apache/spec/spec_helper.rb

```
require 'chefspec'  
require 'chefspec/berkshelf'  
require 'chefspec/ohai'
```

Creating a Directory for Plugins Tests



```
> mkdir ~apache/spec/unit/plugins
```

Defining the First Expectation for Plugin

~/apache/spec/unit/plugins/apache_modules_spec.rb

```
require 'spec_helper'

describe_ohai_plugin :Apache do
  let(:plugin_file) { 'files/default/apache_modules.rb' }

  it 'provides apache/modules' do
    expect(plugin).to provides_attribute('apache/modules')
  end
end
```

Executing the Tests to See Failure



```
> chef exec rspec
```

```
F.....
```

Failures:

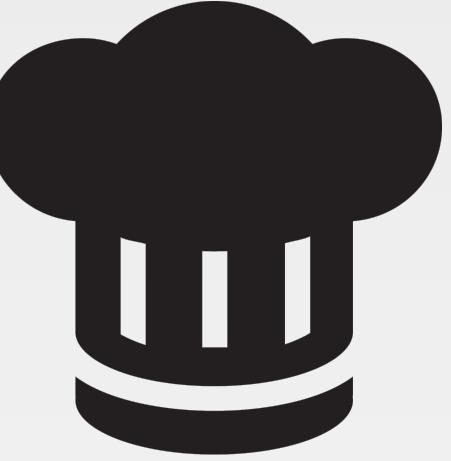
1) Apache provides 'apache/modules'

```
Failure/Error: let(:plugin_source) { File.read(plugin_file) }
```

Errno::ENOENT:

```
No such file or directory @ rb_sysopen -  
files/default/apache_modules.rb
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- Define expectations for the Ohai plugin
- Create the Ohai plugin
- Define expectations for the recipe to deliver the Ohai plugin
- Create the recipe that delivers the Ohai plugin
- Define an integration test

Creating the Plugin as a Cookbook File



```
> chef generate file apache_modules.rb
```

```
Recipe: code_generator::cookbook_file
  * directory[/home/chef/apache/files/default] action create (up to date)
  * template[/home/chef/apache/files/default/apache_modules.rb] action create
    - create new file /home/chef/apache/files/default/apache_modules.rb
    - update content in file /home/chef/apache/files/default/
```

Defining the Plugin that Provides Values

~/apache/files/default/apache_modules.rb

```
Ohai.plugin :Apache do
  provides 'apache/modules'
end
```

Executing the Tests to See Success



```
> chef exec rspec
```

```
.....
```

```
Finished in 5.85 seconds (files took 2.74 seconds to load)  
10 examples, 0 failures
```

Defining an Expectation for Attribute Value

~/apache/spec/unit/plugins/apache_modules_spec.rb

```
require 'spec_helper'

describe_ohai_plugin :Apache do
  let(:plugin_file) { 'files/default/apache_modules.rb' }

  it 'provides apache/modules' do
    expect(plugin).to provides_attribute('apache/modules')
  end

  it 'correctly captures output' do
    allow(plugin).to receive(:shell_out).with('apachectl -t -D
DUMP_MODULES').and_return(double(stdout: 'OUTPUT'))
    expect(plugin_attribute('apache/modules')).to eq('OUTPUT')
  end
end
```

Executing the Tests to See Failure



```
> chef exec rspec
```

```
.F.....
```

Failures:

1) Apache correctly captures output

Failure/Error: expect(plugin_attribute('apache/modules')).to
eq("OUTPUT")

NoMethodError:

undefined method `[]' for nil:NilClass

Capturing the Apache Modules Content

~/apache/files/default/apache_modules.rb

```
Ohai.plugin :Apache do
  provides 'apache/modules'

  collect_data :default do
    apache(Mash.new)
    modules_cmd = shell_out('apachectl -t -D DUMP_MODULES')
    apache[:modules] = modules_cmd.stdout
  end
end
```

Executing the Tests to See Success



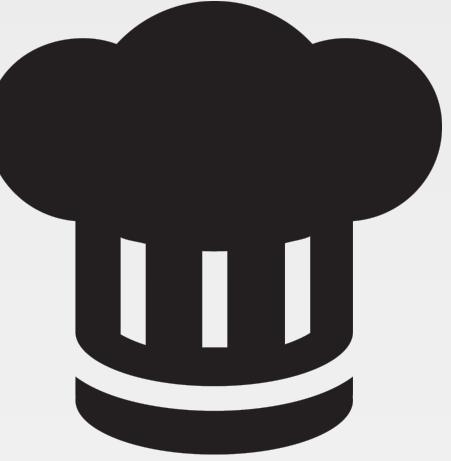
```
> chef exec rspec
```

```
. [2017-10-05T17:00:43-05:00] WARN: Plugin Definition Error:  
<files/default>: collect_data already defined on platform default
```

```
.....
```

```
Finished in 5.84 seconds (files took 3.01 seconds to load)  
11 examples, 0 failures
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the Ohai plugin
- ✓ Create the Ohai plugin
- Define expectations for the recipe to deliver the Ohai plugin
- Create the recipe that delivers the Ohai plugin
- Define an integration test

Adding a Dependency on the Ohai Cookbook

~/apache/metadata.rb

```
name 'apache'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'all_rights'
description 'Installs/Configures apache'
long_description 'Installs/Configures apache'
version '0.1.0'

depends 'ohai'

# If you upload to Supermarket you should set this so your cookbook
# gets a `View Issues` link
# issues_url 'https://github.com/<insert_org_here>/apache/issues' if respond_to?(:issues_url)

# If you upload to Supermarket you should set this so your cookbook
# gets a `View Source` link
# source_url 'https://github.com/<insert_org_here>/apache' if respond_to?(:source_url)
```

Creating the Recipe to Add the Plugin



```
> chef generate recipe ohai_apache_modules
```

```
Recipe: code_generator::recipe
  * directory[/home/chef/apache/spec/unit/recipes] action create
    (up to date)
    * cookbook_file[/home/chef/apache/spec/spec_helper.rb] action
      create_if_missing (up to date)
    *
    template[/home/chef/apache/spec/unit/recipes/ohai_apache_modules_s
      pec.rb] action create_if_missing
    ...
  ...
```

Defining the Expectation Plugin Deployed

```
~/apache/spec/unit/recipes/ohai_apache_modules_spec.rb
```

```
# ... UPPER PORTION OF THE SPECIFICATION ...
let(:chef_run) do
  runner = ChefSpec::ServerRunner.new
  runner.converge(described_recipe)
end

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

it 'installs the modules plugin' do
  expect(chef_run).to create_ohai_plugin('apache_modules')
end
end
end
```

Execute the Tests to See Failure



```
> chef exec rspec
```

```
.....F
```

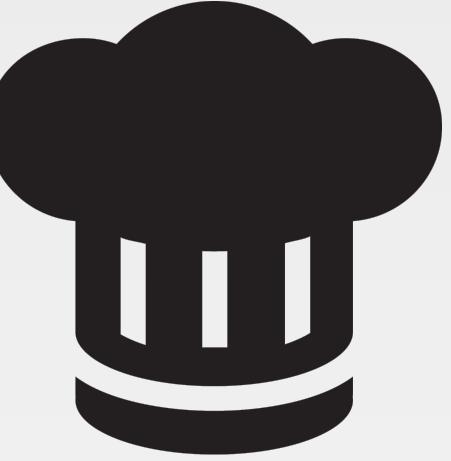
Failures:

```
1) apache::ohai_apache_modules When all attributes are default,  
on an unspecified platform installs the modules plugin
```

```
Failure/Error: expect(chef_run).to  
create_ohai_plugin('apache_modules')
```

```
expected "ohai_plugin[apache_modules]" with action :create  
to be in Chef run. Other ohai_plugin resources:
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the Ohai plugin
- ✓ Create the Ohai plugin
- ✓ Define expectations for the recipe to deliver the Ohai plugin
- Create the recipe that delivers the Ohai plugin
- Define an integration test

Defining the Ohai Plugin in the Recipe

~/apache/recipes/ohai_apache_modules.rb

```
#  
# Cookbook Name:: apache  
# Recipe:: ohai_apache_modules  
  
#  
# Copyright (c) 2017 The Authors, All Rights Reserved.  
include_recipe 'apache::default'  
ohai_plugin 'apache_modules'
```

Executing the Tests to See Success

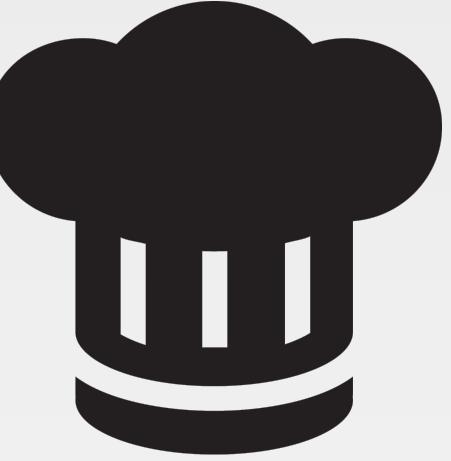


```
> chef exec rspec
```

```
.....
```

```
Finished in 5.77 seconds (files took 2.62 seconds to load)  
12 examples, 0 failures
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the Ohai plugin
- ✓ Create the Ohai plugin
- ✓ Define expectations for the recipe to deliver the Ohai plugin
- ✓ Create the recipe that delivers the Ohai plugin
- Define an integration test

Appending the New Recipe to the Suite

~/apache/.kitchen.yml

```
# ... TOP OF KITCHEN CONFIGURATION ...

suites:
  - name: default
    run_list:
      - recipe[apache::default]
      - recipe[apache::ohai_apache_modules]
    verifier:
      inspec_tests:
        - test/smoke/default
    attributes:
```

Converging the Updated Default Suite



```
> kitchen converge
```

```
-----> Starting Kitchen (v1.11.1)
-----> Converging <default-centos-67>...
```

```
      resolving cookbooks for run list: ["apache::default",
"apache::ohai_apache_modules"]
```

Synchronizing Cookbooks:

- ohai (4.2.2)
- apache (0.1.0)
- compat_resource (12.14.7)

Installing Cookbook Gems:

Compiling Cookbooks...

Recipe: apache::ohai apache modules

Defining an Expectation for the Plugin

~/apache/test/smoke/default/ohai_apache_modules.rb

```
plugin_directory = '/tmp/kitchen/ohai/plugins'

describe command("ohai -d #{plugin_directory} apache") do
  its(:stdout) { should match(/core_module/) }
end
```

Verifying the New Expectation



```
> kitchen verify
```

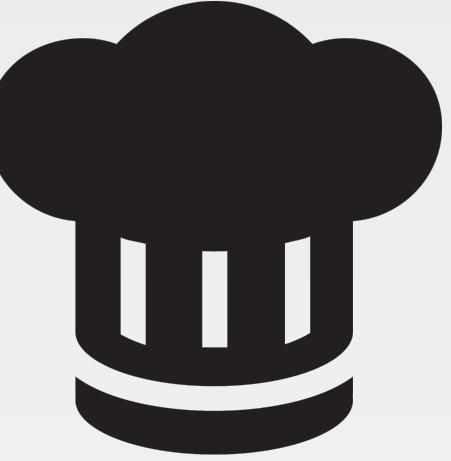
```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <ohai-plugin-centos-67>...
      Use `/home/chef/apache/test/smoke/default` for testing
```

```
Target: ssh://kitchen@localhost:32768
```

```
✓ Command ohai -d /tmp/kitchen/ohai/plugins apache stdout
should match /core_module \static\ /
```

```
Summary: 1 successful, 0 failures, 0 skipped
```

EXERCISE



Creating an Ohai Plugin

Being able to learn more about our nodes will make our reporting more powerful and benefit the recipes we write.

Objective:

- ✓ Define expectations for the Ohai plugin
- ✓ Create the Ohai plugin
- ✓ Define expectations for the recipe to deliver the Ohai plugin
- ✓ Create the recipe that delivers the Ohai plugin
- ✓ Define an integration test

DISCUSSION



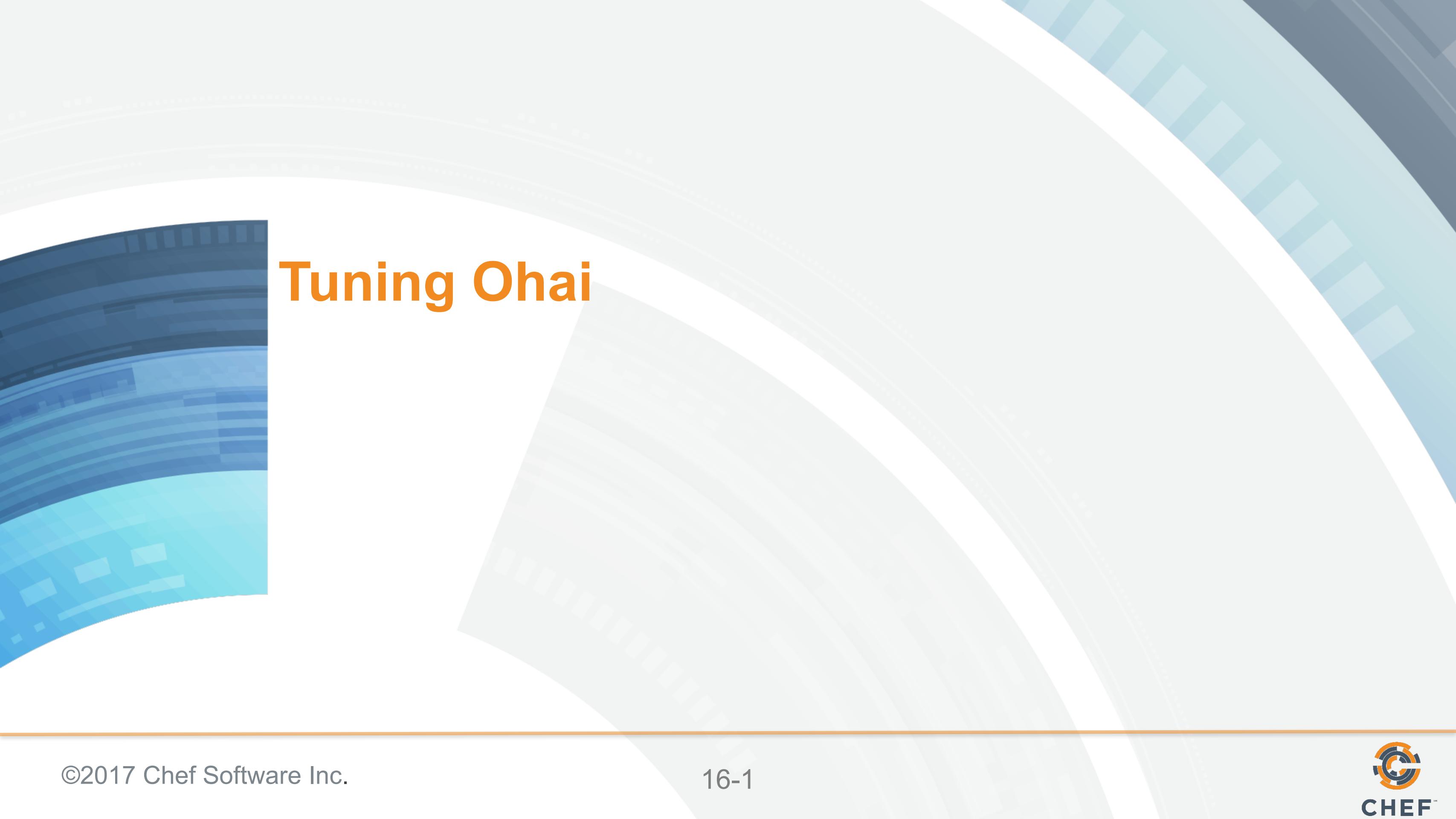
Q&A

What questions can we answer for you?



CHEF™





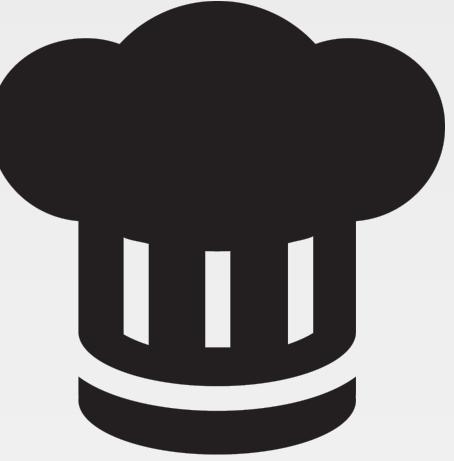
Tuning Ohai

Objectives

After completing this module, you should be able to:

- Describe how you configure the node to automatically load ohai plugins
- Describe how to enable ohai hints
- Describe how to remove plugins that you do not want executed
- Describe where you can find more information about better Ohai performance

EXERCISE



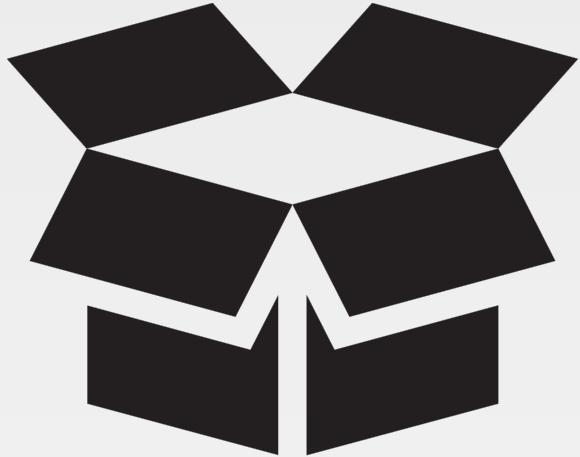
Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- Configure the node to automatically load ohai plugins
- Enable ohai hints
- Remove plugins that you do not want executed
- Choose only the plugins you want executed

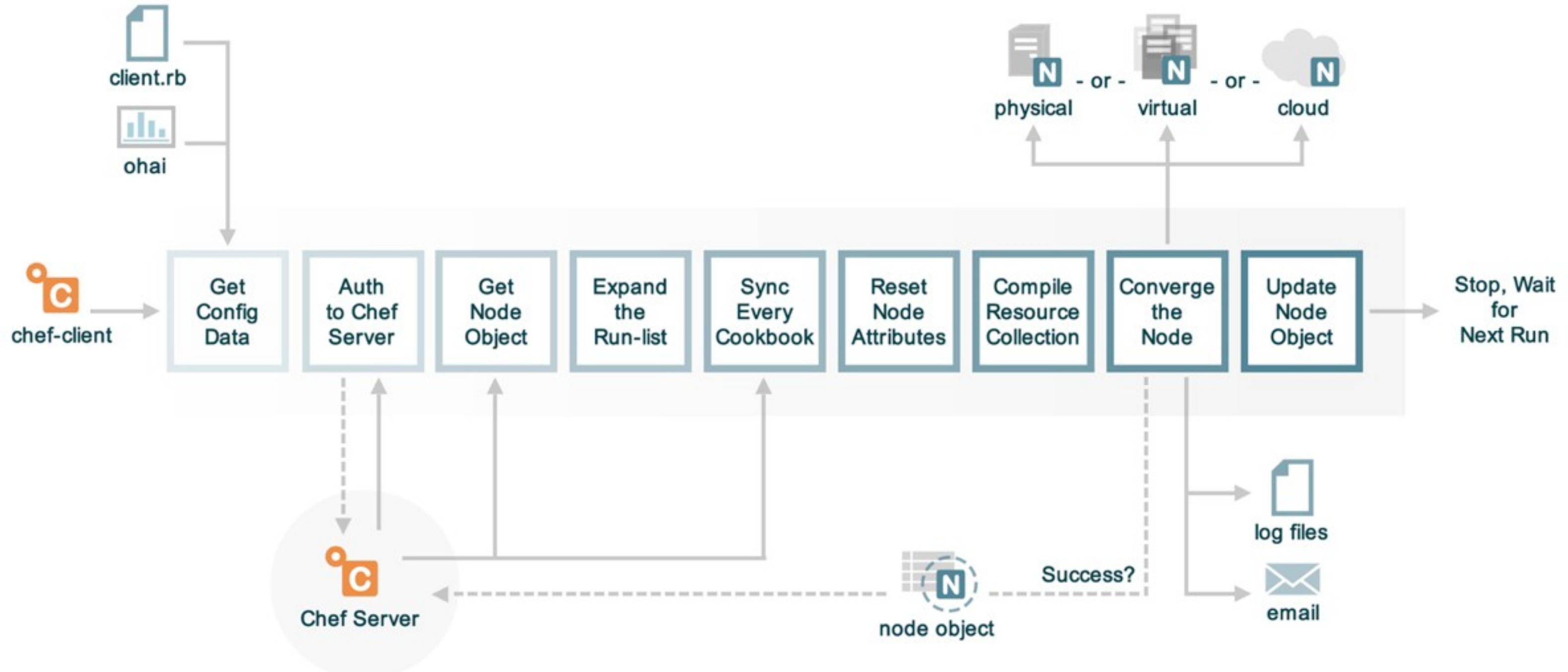
CONCEPT



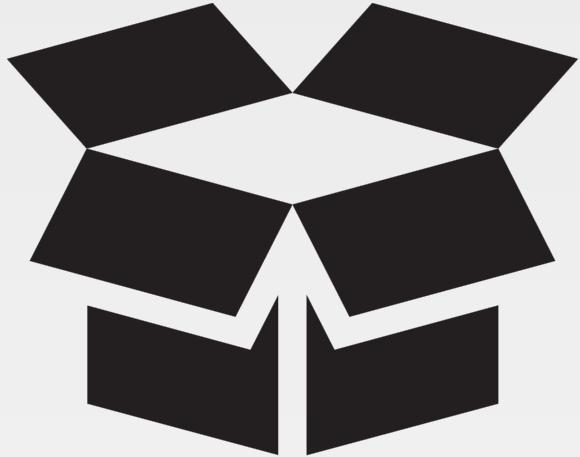
Node Configuration File

All nodes have a configuration file that contain details about node, overrides, and other data.

The Anatomy of a chef-client Run



CONCEPT



chef-client cookbook

The chef-client cookbook contains a number of useful recipes:

- **service (default)**
Run chef-client as a service, converging at a periodic interval
- **delete_validation**
Remove the organization validation key [SECURITY ISSUE]
- **config**
Define node configuration

<https://supermarket.chef.io/cookbooks/chef-client>

Viewing the chef-client config Recipe

~/cookbooks/chef-client/recipes/config.rb

```
# ... OTHER RESOURCES ...
template "#{node['chef_client']['conf_dir']}/client.rb" do
  source 'client.rb.erb'
  owner d_owner
  group d_group
  mode 00644
  variables(
    :chef_config => node['chef_client']['config'],
    :chef_requires => chef_requires,
    :ohai_disabled_plugins => node['ohai']['disabled_plugins'],
    :start_handlers => node['chef_client']['config']['start_handlers'],
    :report_handlers => node['chef_client']['config']['report_handlers'],
    :exception_handlers => node['chef_client']['config']['exception_handlers']
  )
  notifies :create, 'ruby_block[reload_client_config]', :immediately
end
```



~/cookbooks/chef-client/templates/default/client.rb.erb

```
<% if node.attribute?("ohai") && node["ohai"].attribute?("plugin_path") -%>

Ohai::Config[:plugin_path] << "<%= node["ohai"]["plugin_path"] %>"

<% end -%>

<% unless @ohai_disabled_plugins.empty? -%>
Ohai::Config[:disabled_plugins] = [<%= @ohai_disabled_plugins.map { |k| k... <% end -%>
```

PROBLEM



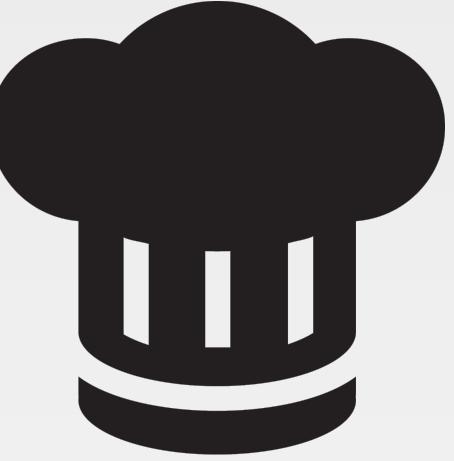
The Work

- Add the chef-client cookbook to your cookbook collection
- Provide attributes in a wrapper cookbook, role, or environment for:

```
node['ohai']['plugin_path']
```

- Add chef-client cookbook to every node's run list

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- Configure the node to automatically load ohai plugins
- Enable ohai hints
- Remove plugins that you do not want executed
- Choose only the plugins you want executed

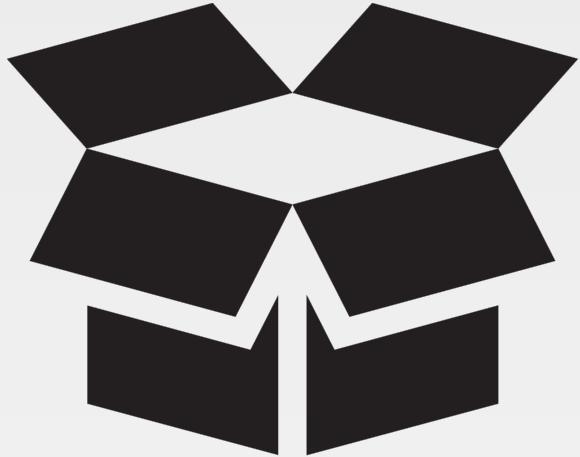
CONCEPT



Not all Node Plugins are Executed

Some of the plugins will not automatically run against your node. This is particularly true of the cloud provider plugins. As the node does not often know the environment in which it is being run.

CONCEPT



Ohai Hints

For those plugins that are not executed you can leave them a hint file that will ensure they are executed.

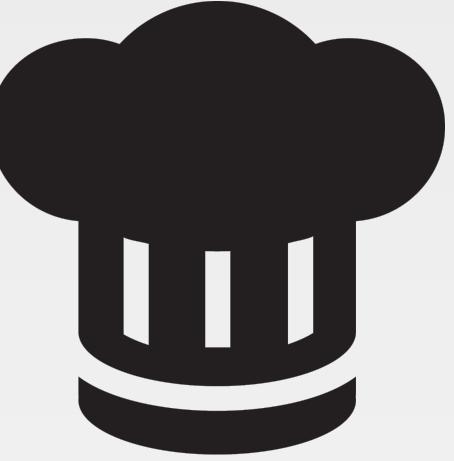
PROBLEM



The Work

- Add the ohai cookbook to your cookbook collection
- Define a recipe that uses the ohai cookbook's `ohai_hint` resource
- Add this recipe to every node's run list

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ✓ Enable ohai hints
- Remove plugins that you do not want executed
- Choose only the plugins you want executed



~/cookbooks/chef-client/templates/default/client.rb.erb

```
<% if node.attribute?("ohai") && node["ohai"].attribute?("plugin_path") -%>

Ohai::Config[:plugin_path] << "<%= node["ohai"]["plugin_path"] %>"
<% end -%>

<% unless @ohai_disabled_plugins.empty? -%>
Ohai::Config[:disabled_plugins] = [<%= @ohai_disabled_plugins.map { |k| k... %>

<% end -%>
```

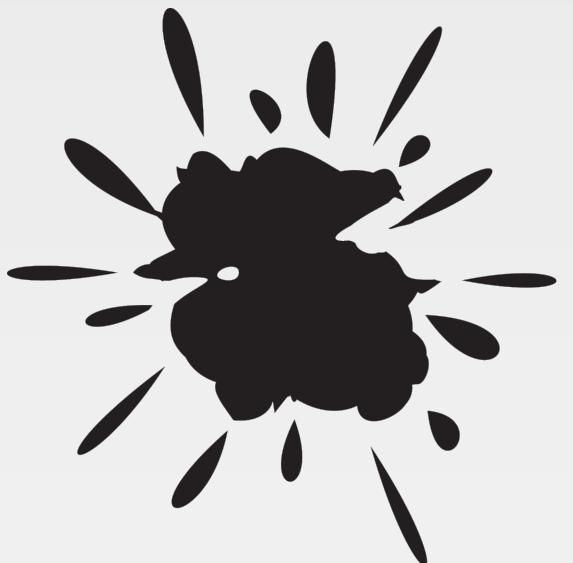
Viewing the chef-client config Recipe

~/cookbooks/chef-client/recipes/config.rb

```
# ... OTHER RESOURCES ...

template "#{node['chef_client']['conf_dir']}/client.rb" do
  source 'client.rb.erb'
  owner d_owner
  group d_group
  mode 00644
  variables(
    :chef_config => node['chef_client']['config'],
    :chef_requires => chef_requires,
    :ohai_disabled_plugins => node['ohai']['disabled_plugins'],
    :start_handlers => node['chef_client']['config']['start_handlers'],
    :report_handlers => node['chef_client']['config']['report_handlers'],
    :exception_handlers => node['chef_client']['config']['exception_handlers']
  )
  notifies :create, 'ruby_block[reload_client_config]', :immediately
end
```

PROBLEM



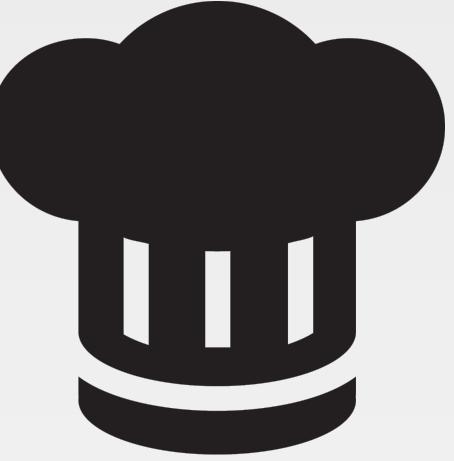
The Work

- Add the chef-client cookbook to your cookbook collection
- Select attributes you want to remove
- Find the name of the plugin that provides those attributes
- Provide attributes in a wrapper cookbook, role, or environment for:

```
node [ 'ohai' ] [ 'disabled_plugins' ]
```

- Add chef-client cookbook to every node's run list

EXERCISE



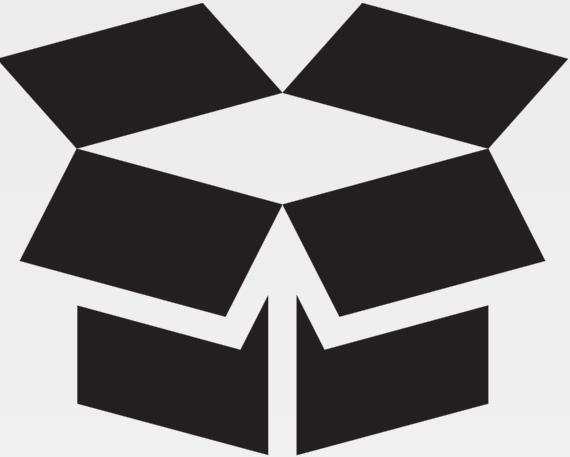
Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ✓ Enable ohai hints
- ✓ Remove plugins that you do not want executed
- Choose only the plugins you want executed

CONCEPT

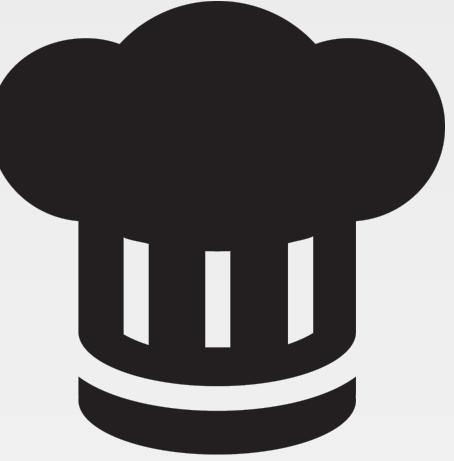


Whitelist Node Attributes

When it seems like you are disabling far too many plugins and you want to consider attacking it from the other side:

<http://ckbk.it/whitelist-node-attrs>

EXERCISE



Run Ohai Smoother

There are a few more things to learn about Ohai that could help increase performance.

Objective:

- ✓ Configure the node to automatically load ohai plugins
- ✓ Enable ohai hints
- ✓ Remove plugins that you do not want executed
- ✓ Choose only the plugins you want executed

DISCUSSION



Q&A

What questions can we answer for you?



CHEF™

