Welcome to Eloquent Tests.

Franklin Webber

Training and Technical Content Lead

US Army - Radio/COMSEC Repairer (35E)

Tech Support

Quality Assurance

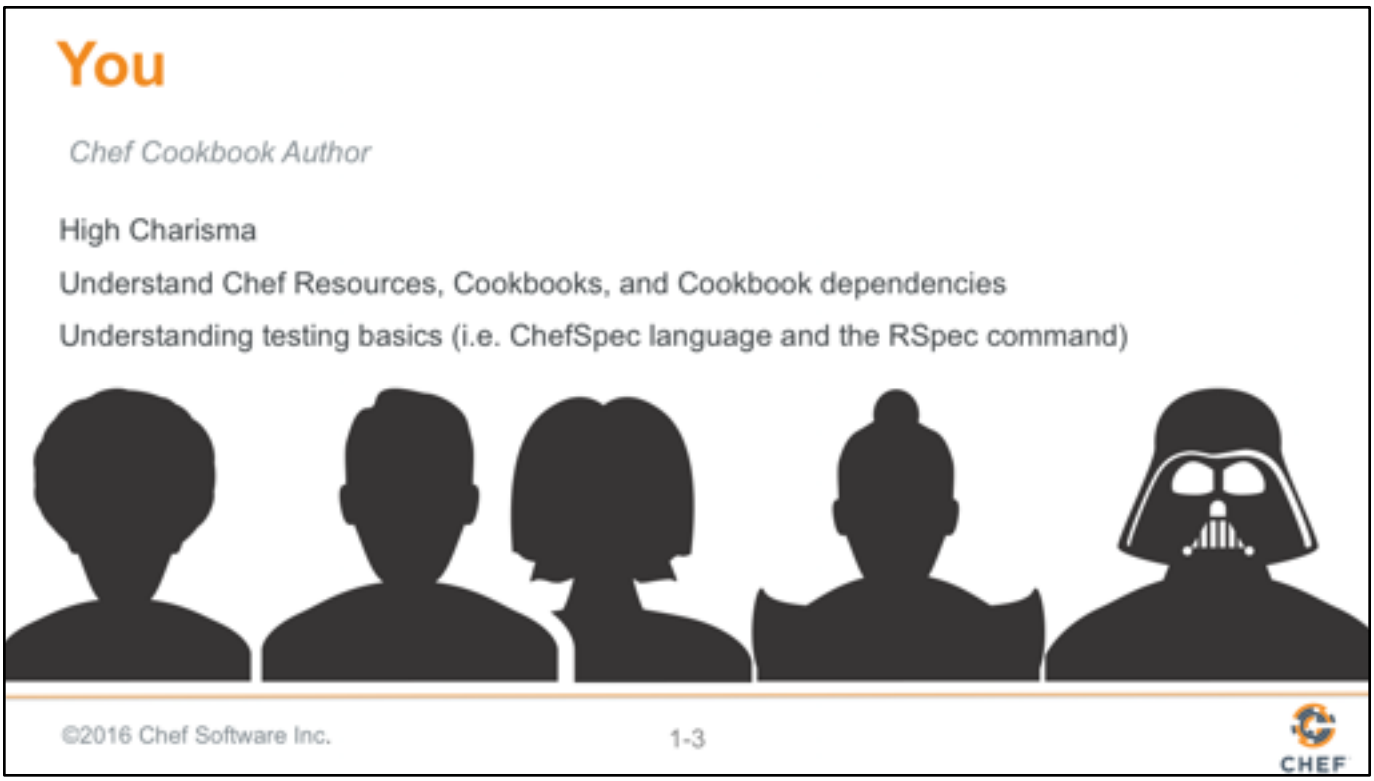
Software Developer Eng. in Test

Software Developer

Instructor / Trainer

CHEF

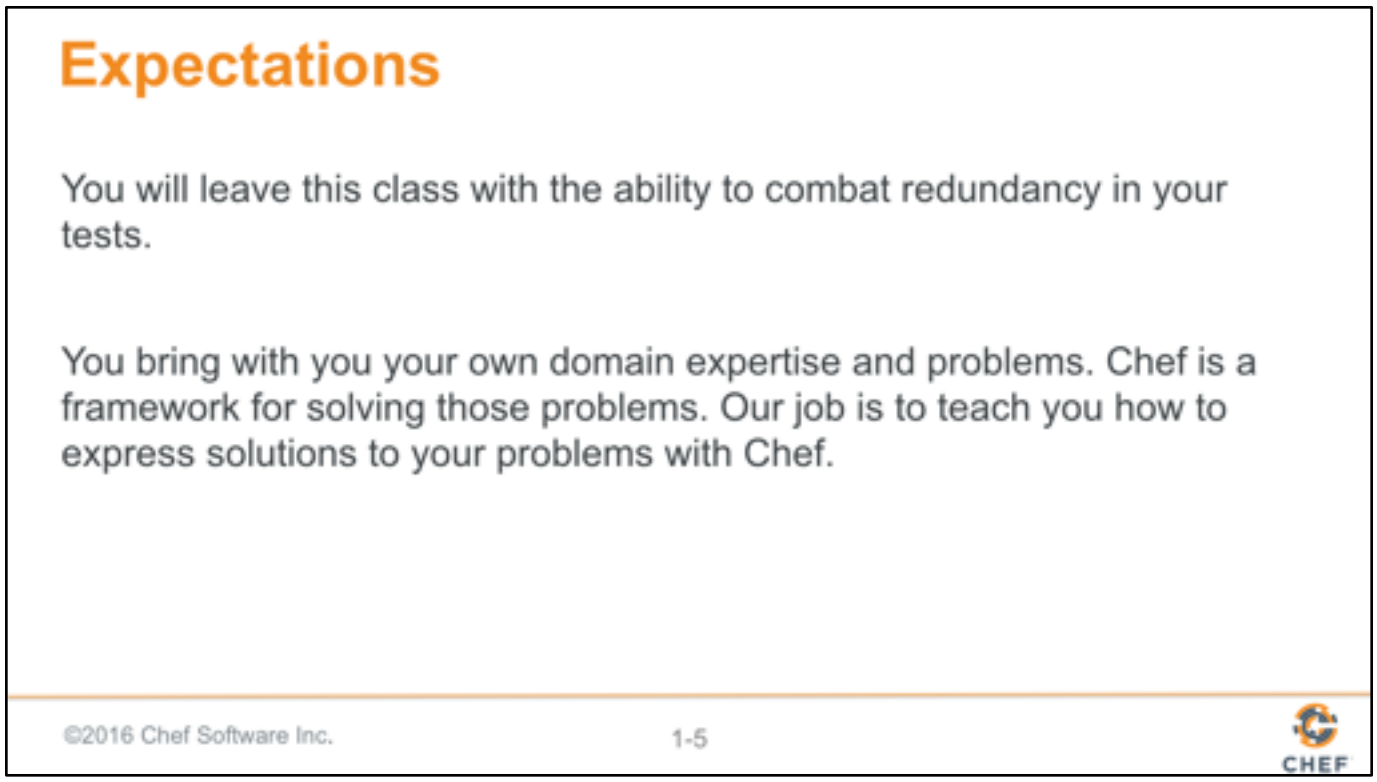
Before we start let me introduce myself.



Let me tell who I think you are and for whom this content has been designed.

Before we start let me introduce myself. Then I would like it if everyone had a chance to introduce themselves.

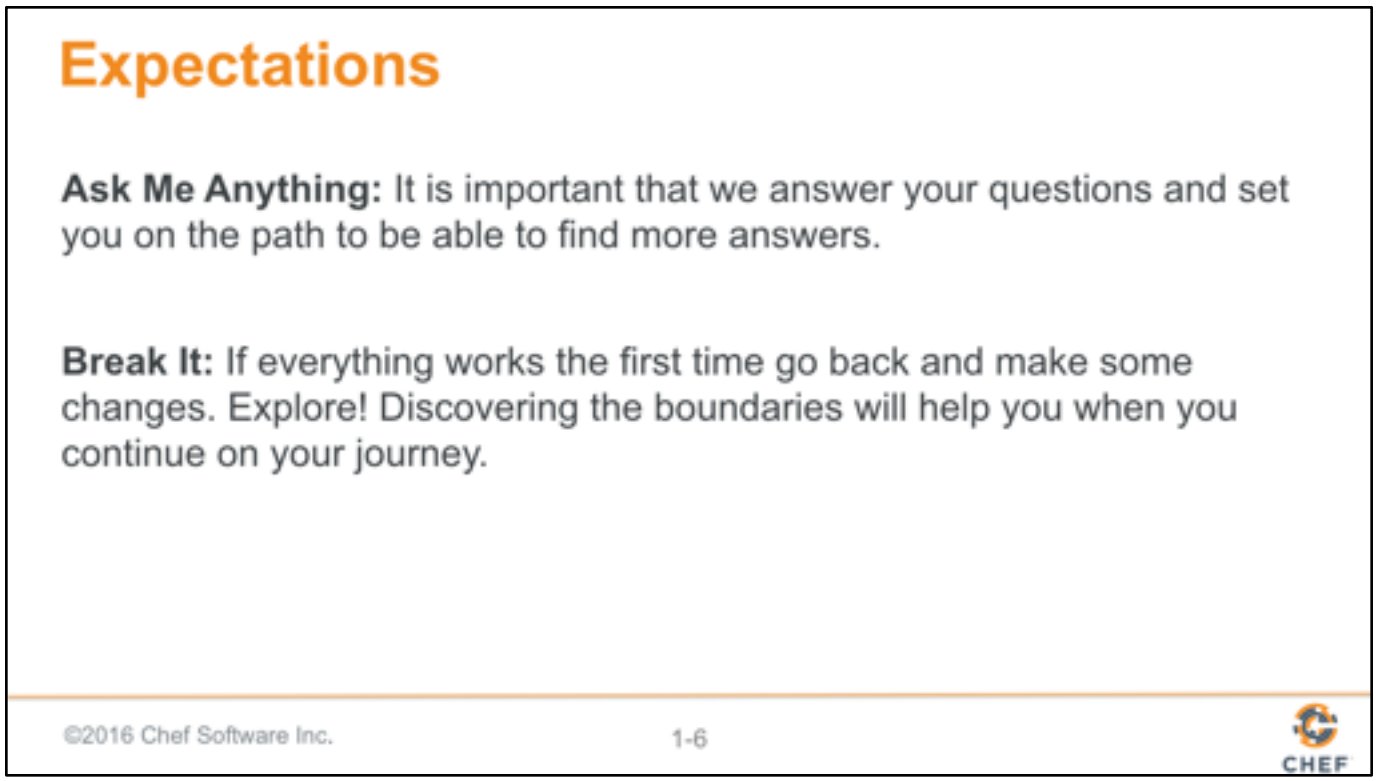
Instructor Note: Often times with larger, in-person groups I prefer to have the individuals perform this introduction one-on-one. Having people leave their desks and greet as many people as they can during the time allotted. I often feel this works better as it removes the pressure from the single individual to introduce themselves in a way that is presenting themselves and not actually greeting people. When Online, I create a pre-defined order, announce that order, and then invite a person to speak, thank them when they are done.



The goal of this training is to teach you techniques that will combat the redundancy in your test suite and then extract those helpers into a Ruby gem.

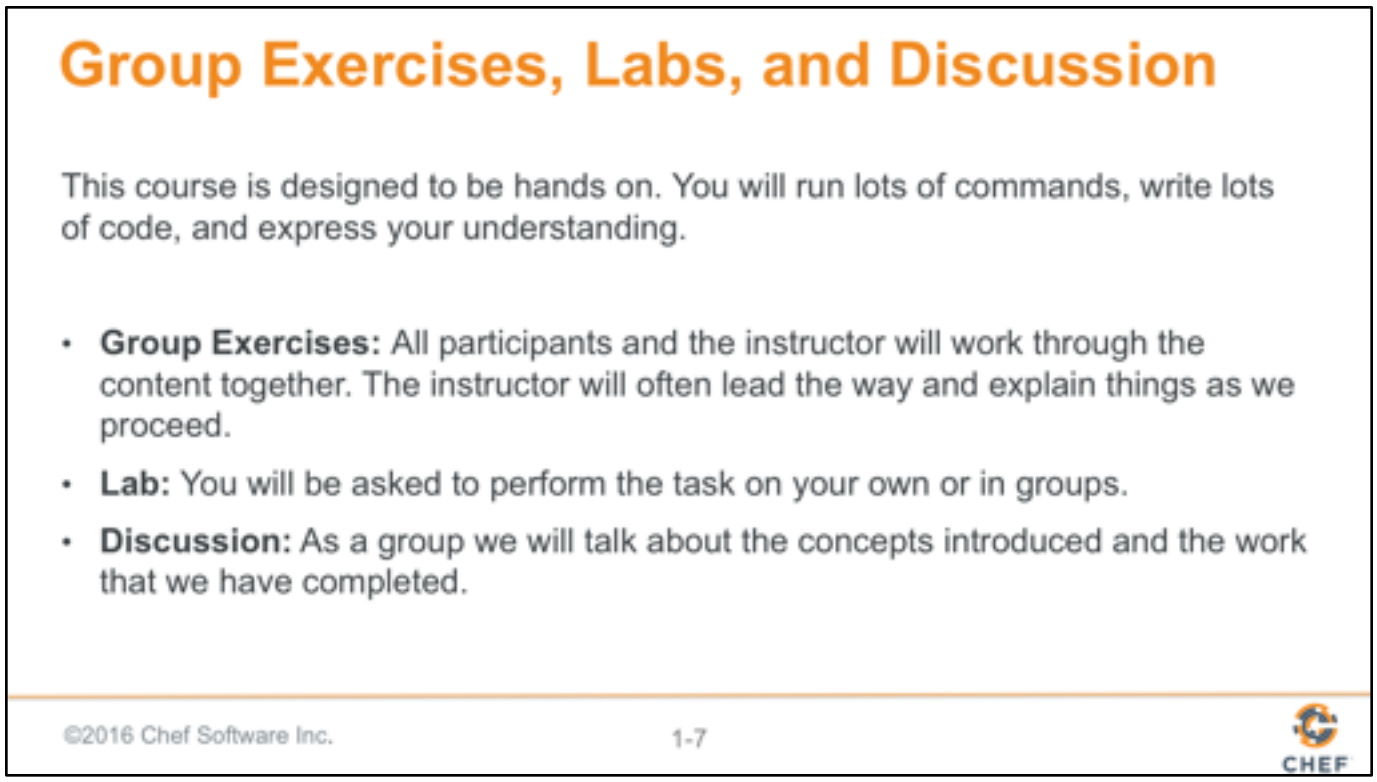
Chef is built on top of Ruby. The testing tools built on top of RSpec. This means you have the power of a programming language at your disposal and we will have to keep a tight focus on the challenges and exercises presented in this content. During and throughout the content we will have discussion where we may have additional time to talk about many different topics but in this interest of time and popular opinion we may need to leave those discussions.

During the introductions you learned about the other individuals here in the course with you. They may have shared similar problems and domains. During the time that we are here respectfully reach out them so that you can continue the conversation, grow each others' knowledge, and become better professionals.

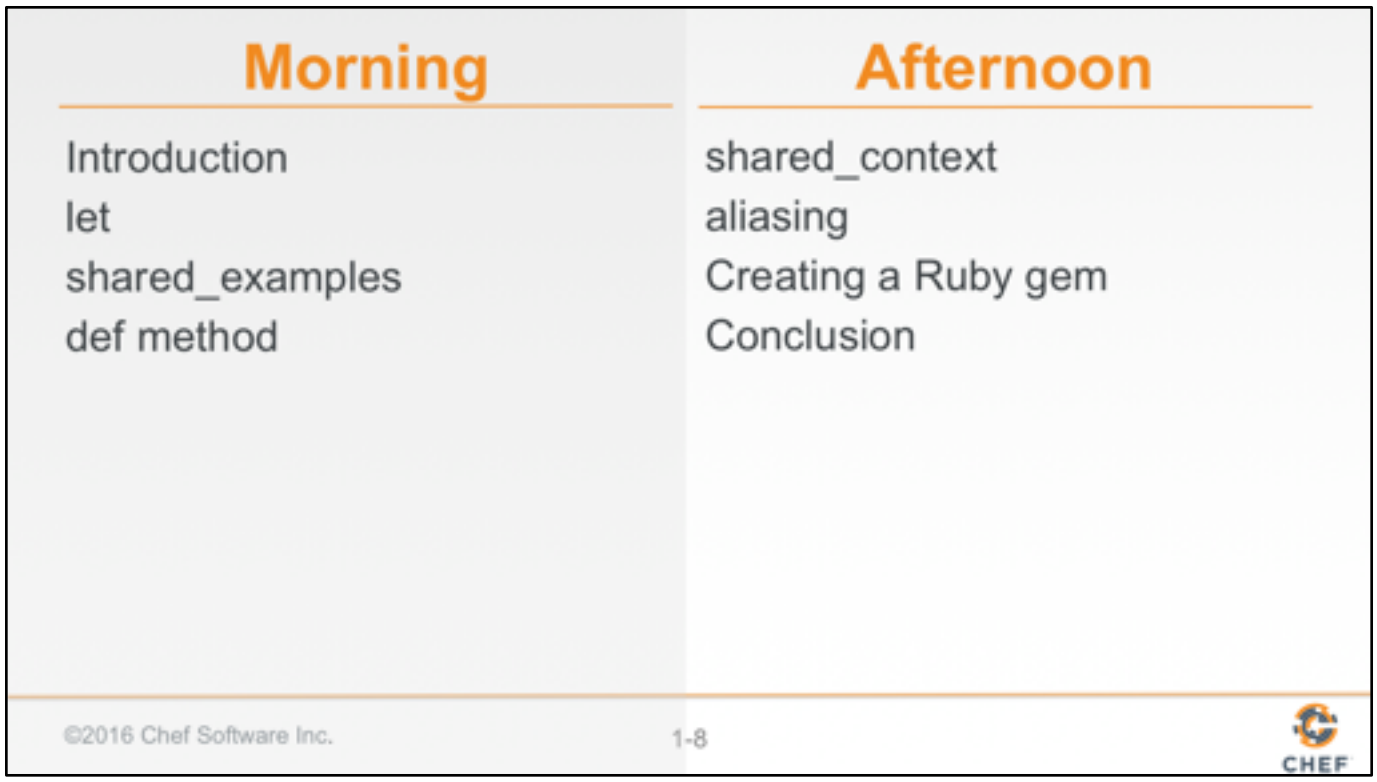


All throughout this training I strongly encourage you to ask questions whenever you do not understand a topic, an acronym, concept, or software. By asking a question you better your learning and often times better the learning of those with you in this training. Asking questions is a sign of curiosity that we want to encourage and foster while we are here together.

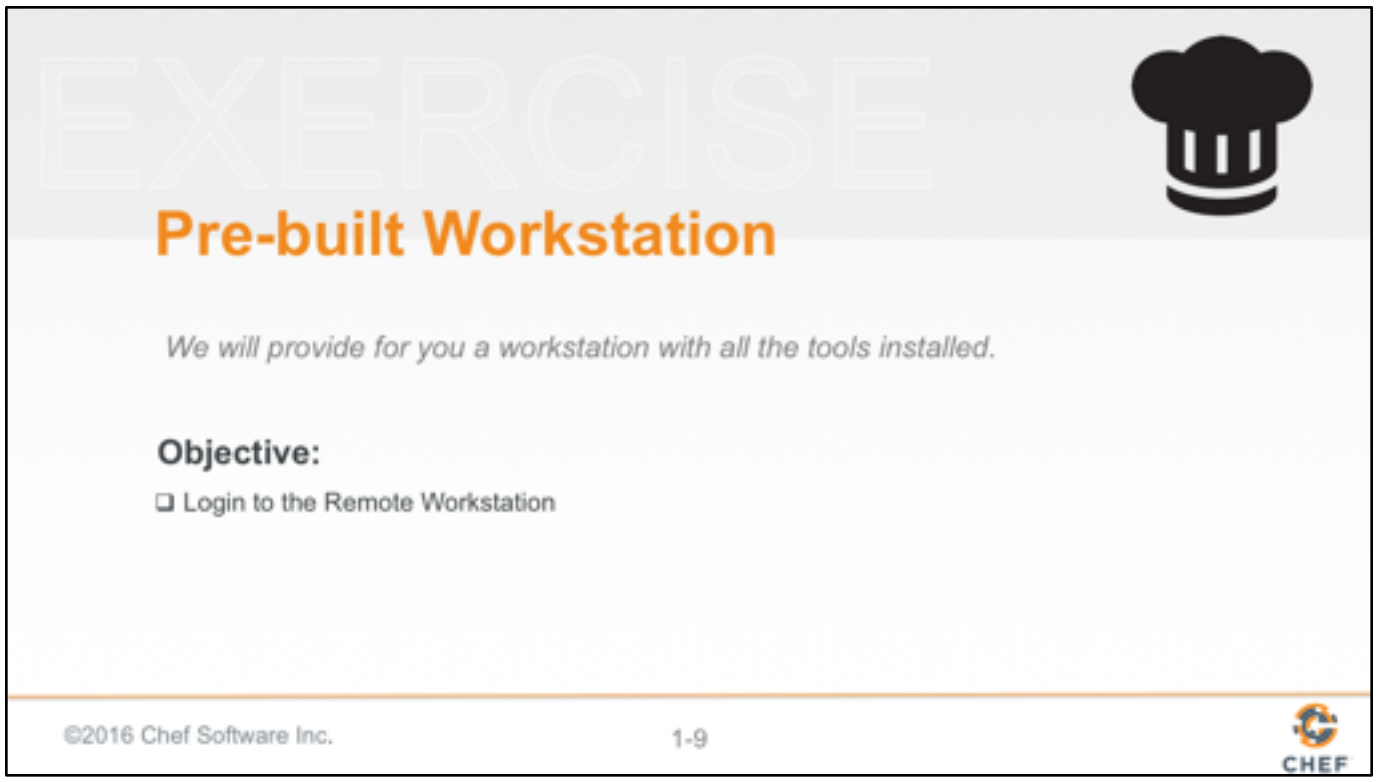
This curiosity can also be employed by exploring the boundaries of the tools you are using and the language you are writing. The exercises and the labs we will perform will often lead you through examples that work from the beginning to the end. When you develop solutions it is rare that something works from the start all the way to the end. Errors and issues come up from typos or the incorrect usage of a command of the programming language. When you fall off the path it can often be hard to find your way back. Here, if you find yourself always on the correct path explore what happens when you step off of it, what you see, the error messages you are presented with, the new results you might find.



The content of this training has been designed in a way to emphasize this hands-on approach to the content. Together, we will perform exercises together that accomplish an understood objective. After that is done you will often emphasize an activity by performing a lab. The lab is designed to challenge your understanding and retention of the previously accomplished exercises. You can work through this labs on your own or in groups. After completing the labs we will all come together again to review the exercise. Finally, we will end each section with a discussion about the topics that we introduced. These discussions will often ask you to share your opinions, recent experiences, or previous experiences within this domain.



This is the outline of the events for this training. Please take a moment to review this list to ensure that the topics listed here meet your expectations. Take a moment to note which topics are of most interest to you. Also note which topics are not present here on this list. We will discuss your thoughts at the end of the section.

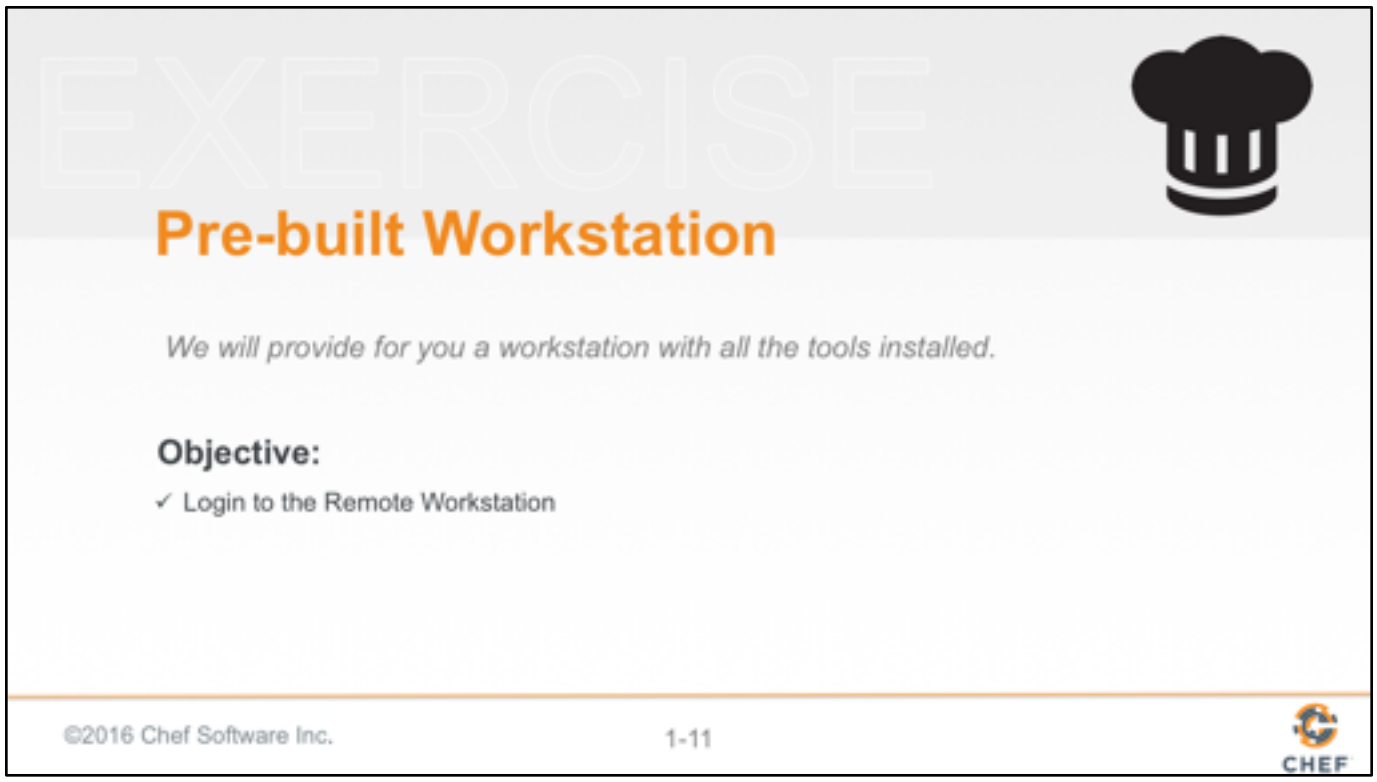


As I mentioned there is a lot work planned for the day. To ensure we focus on the concepts we introduce and not on troubleshooting systems we are providing you a workstation with the necessary tools installed to get started right away.

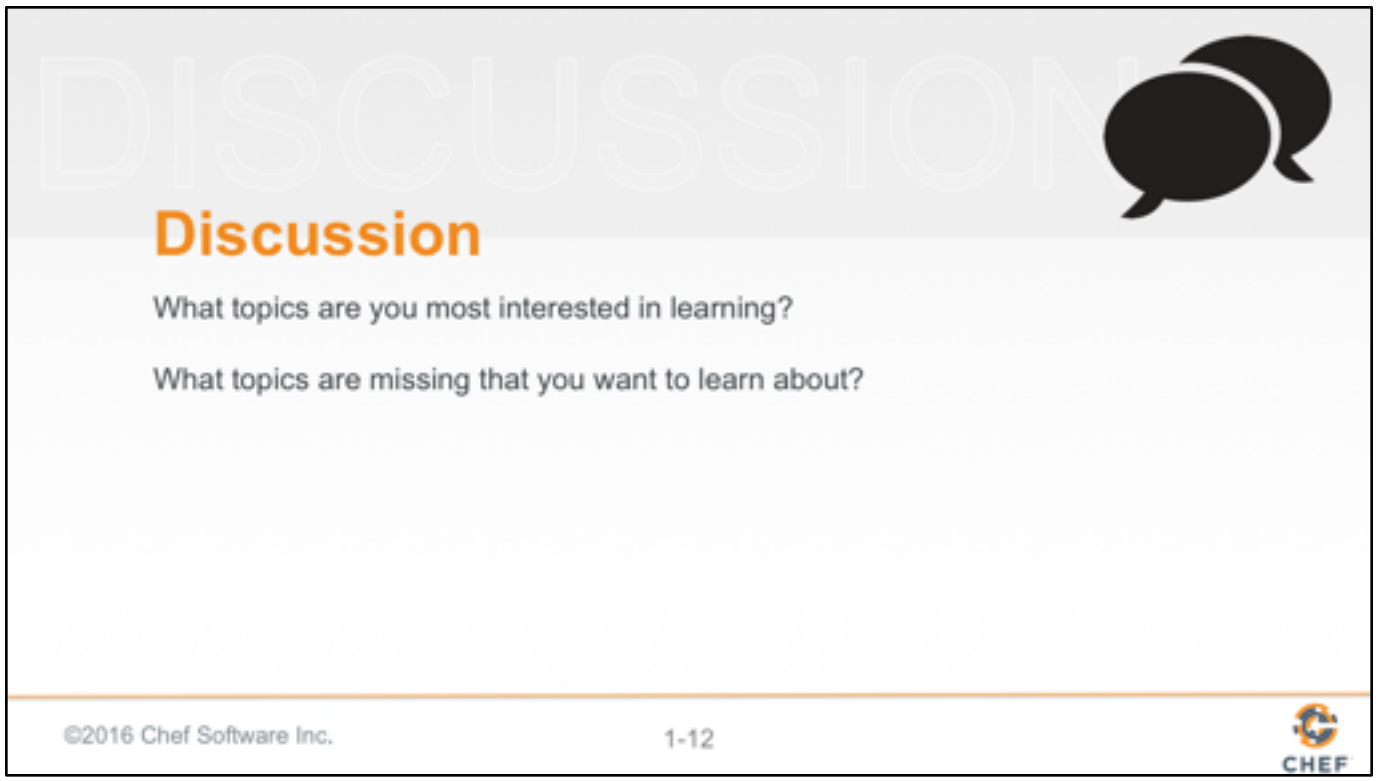
Instructor Note: At the end of the training it is often a good idea to offer your services to help individuals install necessary software or troubleshoot their systems.

I will provide you with the address, username and password of the workstation. With that information you will need to use the SSH tool that you have installed to connect that workstation.

This demonstrates how you might connect to the remote machine using your terminal or command-prompt if you have access to the application ssh. This may be different based on your operating system.



Now that you are connected to that workstation we have taken care of all the necessary work to get started with the training.



Let us end with a discussion about the following topics.

Instructor Note: With large groups I often find it better to have individuals turn to the individuals around them, form groups of whatever size they feel comfortable, and have them take turns asking and answering the questions. When all the groups are done I then open the discussion up to the entire group allowing each group or individuals to share their answers.

Before we continue let us stop for a moment answer any questions that anyone might have at this time.


CHEF™


code review
noun
1. Code review is systematic examination (sometimes referred to as peer review) of computer source code. It is intended to find mistakes overlooked in the initial development phase, improving the overall quality of software. Reviews are done in various forms such as ...
©2016 Chef Software Inc.
2-1
CHEF

Objective

Read. Evaluate. Judge.

CHEF

code review

I finished all the tests for the Ark cookbook! Code review?

Any thoughts on how to make the tests clearer?

CHEF

Changing into the Cookbook Directory

```
> cd ~/ark
```


CHEF

Executing the Test Suite

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

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

Finished in 3.58 seconds (files took 2.73 seconds to load)
19 examples, 0 failures


ChefSpec Coverage report generated...

  Total Resources:   41
  Touched Resources: 41
  Touch Coverage:    100.0%
```

Viewing the Recipe Specification

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

```
require 'spec_helper'

describe 'ark::default' do
  context 'when no attributes are specified, on an unspe...orm' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end

    it 'installs necessary packages' do
```


CHEF

EXERCISE

Code Review

- ❑ Form a group
- ❑ Review the code
- ❑ Create a list of feedback

CHEF

Sharing Feedback

Each team will take turns sharing a single item of feedback on the code.

CHEF

DISCUSSION

If You Could Only Choose One

Each team will choose a single, most crucial item of feedback and share it.

CHEF


CHEF™

let

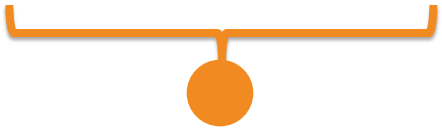
verb

1. not prevent or forbid; allow.
2. memoized helper method that allows you to express your tests succinctly.

CHEF

Objective

Use **`let`** *to express the tests succinctly.*



CHEF

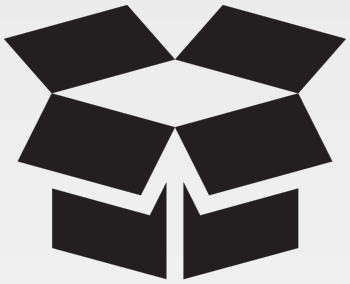
let

"Since we cannot change reality, let us change the eyes which see reality."
~ Nikos Kazantzakis

- ❑ **Concepts**
- ❑ **Demonstration**
- ❑ **Review**
- ❑ **Exercise**

CHEF

CONCEPT



`let`

Use let to define a memoized helper method.

The value will be cached across multiple calls in the same example but not across examples. It is also lazy-evaluated: it is not evaluated until the first time the method it defines is invoked. See `let!` if you want to force the invocation before each example.

https://goo.gl/BJp0lQ

CHEF

Diagramming the let Helper Method

```
describe 'ark::default' do
  context 'when no attributes are ...
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
      runner
    end

    it 'installs necessary packages' do
      expect(chef_run).to install_p...
      expect(chef_run).to install_p...
    end

    it "does not install the gcc-c+...
      expect(chef_run).not_to insta...
    end
```

1. let is a RSpec helper method
2. Ruby Symbol
3. Code Block
4. Invocation

Invoking the Helper Method

```
describe 'ark::default' do
  context 'when no attributes are ...
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
      runner
    end

    it 'installs necessary packages' do
      expect(chef_run).to install_p...
      expect(chef_run).to install_p...
    end

    it "does not install the gcc-c+...
      expect(chef_run).not_to insta...
    end
```

1. chef_run sends a message
2. RSpec invokes the contents of the block
3. RSpec stores the contents of the execution
4. chef_run sends a message
5. RSpec retrieves the stored execution

Cached Within each Example

```
describe 'ark::default' do
  context 'when no attributes are ...
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
      runner
    end

    it 'installs necessary packages' do
      expect(chef_run).to install_p...
      expect(chef_run).to install_p...
    end

    it "does not install the gcc-c+...
      expect(chef_run).not_to insta...
    end
```

1. chef_run is loaded and stored
2. chef_run uses the stored invocation
3. chef_run is loaded and stored

A chef_run with Node Attributes

```
describe 'ark::default' do
  context 'when no attributes are specified, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new({ platform: 'centos',
                                          version: '6.7' })
      runner.converge(described_recipe)
    end

    # ... EXAMPLES WITHIN CONTEXT
  end
end
```

let

"If the path be beautiful, let us not ask where it leads."

~ Anatole France

- ✓ **Concepts**
- ❑ **Demonstration**
- ❑ **Review**
- ❑ **Exercise**

DEMO

Live Demonstration

https://goo.gl/ChkP47

CHEF

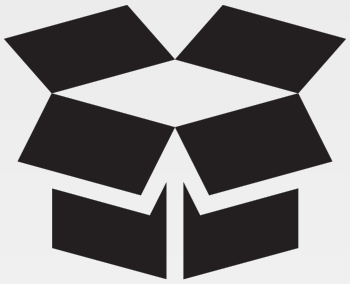
let

"Let us always meet each other with smile, for the smile is the beginning of love."
~ Mother Teresa

- ✓ **Concepts**
- ✓ **Demonstration**
- ❑ **Review**
- ❑ **Exercise**

CHEF

CONCEPT



Using let for clarity

The use of the let to create the chef_run saves us from having the write the same code over-and-over again within each example.

We can define our own let helpers to increase the readability of our test code. Extracting important details and giving them a name.

https://goo.gl/BJp0IQ

CHEF

A chef_run with Node Attributes

```
describe 'ark::default' do
  context 'when no attributes are specified, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new({ platform: 'centos',
                                          version: '6.7' })
      runner.converge(described_recipe)
    end

    # ... EXAMPLES WITHIN CONTEXT
  end
end
```

Using let to Create Clearer Examples

```
describe 'ark::default' do
  context 'when no attributes are specified, on CentOS' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new(node_attributes)
      runner.converge(described_recipe)
    end

    let(:node_attributes) do
      { platform: 'centos', version: '6.7' }
    end

    # ... EXAMPLES WITHIN CONTEXT
  end
```

let

"Let us not pray to be sheltered from dangers but to be fearless when facing them."

~ Rabindranath Tagore

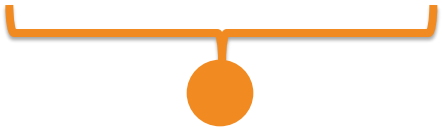
- ✓ **Concepts**
- ✓ **Demonstration**
- ✓ **Review**
- ❑ **Exercise**

CHEF

Exercise

Refactor. Execute the Tests.

Find Success.



CHEF

let

"Come, let us have some tea and continue to talk about happy things."

~ Chaim Potok

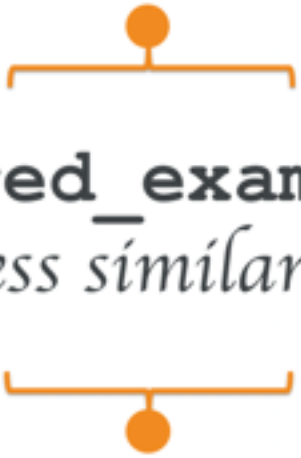
- ✓ **Concepts**
- ✓ **Demonstration**
- ✓ **Review**
- ✓ **Exercise**

CHEF

CHEF™


shared_examples
noun
1. one of a number of things, or a part of something, taken to show the character of the whole.
2. describe similar behaviors in different contexts.
©2016 Chef Software Inc.
3-1
CHEF

Objective



Use **`shared_examples`** *to express similarities.*

shared_examples

- ❑ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

CONCEPT

shared_examples

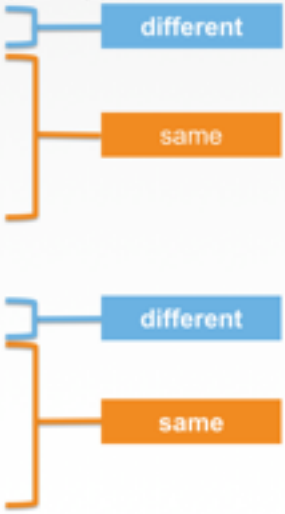
Shared examples let you describe behavior of classes or modules. When declared, a shared group's content is stored. It is only realized in the context of another example group, which provides any context the shared group needs to run.

https://goo.gl/yi12tM

CHEF

Finding Similar Expressed Expectations

```
context 'when no attributes are specified, on an unspecif...orm' do
  let(:installed_packages) ...
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
context 'when no attributes are specified, on CentOS' do
  let(:installed_packages) ...
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
```



CHEF

shared_examples

- ✓ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

DEMO

Live Demonstration

https://goo.gl/ChkP47

CHEF

shared_examples

- ✓ Concepts
- ✓ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

CONCEPT

shared_examples

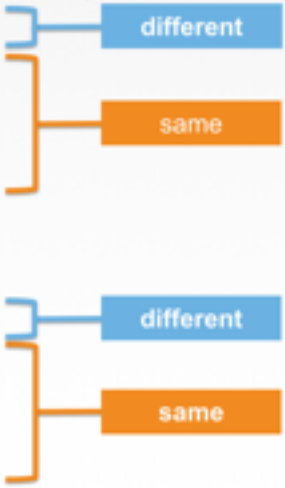
Shared examples let you describe behavior of classes or modules. When declared, a shared group's content is stored. It is only realized in the context of another example group, which provides any context the shared group needs to run.

https://goo.gl/yi12tM

CHEF

Finding Similar Expressed Expectations

```
context 'when no attributes are specified, on an unspecif...orm' do
  let(:installed_packages) ...
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
context 'when no attributes are specified, on CentOS' do
  let(:installed_packages) ...
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
```



CHEF

A Place to Share Examples

```
shared_examples 'installs packages' do
  it 'installs the necessary packages' do
    installed_packages.each do |name|
      expect(chef_run).to install_package(name)
    end
  end
end
context 'when no attributes are specified, on an unspecif...orm' do
  let(:installed_packages) ...
  it_behaves_like 'installs packages'
end
context 'when no attributes are specified, on CentOS' do
  let(:installed_packages) ...
  it_behaves_like 'installs packages'
end
```

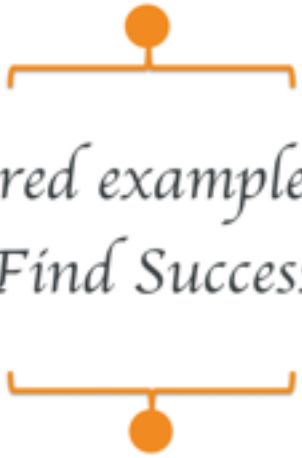

CHEF

shared_examples

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ❑ Exercise

CHEF

Exercise



Find shared examples; extract.
Find Success.

CHEF

davidd @ https://flic.kr/p/DyTeW1

shared_examples

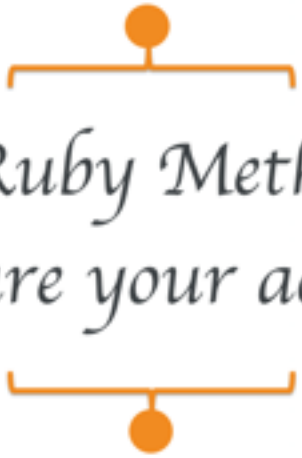
- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ✓ Exercise

CHEF


CHEF™


def method
Rub
y
1. not prevent or forbid; allow.
2. memoized helper method that allows you to express your tests succinctly.
©2016 Chef Software Inc.
5-1
CHEF

Objective



Use Ruby Methods to capture your actions.

CHEF

def method

- ❑ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

CONCEPT

require

Loads the given name, returning true if successful and false if the feature is already loaded.

If the filename does not resolve to an absolute path, it will be searched for in the directories listed in $LOAD_PATH ($:).

http://goo.gl/cLKY37

CHEF

CONCEPT

Create helper method

You can define a Ruby method that takes care of some of the tedious work of retrieving node attributes.

CHEF

A Ruby Method with One Parameter

```ruby
def file(name)
  # contents of method
  # last line automatically returns the value
end
```

The method named 'file' has a single parameter named 'name'.

CHEF

def method

- ✓ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

DEMO

Live Demonstration

https://goo.gl/9mRNlD
(lowercase L)

CHEF

def method

- ✓ Concepts
- ✓ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

CONCEPT

require

Loads the given name, returning true if successful and false if the feature is already loaded.

If the filename does not resolve to an absolute path, it will be searched for in the directories listed in $LOAD_PATH ($:).

http://goo.gl/cLKY37

Requiring the spec_helper file

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

```
require 'spec_helper'

describe 'ark::default' do
  context 'when no attributes are specified, on an ...platform' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new
      runner.converge(described_recipe)
    end
```


CHEF

Viewing the spec_helper file

~/ark/spec/spec_helper.rb

```
require 'chefspec'
require 'chefspec/berkshelf'

at_exit { ChefSpec::Coverage.report! }

RSpec.configure do |config|
  config.color = true
end

# puts $LOAD_PATH
# ... define test helpers and content in this file
```


CHEF

Viewing the $LOAD_PATH

```
> chef exec rspec
```

```
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/uuidtools-2.1.5/lib
/Users/franklinwebber/ark/spec
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-support-3.4.1/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-core-3.4.4/lib
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/net-ssh-3.1.1/lib
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/fauxhai-3.5.0/lib
/opt/chefdk/embedded/lib/ruby/gems/2.1.0/gems/diff-lcs-1.2.5/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-expectations-3.4.0/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-mocks-3.4.1/lib
/Users/franklinwebber/.chefdk/gem/ruby/2.1.0/gems/rspec-3.4.0/lib
```


CHEF

CONCEPT

Create helper method

You can define a Ruby method that takes care of some of the tedious work of retrieving node attributes.

CHEF

A Ruby Method with One Parameter

```
def file(name)
  # contents of method
  # last line automatically returns the value
end
```

The method named 'file' has a single parameter named 'name'.

CHEF

Viewing the Repetition of Retrieving Attributes

```
it "apache mirror" do
  attribute = chef_run.node['ark']['apache_mirror']
  expect(attribute).to eq "http://apache.mirrors.tds.net"
end

it "prefix root" do
  attribute = chef_run.node['ark']['prefix_root']
  expect(attribute).to eq "/usr/local"
end
```

Refactoring with let to help ease the pain

```
let(:node) do
  chef_run.node
end

it "apache mirror" do
  attribute = node['ark']['apache_mirror']
  expect(attribute).to eq "http://apache.mirrors.tds.net"
end

it "prefix root" do
  attribute = node['ark']['prefix_root']
  expect(attribute).to eq "/usr/local"
end
```

Implementing a Helper Method

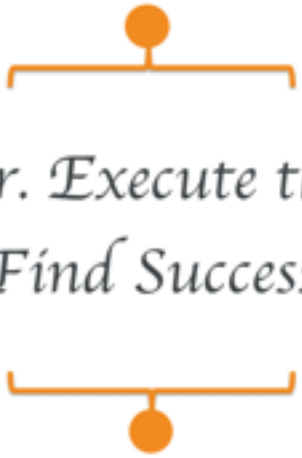
```
let(:node) do
  chef_run.node
end

def attribute(name)
  node[described_cookbook][name]
end
it "apache mirror" do
  expect(attribute('apache_mirror')).to eq "http://apache.mirr....net"
end

it "prefix root" do
  expect(attribute('prefix_root')).to eq "/usr/local"
end
```

def method

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ❑ Exercise

CHEF

Exercise



Refactor. Execute the Tests.
Find Success.

CHEF

def method

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ✓ Exercise

CHEF


CHEF™

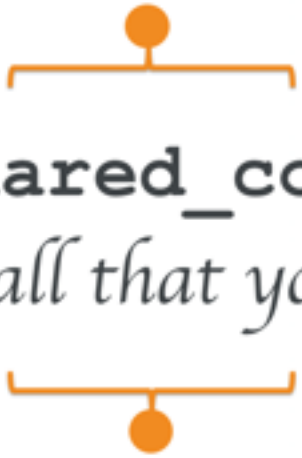
shared_context

noun

1. the parts of a written or spoken statement that precede or follow a specific word or passage, usually influencing its meaning or effect.
2. define a block that will be evaluated in the context of example groups.

CHEF

Objective



Use **`shared_context`**
to save all that you wrote.

CHEF

shared_context

- ❑ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

Ray of shared_context

Use shared_context to define a block that will be evaluated in the context of example groups either explicitly, using include_context, or implicitly by matching metdata.

http://goo.gl/R0ujTA

shared_context

- ✓ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

Live Demonstration

https://goo.gl/9mRNlD
(lowercase L)

CHEF

shared_context

- ✓ Concepts
- ✓ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

Ray of shared_context

Use shared_context to define a block that will be evaluated in the context of example groups either explicitly, using include_context, or implicitly by matching metdata.

http://goo.gl/R0ujTA

CHEF

Repeating More than Examples

```
describe 'ark::default' do
  context 'when no attributes are specified, on an uns...platform' do
    let(:chef_run) do
      runner = ChefSpec::SoloRunner.new(node_attributes)
      runner.converge(described_recipe)
    end

    let(:node_attributes) do
      {}
    end

    def attribute(name) ...
```


CHEF

Repeating More than Examples

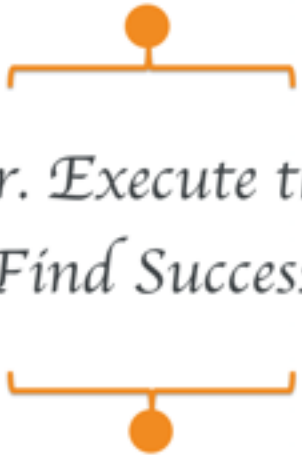
```
shared_context 'converged recipe' do
 let(:chef_run) do
   runner = ChefSpec::SoloRunner.new(node_attributes)
   runner.converge(described_recipe)
 end

  let(:node_attributes) do
    {}
  end

  def attribute(name) ...
end

describe 'ark::default' do
  context 'when no attributes are specified, on an uns...platform' do
    include_context 'converged recipe'
```

shared_context

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ❑ Exercise

CHEF

Exercise



Refactor. Execute the Tests.
Find Success.

CHEF

shared_context

Motivational thing to say

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ✓ Exercise

CHEF


CHEF™

alias_example_group_to

noun

1. a false name used to conceal one's identity; an assumed name.
2. define a new name for an example group that optionally can be given metadata to load a shared_context

CHEF

Objective

Eloquence comes to automatically assuming context.

alias_example_group_to

- ❑ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

alias_example_group_to

describe and **context** are the default aliases for **example_group**. You can define your own aliases for **example_group** and give those custom aliases default metadata.

http://goo.gl/DfUChL

CHEF

alias_example_group_to

- ✓ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

Live Demonstration

https://goo.gl/9mRNlD
(lowercase L)

CHEF

alias_example_group_to

- ✓ Concepts
- ✓ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

CONCEPT

alias_example_group_to

describe and **context** are the default aliases for **example_group**. You can define your own aliases for **example_group** and give those custom aliases default metadata.

http://goo.gl/DfUChL

CHEF

Viewing a Spec with include_context

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

```
describe 'ark::default' do
  context 'when no attributes are specified, on an ...platform' do
    include_context 'converged recipe'

    # ... examples ...

  end

  # ... other contexts ...
end
```

Refactoring the Spec to auto include the context

```
describe_recipe 'ark::default' do
  context 'when no attributes are specified, on an uns...platform' do
    include_context 'converged recipe'

    # ... examples ...

  end

  # ... other contexts
end
```

Defining the alias and tying it to the shared_context

~/ark/spec/spec_helper.rb

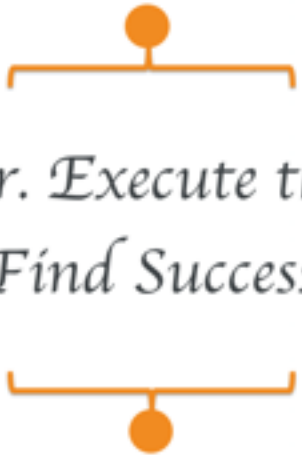
```
require 'chefspec'
require 'chefspec/berkshelf'

at_exit { ChefSpec::Coverage.report! }

RSpec.configure do |config|
  config.color = true
  config.alias_example_group_to :describe_recipe, :type => :recipe
end

shared_context 'converged recipe', :type => :recipe do
```

alias_example_group_to

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ❑ Exercise

CHEF

Exercise



Refactor. Execute the Tests.
Find Success.

CHEF

alias_example_group_to

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ✓ Exercise

CHEF


CHEF™

Ruby Gem

noun

1. a cut and polished precious stone fine enough for use in jewelry.
2. is a package manager for the Ruby programming language that provides a standard format for distributing Ruby programs and libraries ...

CHEF

Objective

CHEF

Creating a Ruby Gem

- ❑ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

CONCEPT

RubyGems

RubyGems is a package manager for the Ruby programming language that provides a standard format for distributing Ruby programs and libraries (in a self-contained format called a "gem"), a tool designed to easily manage the installation of gems, and a server for distributing them.

http://guides.rubygems.org/rubygems-basics

CHEF

gem

```
> chef gem --help
```

The command allows you build gems for distribution, install gems locally, and push gems to Gem server.

http://guides.rubygems.org/command-reference

CHEF

CONCEPT

Bundler

Bundler provides a consistent environment for Ruby projects by tracking and installing the exact gems and versions that are needed.

Bundler is an exit from dependency hell, and ensures that the gems you need are present in development, staging, and production. Starting work on a project is as simple as bundle install.

http://bundler.io

CHEF


CONCEPT
bundle
> chef exec bundle --help
The command allows you to install and update a project's dependencies. It will also allow you to generate a cookbook.
©2016 Chef Software Inc.
8-7
CHEF

Creating a Ruby Gem

- ✓ Concepts
- ❑ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

DEMO

Live Demonstration

CHEF

Creating a Ruby Gem

- ✓ Concepts
- ✓ Demonstration
- ❑ Review
- ❑ Exercise

CHEF

Returning to the home directory

CHEF

Generating the Gem

```
> chef exec bundle gem chefspec-myhelpers
```

```
Creating gem 'chefspec-myhelpers'...
      create  chefspec-myhelpers/Gemfile
      create  chefspec-myhelpers/.gitignore
      create  chefspec-myhelpers/lib/chefspec/myhelpers.rb
      create  chefspec-myhelpers/lib/chefspec/myhelpers/version.rb
      create  chefspec-myhelpers/chefspec-myhelpers.gemspec
      create  chefspec-myhelpers/Rakefile
      create  chefspec-myhelpers/README.md
      create  chefspec-myhelpers/bin/console
      create  chefspec-myhelpers/bin/setup
```

Updating the Gem Specification

~/chefspec-myhelpers/chefspec-myhelpers.gemspec

```
Gem::Specification.new do |spec|
  spec.name          = "chefspec-myhelpers"
  spec.version       = Chefspec::Myhelpers::VERSION
  spec.authors       = ["Franklin Webber"]
  spec.email         = ["franklin.webber@gmail.com"]

  spec.summary       = %q{Provides common ChefSpec Helpers.}
  spec.description   = %q{ChefSpec Helpers that define an alia...}
  spec.homepage      = "https://github.com/burtlo/chefspec...com"
  spec.license       = "MIT"
```


CHEF

Moving the Shared Context to the Gem

~/chefspec-myhelpers/lib/chefspec/myhelpers.rb

```
RSpec.configure do |config|
  config.color = true
  config.alias_example_group_to :describe_recipe, :type => :recipe
end

shared_context 'converged recipe' do
 let(:chef_run) do
   runner = ChefSpec::SoloRunner.new(node_attributes)
   runner.converge(described_recipe)
 end

  let(:node_attributes) do
    {}
  end
```


CHEF

Changing into the Gem's Directory

```
> cd chefspec-myhelpers
```


CHEF

Creating the Gem

```
> chef gem build chefspec-myhelpers.gemspec
```

```
Successfully built RubyGem
 Name: chefspec-myhelpers
 Version: 0.1.0
 File: chefspec-myhelpers-0.1.0.gem
```

Installing the Gem

```
> chef gem install chefspec-myhelpers-0.1.0.gem
```

```
 install chefspec-myhelpers-0.1.0.gem
Successfully installed chefspec-myhelpers-0.1.0
1 gem installed
```


CHEF

Listing the installed Gem

```
> chef gem list chefspec
```

```
*** LOCAL GEMS ***

chefspec (4.7.0, 4.2.0)
chefspec-myhelpers (0.1.0)
```


CHEF

Updating the Cookbook's spec_helper

~/ark/spec/spec_helper.rb

```
require 'chefspec'
require 'chefspec/berkshelf'
require 'chefspec/myhelpers'                                    +

at_exit { ChefSpec::Coverage.report! }

RSpec.configure do |config|
  config.color = true
  config.alias_example_group_to :describe_recipe, :type => :recipe-
end

shared_context 'converged recipe', :type => :recipe do          -
```

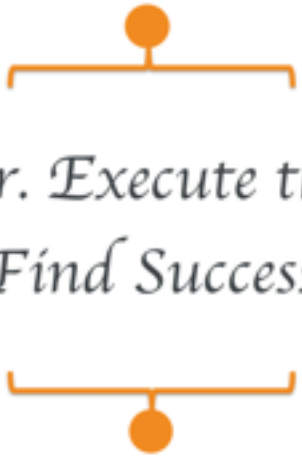

CHEF

Creating a Ruby Gem

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ❑ Exercise

CHEF

Exercise



Refactor. Execute the Tests.
Find Success.

Creating a Ruby Gem

- ✓ Concepts
- ✓ Demonstration
- ✓ Review
- ✓ Exercise


CHEF™