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

CHEF

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.

CHEF

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

CHEF™

Let's Start this Journey in the Home Directory

```
> cd ~
```

View the Tests in the Generated Cookbook

```
> tree apache
```

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

EXERCISE

Build a Reliable Cookbook

This time it will be different.

Objective:

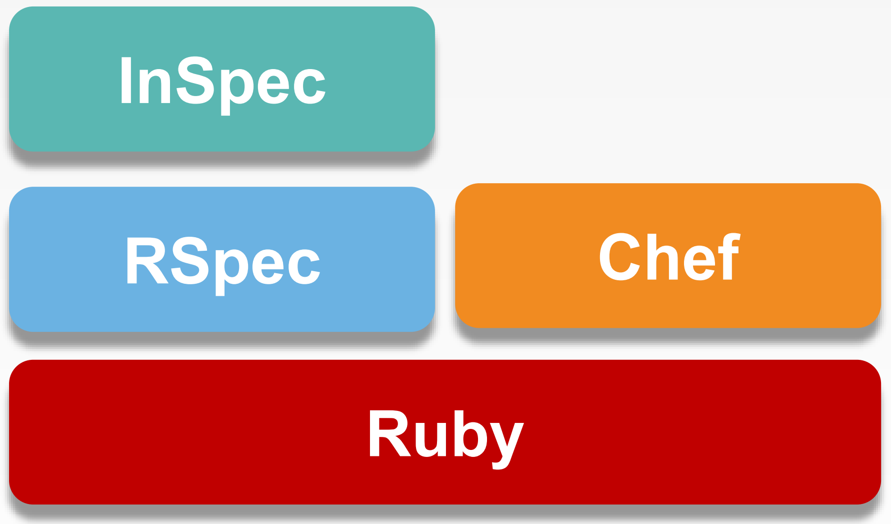
- ✓ Examine the cookbook
- ❑ Write tests that verifies the cookbook does what we want it to do
- ❑ Execute the tests and see failure
- ❑ Write the recipe to make the test pass
- ❑ Execute the tests and see success

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.



Auto-generated Spec File in Cookbook

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

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

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

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.

CHEF

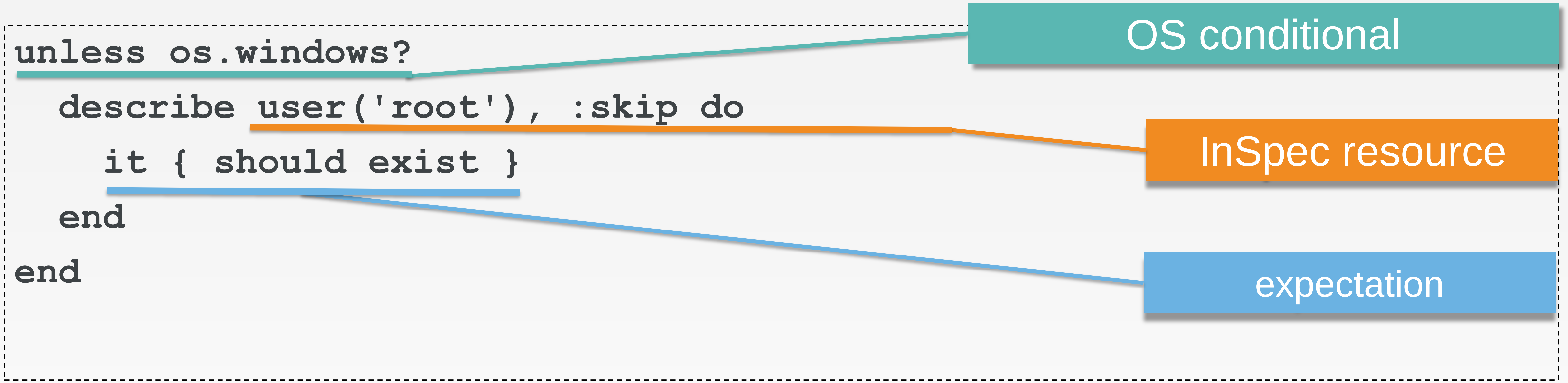
CONCEPT

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
```

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

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

CHEF

Update the Test for Port 80

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

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

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

CHEF

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
```

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

CHEF

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
```

Add Settings to the Kitchen Configuration

~/apache/.kitchen.yml

```
---
driver:
  name: docker

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: centos-6.9
```

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-69   Docker  ChefZero     InSpec    Ssh        <Not Created>
```

CONCEPT

Kitchen Create

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

Create one or more instances.

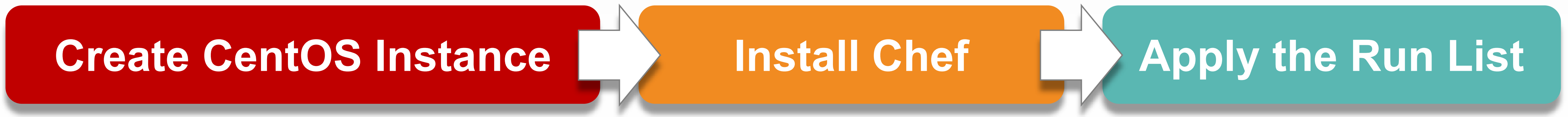
Create CentOS Instance

CONCEPT

Kitchen Converge

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

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

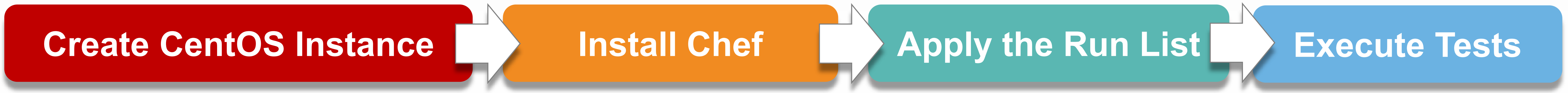


CONCEPT

Kitchen Verify

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

Create, converge, and verify one or more instances.



Create the Virtual Instance

```
> kitchen create
```

```
-----> Starting Kitchen (v1.11.1)
-----> Creating <default-centos-69>...
       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
```

```
Last login: Fri Mar 23 15:48:26 2018 from 172.17.42.1
[kitchen@bc530336220c ~]$
```

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-69>...
$$$$$$ 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-69> (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-69>...
-----> Verifying <default-centos-69>...
       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/...
```

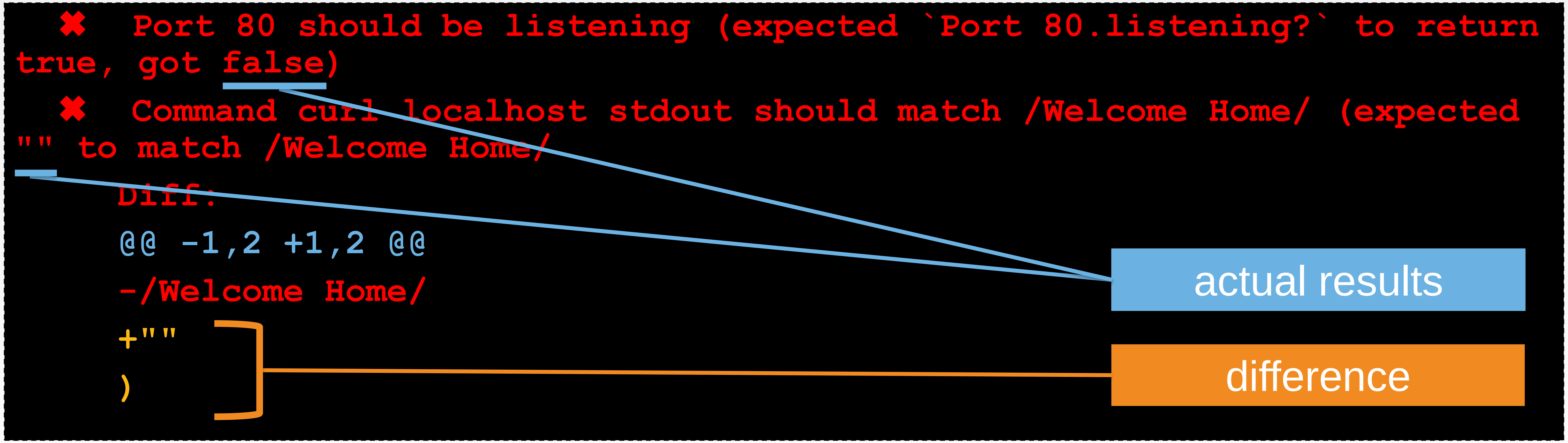
Understanding the Failure Message

```
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-69>.  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/
     +""
     )
```

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

CHEF™

Re-Verify the Virtual Instance

```
> kitchen verify
```

```
-----> Starting Kitchen (v1.11.1)
-----> Verifying <default-centos-69>...
       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
```

CHEF

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

CHEF™

Discussion

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

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

CHEF™

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™