Testing Cookbooks

Validating Our Recipes in Virtual Environments

CHEF™

Objectives

After completing this module, you should be able to

- ➢ Use Test Kitchen to verify your recipes converge on a virtual instance
- ➢ Define an InSpec test
- ➢ Write and execute tests

Can We Test Cookbooks?

As we start to define our infrastructure as code we also need to start thinking about testing it.

DISCUSSION

Mandating Testing

What steps would it take to test one of the cookbooks that we have created?

Steps to Verify Cookbooks

Create Virtual Machine

Install Chef Tools

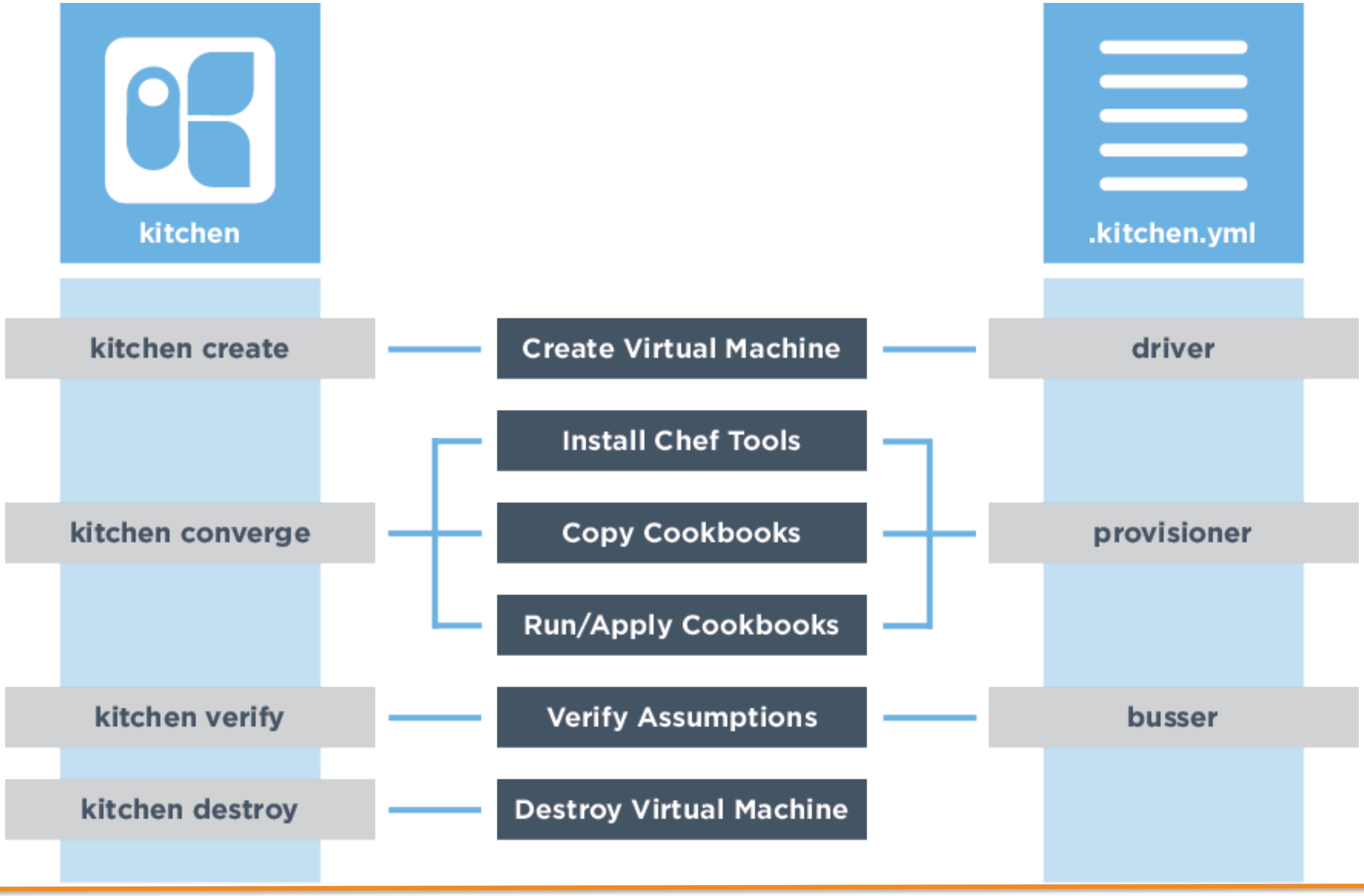
Copy Cookbooks

Run/Apply Cookbooks

Verify Assumptions

Destroy Virtual Machine

Test Kitchen Commands and Configuration



What Can 'kitchen' Do?

```
$ kitchen --help
```

```
Commands:
  kitchen console                              # Kitchen Console!
  kitchen converge [INSTANCE|REGEXP|all]       # Converge one or more instances
  kitchen create [INSTANCE|REGEXP|all]         # Create one or more instances
  kitchen destroy [INSTANCE|REGEXP|all]        # Destroy one or more instances
  ...
  kitchen help [COMMAND]                       # Describe available commands or one specif...
  kitchen init                                 # Adds some configuration to your cookbook...
  kitchen list [INSTANCE|REGEXP|all]           # Lists one or more instances
  kitchen setup [INSTANCE|REGEXP|all]          # Setup one or more instances
  kitchen test [INSTANCE|REGEXP|all]           # Test one or more instances
  kitchen verify [INSTANCE|REGEXP|all]         # Verify one or more instances
  kitchen version                              # Print Kitchen's version information
```

CHEF

What Can 'kitchen init' Do?

```
$ kitchen --help init
```

```
UsaGL:
  kitchen init
  -D, [--driver=one two three]                      # One or more Kitchen Driver gems ...
                                                    # Default: kitchen-vagrant
  -P, [--provisioner=PROVISIONER]                   # The default Kitchen Provisioner to
use
                                                    # Default: chef_solo
      [--create-gemfile], [--no-create-gemfile]     # Whether or not to create a Gemfi ...

Description:
  Init will add Test Kitchen support to an existing project for convergence
  integration testing. A default .kitchen.yml file (which is intended to be
  customized) is created in the project's root directory and one or more gems will be
  added to the project's Gemfile.
```

Do We Have a kitchen.yml?

```
$ tree cookbooks/workstation -a
```

```
...
├── .kitchen.yml
├── metadata.rb
├── README.md
├── recipes
│   ├── default.rb
│   └── setup.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
└── test
```

What is Inside .kitchen.yml?

```
$ cat cookbooks/workstation/kitchen.yml
```

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

kitchen.yml

When chef generates a cookbook, a default kitchen.yml is created. It contains kitchen configuration for the driver, provisioner, platform, and suites.

http://kitchen.ci/docs/getting-started/creating-cookbook

Demo: The kitchen Driver

~/cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

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

Example Drivers:

- docker
- vagrant

Demo: The kitchen Provisioner

~/cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

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**.

Chef Zero is a simple, easy-install, in-memory Chef server that can be useful for Chef Client testing and chef-solo-like tasks that require a full Chef Server.

chef_solo is an open source version of the chef-client that allows using cookbooks with nodes without requiring access to a Chef server. chef-solo runs locally and requires that a cookbook (and any of its dependencies) be on the same physical disk as the node.

Demo: The kitchen Verifier

~/cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

This tells Test Kitchen how to verify the converged instances.

The default approach is to use InSpec.

Demo: The kitchen Platforms

~/cookbooks/workstation/.kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-7.2
```

This is a list of operation systems on which we want to run our code.

Demo: The kitchen Suites

`~/cookbooks/workstation/.kitchen.yml`

```
...

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    verifier:
      inspec_tests:
        - test/recipes
    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".

Demo: The kitchen Suites

~/cookbooks/workstation/.kitchen.yml

```
...

suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    verifier:
      inspec_tests:
        - test/recipes
    attributes:
```

The suite named "default" defines a run_list.

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

Kitchen Test Matrix

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.

Example: Kitchen Test Matrix

```
$ kitchen list

Instance             Driver   Provisioner  Verifier  Transport Last Action
default-ubuntu-1204  Vagrant  ChefZero     Busser    Ssh       <Not Created>
default-centos-65    Vagrant  ChefZero     Busser    Ssh       <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```

Example: Kitchen Test Matrix

```
$ kitchen list


Instance              Driver   Provisioner  Verifier  Transport Last Action
default-ubuntu-1204   Vagrant  ChefZero     Busser    Ssh       <Not Created>
default-centos-65     Vagrant  ChefZero     Busser    Ssh       <Not Created>
```

```
suites:
  - name: default
    run_list:
      - recipe[workstation::default]
    attributes:
```

```
platforms:
  - name: ubuntu-12.04
  - name: centos-6.5
```

Test Workstation cookbook

What are we running in production? Maybe I could test the cookbook against a virtual machine.

Objective:

- ❑ Configure the "**workstation**" cookbook kitchen.yml to use the Docker driver and **ubuntu-16.04** platform
- ❑ Use kitchen converge to apply the recipe on a virtual machine

GL: Move into the Cookbook's Directory

```
$ cd cookbooks/workstation
```

GL: Edit the Kitchen Configuration File

`~/cookbooks/workstation/kitchen.yml`

```
---
driver:
  name: docker
  use_sudo: false

provisioner:
  name: chef_zero
  product_name: chef
  product_version: 15

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04

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

https://github.com/portertech/kitchen-docker

In order to make the driver ‘docker’ work

```
# Install DOCKER
curl https://get.docker.com/ | bash

# Install docker gem
chef gem install kitchen-docker

# Provide docker sudo access
sudo usermod -a -G docker $USER
```

In our AMI, these steps are already done

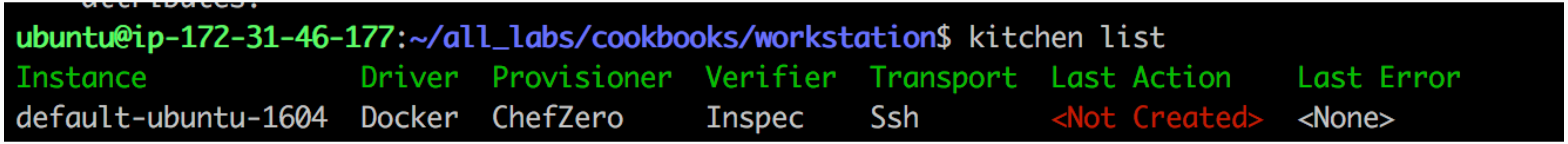
Introduction to Docker

Objective:

- ❑ Introduction to Docker

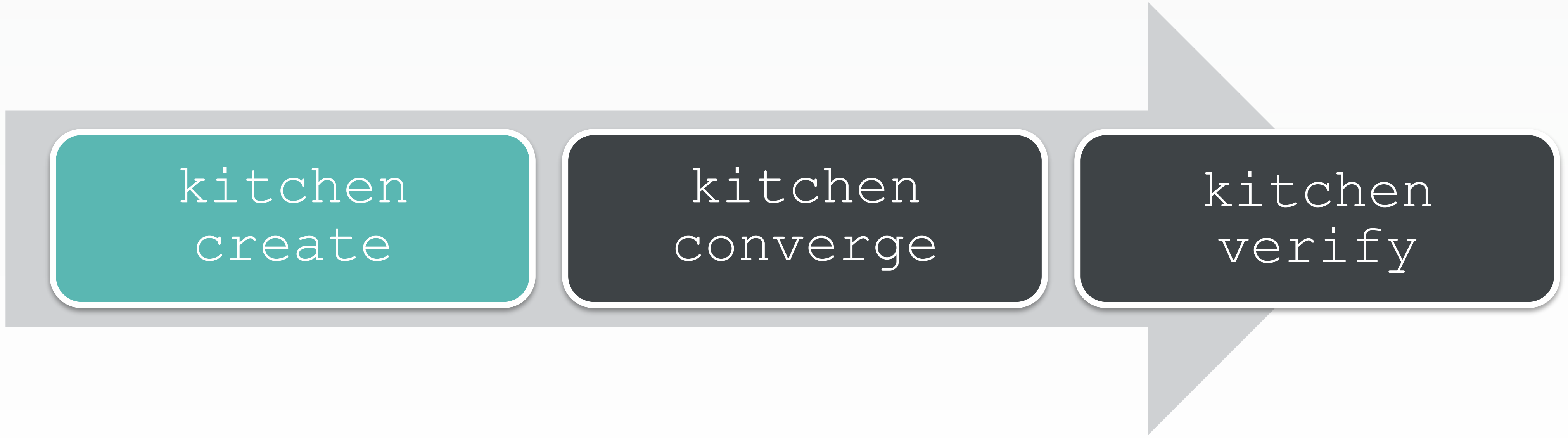
GL: Look at the Test Matrix

```
$ kitchen list
```



```
ubuntu@ip-172-31-46-177:~/all_labs/cookbooks/workstation$ kitchen list
Instance             Driver  Provisioner  Verifier  Transport  Last Action    Last Error
default-ubuntu-1604  Docker  ChefZero     Inspec    Ssh        <Not Created>  <None>
```

Kitchen Create



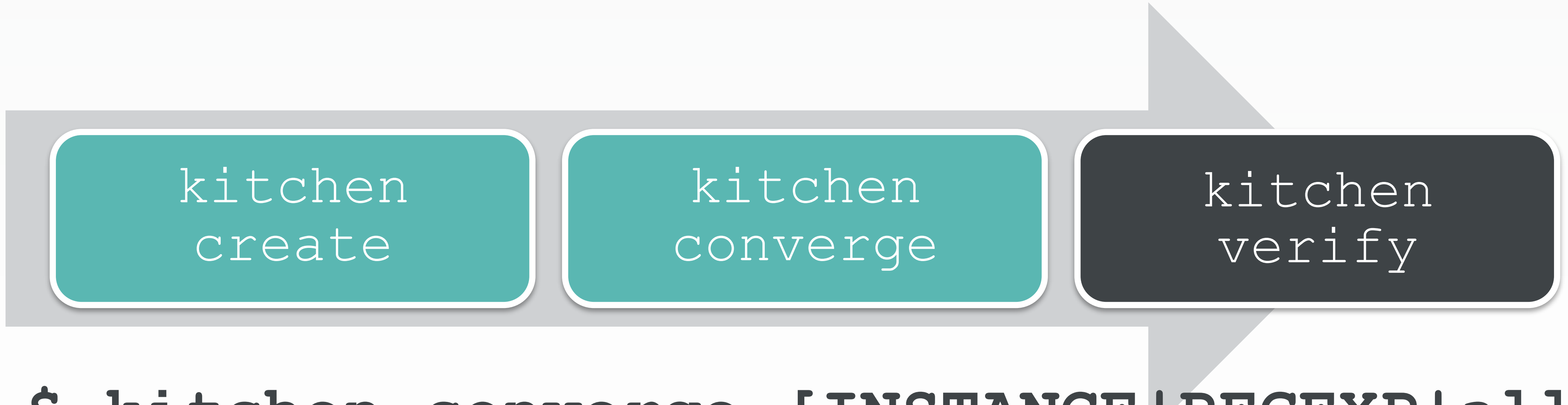
```
$ kitchen create [INSTANCE|REGEXP|all]
```

Create one or more instances.

Check Docker

```
$ sudo docker images
$ sudo docker ps
```

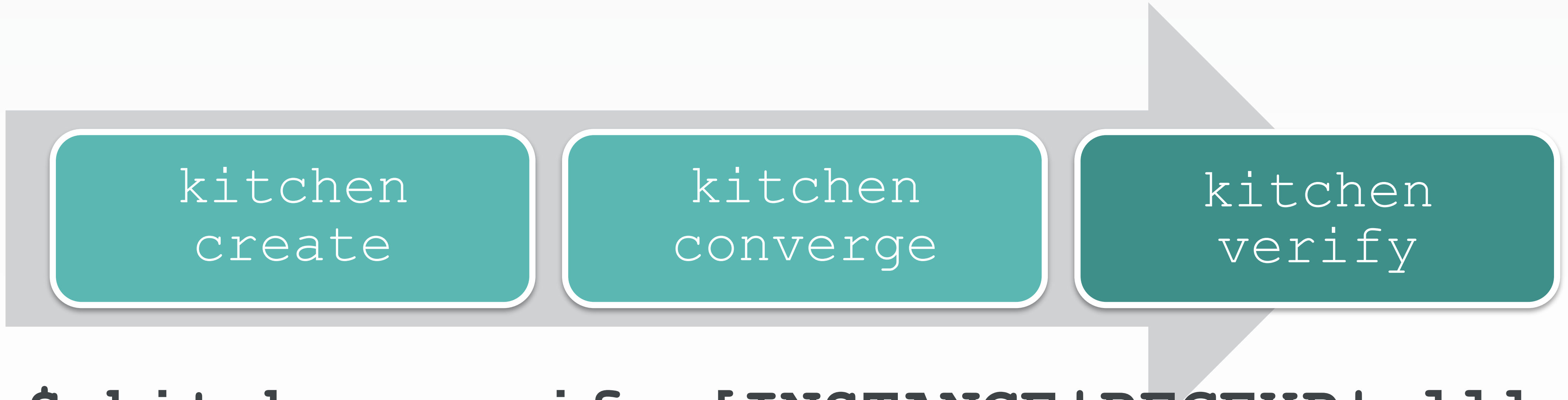
Kitchen Converge



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

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

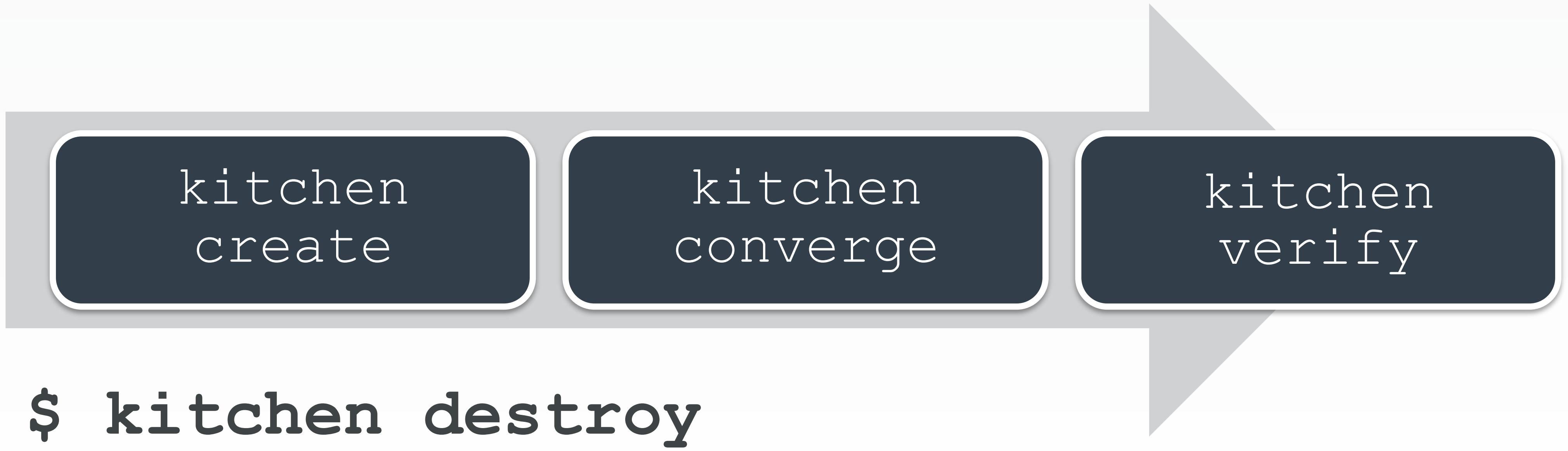
Kitchen Verify



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

Run inspec

Kitchen Destroy



```
$ kitchen destroy
```

Destroy the created kitchen

Recap

So far, cookbook **workstation** has been tested for ubuntu-16

Next, we will

- Test cookbook **apache** for one platform (ubuntu16)
- Test cookbook apache for multiple platforms (ubuntu16 and centos6)
- Test cookbook **apache** with cookbook **workstation** as a dependency

Test Apache cookbook – single platform

We want to validate that our run-list installs correctly.

- ❑ Within the "**apache**" cookbook use kitchen converge for the default suite on the **ubuntu-16.04** platform

Lab: Configuring Test Kitchen for Apache

~/cookbooks/apache/kitchen.yml

```yaml
---
driver:
  name: docker
  use_sudo: false

provisioner:
  name: chef_zero
  product_name: chef
  product_version: 15

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04

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

https://github.com/portertech/kitchen-docker

Lab: Configuring Test Kitchen for Apache

Did you see any error in running **kitchen converge** ?

```
~/cookbooks/apache/recipes/default.rb
```

Remove include_recipe ‘workstation::setup’

```
~/cookbooks/apache/metadata.rb
```

Remove depends ‘workstation’

Run kitchen converge again !!

CHEF

Test Apache cookbook (multi-platform)

We want to validate that our run-list installs correctly.

- ❑ Within the "**apache**" cookbook use kitchen converge for the default suite on the **ubuntu-16.04** and **centos-6.7** platform

Lab: Configuring Test Kitchen for Apache

`~/cookbooks/apache/kitchen.yml`

```yaml
---
driver:
  name: docker
  use_sudo: false

provisioner:
  name: chef_zero
  product_name: chef
  product_version: 15

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-6.7

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

https://github.com/portertech/kitchen-docker

Lab: Configuring Test Kitchen for Apache

Did you see any error in running **kitchen converge** ?

This is because ubuntu and centos supports different web servers.
Let's change the server.rb file

Lab: Configuring Test Kitchen for Apache

`~/cookbooks/apache/recipe/server.rb`

```
case node[:platform]
when "ubuntu", "debian"
  package "apache2" do
    action :install
  end
when "centos", "redhat", "amazon"
  package "httpd" do
    action :install
  end
end
```

Check the platform using NODE object and install appropriate package

```
case node[:platform]
when "ubuntu", "debian"
  service "apache2" do
    action [:start, :enable]
  end
when "centos", "redhat", "amazon"
  service "httpd" do
    action [:start, :enable]
  end
end
```

Check the platform using NODE object and start the appropriate service

```
file '/var/www/html/index.html' do
  content '<h1>Hello, world!</h1>'
end
```

Create index.html file

Lab: Configuring Test Kitchen for Apache

```
** Note: for centos7+ version, make more changes
```

```
---
driver:
  name: docker
  use_sudo: false
  privileged: true

provisioner:
  name: chef_zero

verifier:
  name: inspec

platforms:
  - name: centos-7.3
    driver:
      platform: rhel
      run_command: /usr/lib/systemd/systemd
```

Test Apache cookbook – with dependency on Workstation cookbook (single or multiple platforms)

We want to validate that our run-list installs correctly.

❑ Within the "**apache**" cookbook use kitchen converge for the default suite on the **ubuntu-16.04** and **centos-6.7** platform and **include cookbook workstation**

CHEF

Lab: Configuring Test Kitchen for Apache

~/cookbooks/apache/kitchen.yml

```
---
driver:
  name: docker
  use_sudo: false

provisioner:
  name: chef_zero
  product_name: chef
  product_version: 15

verifier:
  name: inspec

platforms:
  - name: ubuntu-16.04
  - name: centos-6.7

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

https://github.com/portertech/kitchen-docker

Lab: Configuring Test Kitchen for Apache

~/cookbooks/apache/recipe/default.rb

```
include_recipe 'apache::server'
include_recipe 'workstation::setup'
```

include workstation recipe

~/cookbooks/apache/metadata.rb

```
depends 'workstation'
```

Add workstation cookbook dependency

~/cookbooks/apache/Policyfile.rb

```
name 'apache'
default_source :supermarket
run_list 'apache::default'
cookbook 'apache', path: '.'
cookbook 'workstation', path: '../workstation'
```

Add workstation cookbook dependency

CHEF

DISCUSSION

Test Kitchen

What is being tested when kitchen converges a recipe without error?

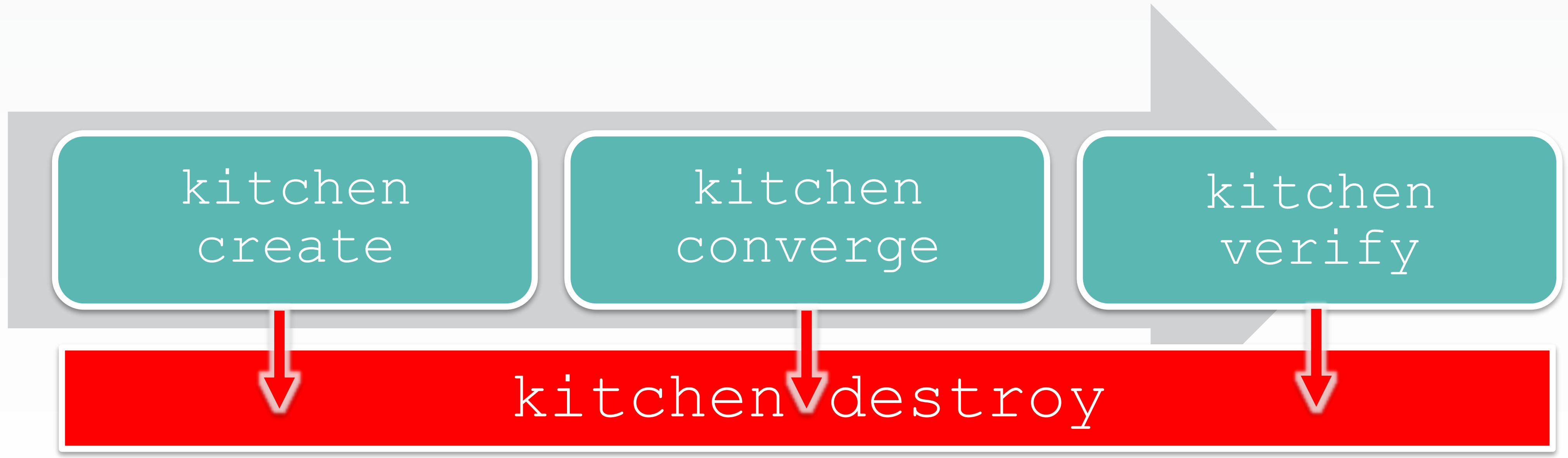
What is NOT being tested when kitchen converges the recipe without error?

DISCUSSION

Test Kitchen

What is left to validate to ensure that the cookbook successfully applied the policy defined in the recipe?

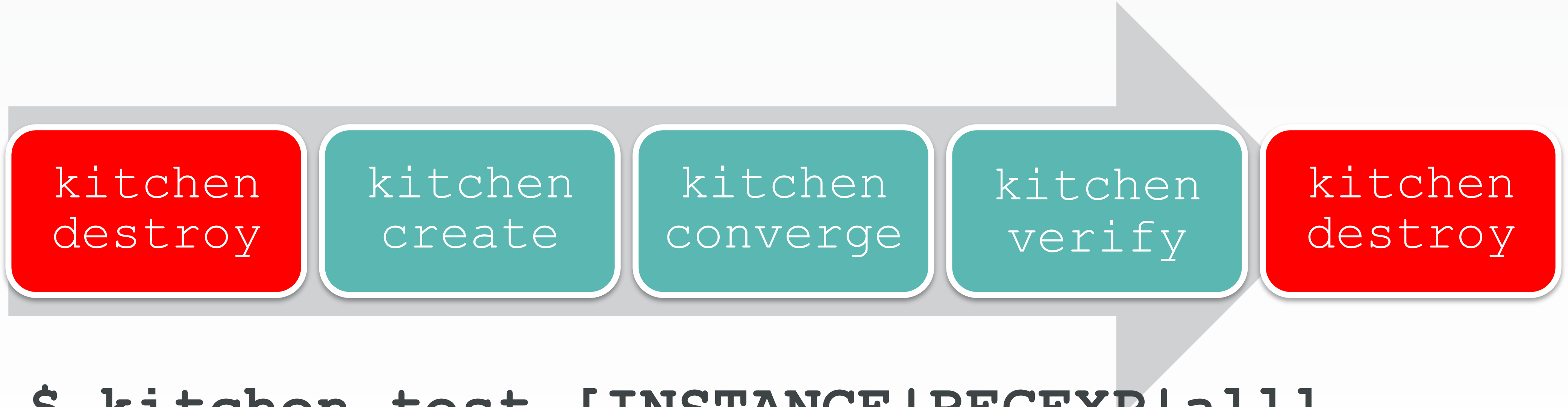
Kitchen Destroy



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

Destroys one or more instances.

Kitchen Test



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

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

InSpec

InSpec tests your servers' actual state by executing command locally, via SSH, via WinRM, via Docker API and so on.

https://www.inspec.io/docs/reference/resources/

http://inspec.io/

Where do Tests Live?

```
apache/test/recipes/default_test.rb
```

Test Kitchen will look for tests to run under this directory.

This is configurable in the kitchen configuration file (.kitchen.yml) in the suites section.

GL: Example Tests

~/cookbooks/apache/test/recipes/default_test.rb

```
unless os.windows?
  describe user('root') do
    it { should exist }
    skip 'This is an example...test.'
  end
end

describe port(80) do
  it { should_not be_listening }
  skip 'This is an example... test.'
end
```

When not on windows a user named `root` should exist.

The port 80 should not be listening for incoming connections.

GL: Describing the Resources

`~/cookbooks/apache/test/recipes/default_test.rb`

```
unless os.windows?
  describe user('root') do
    it { should exist }
    skip 'This is an example...test.'
  end
end

describe port(80) do
  it { should_not be_listening }
  skip 'This is an example... test.'
end
```

A user named 'root'

The port 80

https://relishapp.com/rspec/rspec-core/v/3-3/docs

GL: Describing the State of the Resources

```
~/cookbooks/apache/test/recipes/default_test.rb
```

```
unless os.windows?
  describe user('root') do
    it { should exist }
    skip 'This is an example...test.'
  end
end

describe port(80) do
  it { should_not be_listening }
  skip 'This is an example... test.'
end
```

The user named 'root' should exist.

The port 80 should not be listening.

https://relishapp.com/rspec/rspec-core/v/3-3/docs

GL: The `skip` Reminds Us to Remove These Tests

~/cookbooks/apache/test/recipes/default_test.rb

```
unless os.windows?
  describe user('root') do
    it { should exist }
    skip 'This is an example...test.'
  end
end

describe port(80) do
  it { should_not be_listening }
  skip 'This is an example... test.'
end
```

skip will show a message in the test results.

These skips will remind you that the following expectations are only examples.

GL: Adding a New Test

~/cookbooks/apache/test/recipes/default_test.rb

```
unless os.windows?
  describe user('root') do
    it { should exist }
    skip 'This is an example...test.'
  end
end

describe port(80) do
  it { should_not be_listening }
  skip 'This is an example... test.'
end

describe package('tree') do
  it { should be_installed }
end
```

GL: Return Home and Move into the Cookbook

```
$ cd ~/cookbooks/apache
```

LOCAL

CHEF

GL: Running the Specification

```
$ kitchen verify
```

```
-----> Starting Kitchen (v1.11.1) ...
-----> Verifying <default-centos-67>...
       Use `/home/chef/cookbooks/workstation/test/recipes/default` for testing
Target:  ssh://kitchen@localhost:32768


  ✓  User root should exist
  ✓  Port 80 should not be listening
  ✓  System Package tree should be installed

Summary: 3 successful, 0 failures
       Finished verifying <default-centos-67> (0m1.39s).
-----> Kitchen is finished. (0m2.76s)
```

GL: Commit Your Work

```
$ cd ~/cookbooks/workstation
$ git add .
$ git status
$ git commit -m "Add first test for default test suite"
```

DISCUSSION

More Tests

What are other resources within the recipe that we could test?

CONCEPT

Testing a File

InSpec can help us assert different characteristics about files on the file system. Like if it is a file, directory, socket or symlink.

InSpec can also help us assert the file's mode, owner or group or if the file is readable, writeable, or executable. Inspec is even able to verify the data contained within the file.

http://inspec.io/docs/reference/resources/file

CHEF

Write more test cases

Within the "apache" cookbook, write more test specs

- ❑ Test if the installed package “tree” has version > 1.0 → This should pass
- ❑ Test if file “/etc/motd” is present → This should pass
- ❑ Test if file “/etc/motd” is a file or a directory → This should pass
- ❑ Test if file “/etc/motd” content has the word “Property” → This should pass
- ❑ Test if file “/etc/motd” content exactly matches “xxx” → This should fail
- ❑ Test if port 80 is listening → This should pass
- ❑ Test curl command

File Test Cases

http://inspec.io/docs/reference/resources/file

```
# Test if a file is file
describe file('/etc/motd') do
  it { should be_file }
end

# Test the content of file
describe file ('/etc/motd') do
  its('content') { should match /this is me/ }
end

# Test the owner of the file
describe file('/etc/motd') do
  it { should be_owned_by 'root' }
end

# Test the content of the file
describe file('/etc/motd') do
 its('content') { should match(%r{.*Proper.*}) }
end
```

Package Test Cases

https://www.inspec.io/docs/reference/resources/package/

```
describe package('tree') do
  it { should be_installed }
  its('version') { should cmp >= '1.0' }
end

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

# Test curl command
describe command('curl localhost') do
  its('stdout') { should match('Hello, world') }
end
```

Lab: Running the Specification

```
$ kitchen verify
```

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

Target:  ssh://kitchen@localhost:32768

  ✓  User root should exist
  ✓  Port 80 should not be listening
  ✓  System Package tree should be installed
  ✓  File /etc/motd should be owned by "root"; File /etc/motd con...

Summary: 5 successful, 0 failures, 0 skipped
       Finished verifying <default-centos-67> (0m1.43s).
```

DISCUSSION

Testing

What are some things we could test to validate our web server has deployed correctly?

What manual tests do we use now to validate a working web server?

Lab
120 minutes

https://github.com/shekhar2010us/chef-essentials-repo-15/blob/master/labs/chapter%205.md

DISCUSSION

Discussion

Why do you have to run kitchen within the directory of the cookbook?

Where would you define additional platforms?

Why would you define a new test suite?

What are the limitations of using Test Kitchen to validate recipes?

CHEF™

DISCUSSION

Q&A

What questions can we help you answer?

- Test Kitchen
- kitchen commands
- kitchen configuration
- InSpec

CHEF™