Testing Cookbooks – Prelude

Setting up our test environment in advance of this lesson

Before We Start this Lesson

Lets run a few commands that could save us about 15-20 minutes later

Don’t worry about what we’re doing – we’ll explain as we go through the lesson

Just follow along

GL: Replace the Kitchen Config

```
> cp ~\kitchen-template.yml ~\cookbooks\workstation\kitchen.yml
```

For anyone who has used ChefDK in the past, in the newer Chef Workstation we are using, the leading dot on the kitchen.yml file is no longer needed.

GL: Add Your Name, Company & Cookbook

`~\cookbooks\workstation\kitchen.yml`

```
... CONTENT OMITTED FOR CLARITY...

  instance_type: m3.large
  tags:
    # Replace YOURNAME and YOURCOMPANY here
    Name: "Chef Training Node for YOURNAME,YOURCOMPANY"
    created-by: "test-kitchen"
    user: <%= ENV['USER'] %>
```

Line 14 – Add your name and company

GL: Add chef-client 15 Requirements

~\cookbooks\workstation\kitchen.yml

```
... CONTENT OMITTED FOR CLARITY...


provisioner:
  name: chef_zero
  cookbook_path: C:\Users\Administrator\cookbooks
  client_rb:
   chef_license: accept
   product_name: chef
   product_version: 15
```

Add these lines below line 20.

GL: Add Your Name, Company & Cookbook

~\cookbooks\workstation\kitchen.yml

```
... CONTENT OMITTED FOR CLARITY...


suites:
  - name: default
    run_list:
      # Replace with the name of the COOKBOOK
      - recipe[COOKBOOK::default]
    attributes:
```

Replace ‘COOKBOOK’ with ‘workstation’ in the suites section.

GL: Create the TK Instance

```
> cd ~\cookbooks\workstation
> kitchen create
```

```
-----> Creating <default-windows-2012r2>...
       Detected platform: windows version 2012rtm on x86_64. Instance Type:
m3.large. Default username: administrator (default).
...
Waited 160/600s for instance <i-03e36657d5fd9d8b0> to become ready.
       Waited 165/600s for instance <i-03e36657d5fd9d8b0> to become ready.
       Waited 170/600s for instance <i-03e36657d5fd9d8b0> to become ready.
       Waited 175/600s for instance <i-03e36657d5fd9d8b0> to become ready.
       EC2 instance <i-03e36657d5fd9d8b0> ready (hostname:
ec2-100-25-15-150.compute-1.amazonaws.com).
       [WinRM] Established
       Finished creating <default-windows-2012r2> (3m18.83s).
```

Testing Cookbooks

Validating Our Recipes in Virtual Environments

Objectives

After completing this module, you should be able to:

- Use Test Kitchen to verify your recipes converge on a virtual instance
- Refer to the InSpec documentation
- Write and execute tests

We Should Test Cookbooks

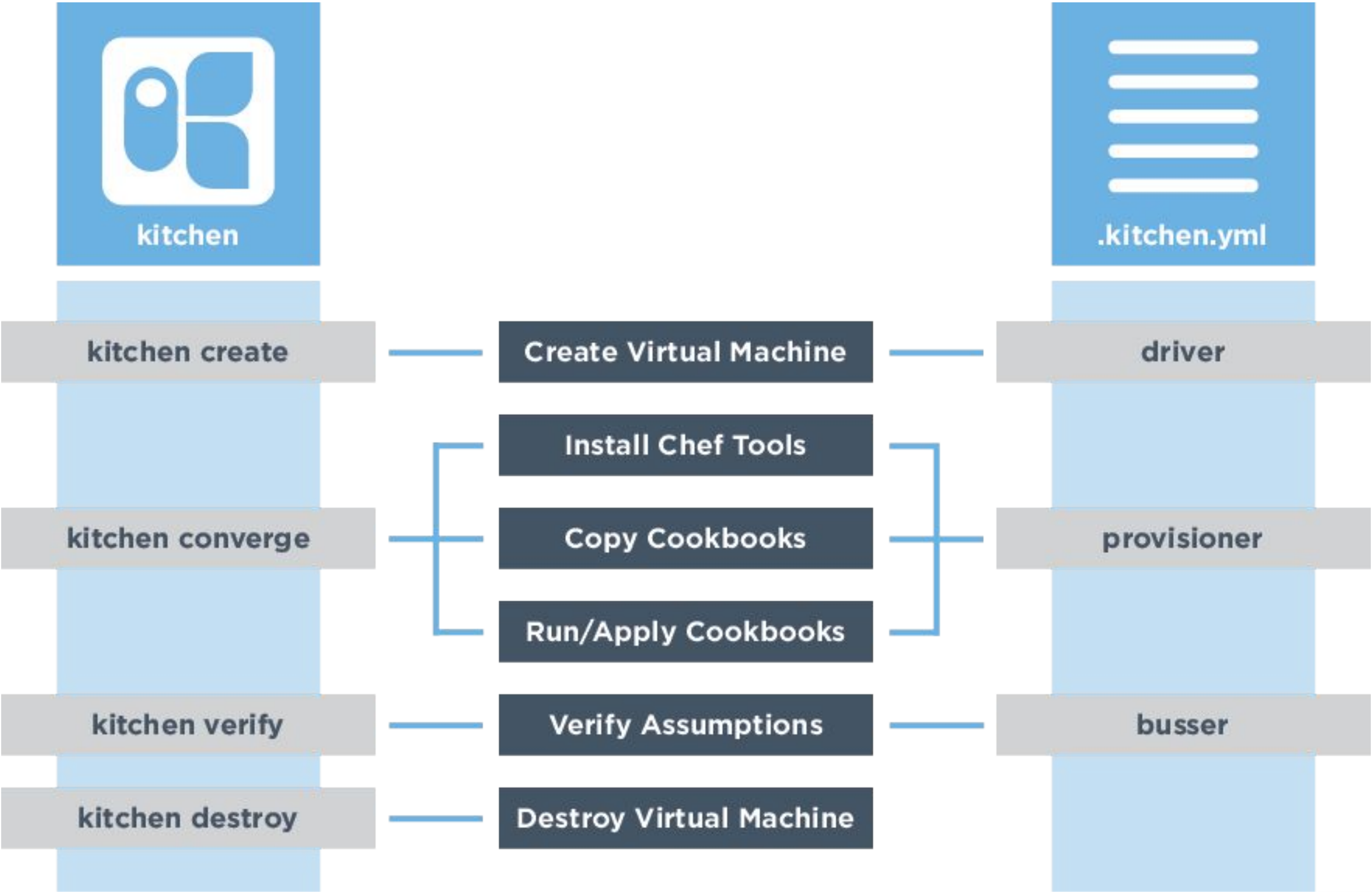
As we start to define our infrastructure as code we also need to start thinking about testing it.

We have an automated way to ensure code accomplishes the intended goal and help the team understand its intent. It's called Test Kitchen.

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



kitchen

.kitchen.yml

kitchen create — Create Virtual Machine — driver

kitchen converge — Install Chef Tools, Copy Cookbooks, Run/Apply Cookbooks — provisioner

kitchen verify — Verify Assumptions — busser

kitchen destroy — Destroy Virtual Machine

Test Configuration

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

Objective:

- □ Configure the "workstation" cookbook to test against a Windows 2012R2 platform
- □ Apply the "workstation" cookbook's default recipe to that virtual machine

kitchen.yml

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

Reminder: The newer Chef Workstation software no longer uses a leading dot in the kitchen.yml file name.

https://kitchen.ci/docs/getting-started/kitchen-yml/

Demo: The kitchen Driver

`~\cookbooks\workstation\kitchen.yml`

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

# Uncomment the following verifier...
# default verifier)
# verifier:
#   name: inspec

# KITCHEN CONFIGURATION CONTINUES...
```

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

Today we're using an EC2 driver that will spin up an instance in AWS that we can run Test Kitchen against.

Demo: The kitchen Provisioner

~\cookbooks\workstation\kitchen.yml

```
---
driver:
  name: vagrant

provisioner:
  name: chef_zero

# Uncomment the following verifier...
# default verifier)
# verifier:
#   name: inspec

# KITCHEN CONFIGURATION CONTINUES...
```

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.

Demo: The kitchen Platforms

~\cookbooks\workstation\kitchen.yml

```
#  TOP PART OF KITCHEN CONFIGURATION

platforms:
  - name: ubuntu-14.04
  - name: centos-7.1

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

This is a list of Operating Systems on which we want to run our code.

Demo: The kitchen Suites

~\cookbooks\workstation\kitchen.yml

```
#  TOP PART OF KITCHEN CONFIGURATION

platforms:
  - name: ubuntu-14.04
  - name: centos-7.1

suites:
  - name: default
    run_list:
      - recipe[workstation::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".

Demo: The kitchen Suites

`~\cookbooks\workstation\kitchen.yml`

```
#  TOP PART OF KITCHEN CONFIGURATION

platforms:
  - name: ubuntu-14.04
  - name: centos-7.1

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

The suite named "default" defines a run_list.

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

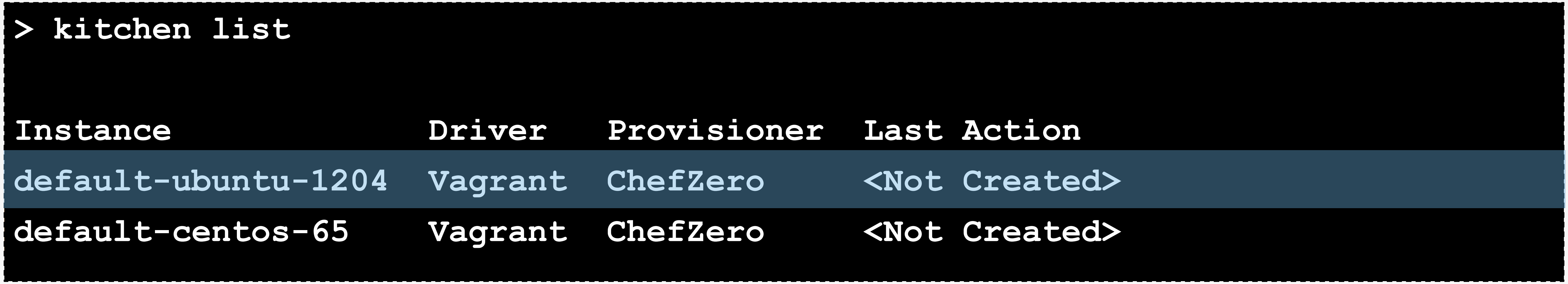
Kitchen Test Matrix

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: View the Kitchen Test Matrix



```
> kitchen list


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

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

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

Example: View the Kitchen Test Matrix

```
> kitchen list

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

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

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

PROBLEM

Virtualization

Vagrant is great for local development but when building cookbooks on a virtual machine in the cloud we will not need to use a local virtualization tool.

Instead, we will ask Amazon EC2 to provision nodes for us to test against.

REVIEW ONLY: You Added Your Name and Company

~\cookbooks\workstation\kitchen.yml

```
---
driver:
  name: ec2
  # ... OTHER DRIVER DETAILS ...
  instance_type: m3.large
  tags:
    # Replace YOURNAME and YOURCOMPANY here
    Name: "Chef Training Node for YOURNAME, YOURCOMPANY"
    created-by: "test-kitchen"
    user: <%= ENV['USER'] %>
# ... REMAINDER OF THE CONFIGURATION FILE ...
```

Review: You added chef-client 15 Requirements

~\cookbooks\workstation\kitchen.yml

```
... CONTENT OMITTED FOR CLARITY...


provisioner:
 name: chef_zero
 cookbook_path: C:\Users\Administrator\cookbooks
 client_rb:
  chef_license: accept
  product_name: chef
  product_version: 15
```

REVIEW: You Updated the kitchen.yml to Test the Workstation Cookbook

```
~\cookbooks\workstation\kitchen.yml
```

```
# ... TOP PART OF THE CONFIGURATION FILE ...


suites:
  - name: default
    run_list:
      # Replace with the name of the COOKBOOK
      - recipe[workstation::default]
    attributes:
```

GL: Ensure You Are in ~\cookbooks\workstation

```
> cd ~\cookbooks\workstation
```

GL: Look at the Test Matrix

```
> kitchen list
```

```
Instance                 Driver  Provisioner  Verifier  Transport  Last Action  Last Error
default-windows-2012r2   Ec2     ChefZero     Inspec    Winrm      Created      <None>
```

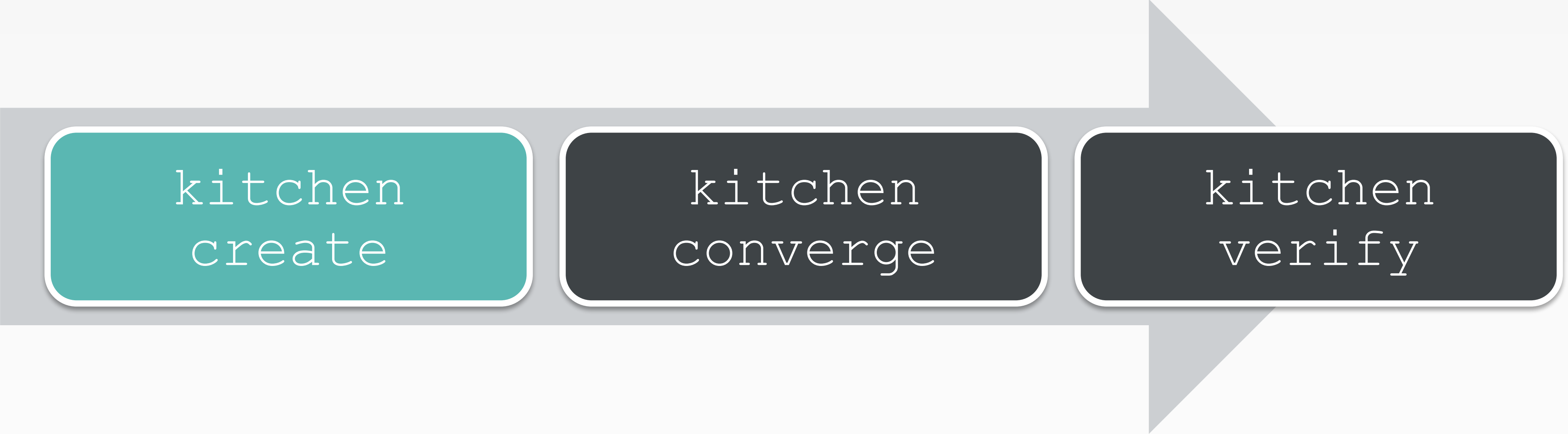
Test Configuration

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

Objective:

- ✔ Configure the "workstation" cookbook to test against a Windows 2012R2 platform
- ❑ Apply the "workstation" cookbook's default recipe to that virtual machine

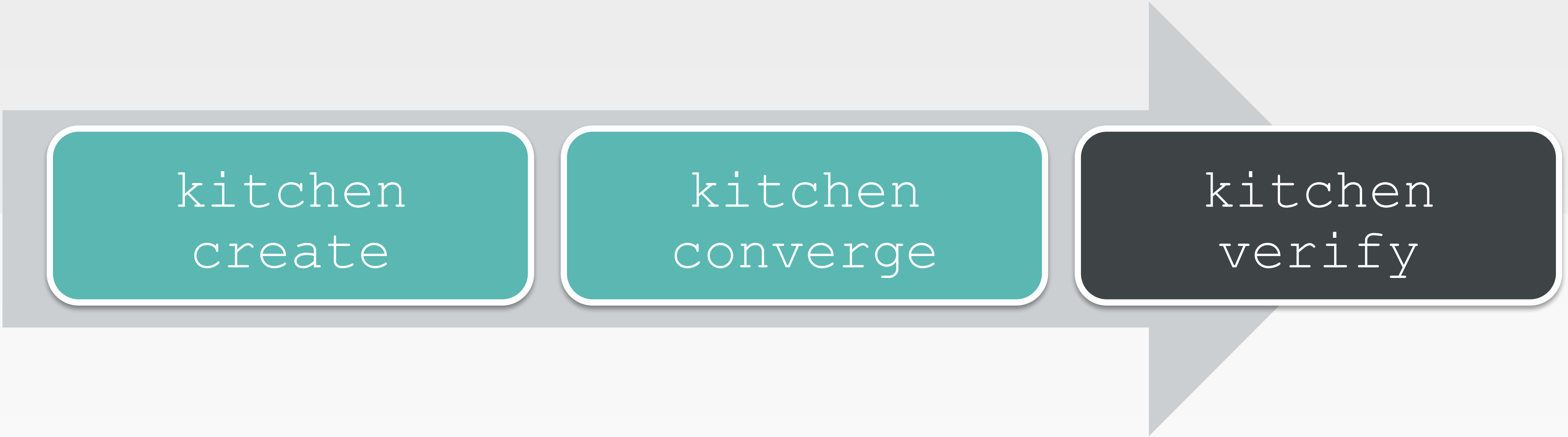
Kitchen Create



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

Create one or more instances.

Kitchen Converge



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

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

GL: Converge the Cookbook

```
> kitchen converge
```

```
-----> Starting Kitchen (v3.0.0)
-----> Converging <default-windows-2012r2>...
       Preparing files for transfer
$$$$$$ You must set your run_list in your Policyfile instead of kitchen config. The run_list
in your config will be ignored.
$$$$$$ Ignored run_list: ["recipe[workstation::default]"]
       Policy lock file doesn't exist, running `chef install` for Policyfile
C:/Users/Administrator/cookbooks/workstation/Policyfile.rb...
       Building policy workstation
       Expanded run list: recipe[workstation::default]
       Caching Cookbooks...
       Installing workstation >= 0.0.0 from path
 Building policy workstation
  Expanded run list: recipe[workstation::default]
       Caching Cookbooks...
```

We will discuss Policyfiles later in this course but for now,`kitchen create` created the Policyfile for us.

Test Configuration

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

Objective:

- ✔ Configure the "workstation" cookbook to test against a Windows 2012R2 platform
- ✔ Apply the "workstation" cookbook's default recipe to that virtual machine

Test Kitchen

What does this test when kitchen converges a recipe?

And what does it NOT test when kitchen converges a recipe?

Test Kitchen

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

The First Test

Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?

Objective:

- ❑ Write a test that asserts that we have disabled UAC when the "workstation" cookbook's default recipe is applied
- ❑ Execute the test that we have written

InSpec

InSpec tests your servers' actual state by executing commands locally or via SSH, via WinRM, via Docker API and so on against a test instance.

https://docs.chef.io/inspec/resources/

Example: Is the Registry Key Set Correctly?

```
system_policies = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'

describe registry_key('System Policies', system_policies) do
  its('EnableLUA') { should eq 0 }
end
```

I expect the 'System Policies' found at registry key 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System' to have the property 'EnableLUA' with the value 0.

https://docs.chef.io/inspec/resources/registry_key/

Where do Tests Live?

```
workstation\test\integration\default\default_test.rb
```

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

Note: Tests can actually live wherever you want them to live. The kitchen.yml gives you the flexibility to point to various places as it just loads tests based on the parameters.

GL: Replace the Existing Test File Contents with:

```
~\cookbooks\workstation\test\integration\default\default_test.rb
```

```
system_policies = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'

describe registry_key('System Policies', system_policies) do
  its('EnableLUA') { should eq 0 }
end
```

The First Test

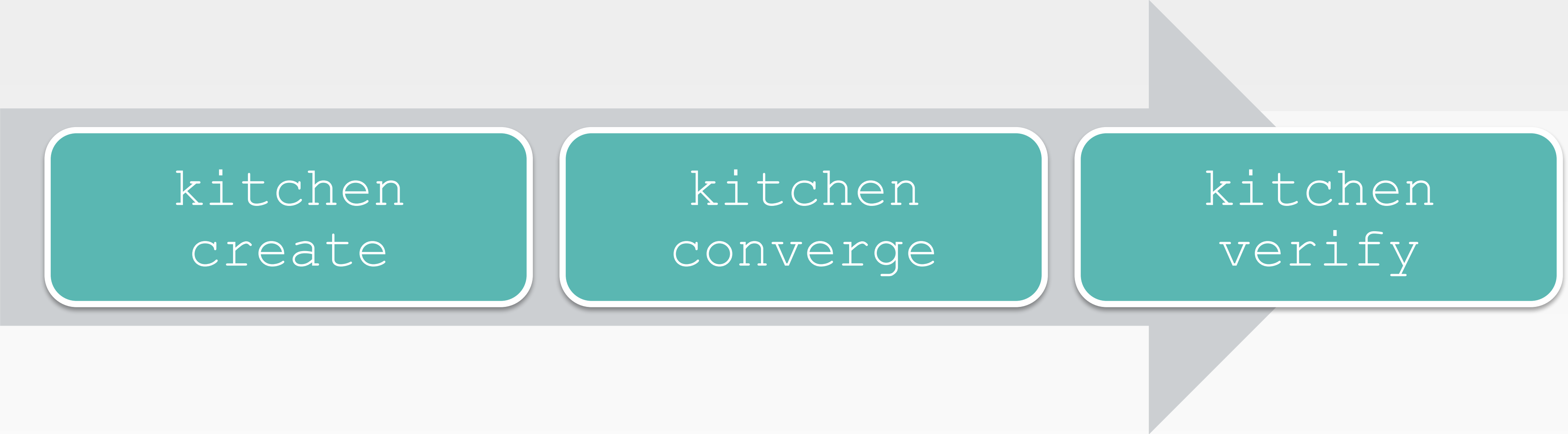
Converging seems to validate that the recipe runs successfully. But does it assert what actually is installed?

Objective:

✔ Write a test that asserts that we have disabled UAC when the "workstation" cookbook's default recipe is applied

❑ Execute the test that we have written

Kitchen Verify



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

Create, converge, and verify one or more
instances.
```

GL: Ensure you are in the Cookbook

```
> cd ~\cookbooks\workstation
```

GL: Running the Test

```
> kitchen verify
```

```
-----> Starting Kitchen (v3.0.0)
-----> Setting up <default-windows-2012r2>...
       Finished setting up <default-windows-2012r2> (0m0.00s).
-----> Verifying <default-windows-2012r2>...
[2020-06-28T17:54:25+00:00] WARN: DEPRECATION: InSpec Attributes are being renamed to InSpec Inputs to
avoid confusion with Chef Attributes. Use --input-file on the command line instead of --attrs.
       Loaded tests from
{:path=>"C:.Users.Administrator.cookbooks.workstation.test.integration.default"}
Profile: tests from {:path=>"C:/Users/Administrator/cookbooks/workstation/test/integration/default"}
(tests from {:path=>"C:.Users.Administrator.cookbooks.workstation.test.integration.default"})
...
  Registry Key System Policies
     [PASS]  EnableLUA should eq 0
Test Summary: 1 successful, 0 failures, 0 skipped
       Finished verifying <default-windows-2012r2> (0m4.98s).
-----> Kitchen is finished. (0m13.04s)
```

Lab: Additional Test

- ❑ Add tests that validate that the registry key 'ConsentPromptBehaviorAdmin' has been set to the value to 0. HINT: You can have multiple 'its' lines.

- ❑ Run kitchen verify to validate the test meets the expectations that you defined

Lab: Add This to the Existing Test File

```
~\cookbooks\workstation\test\integration\default\default_test.rb
```

```
system_policies = 'HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System'

describe registry_key('System Policies', system_policies) do
  its('EnableLUA') { should eq 0 }
  its('ConsentPromptBehaviorAdmin') { should eq 0 }
end
```

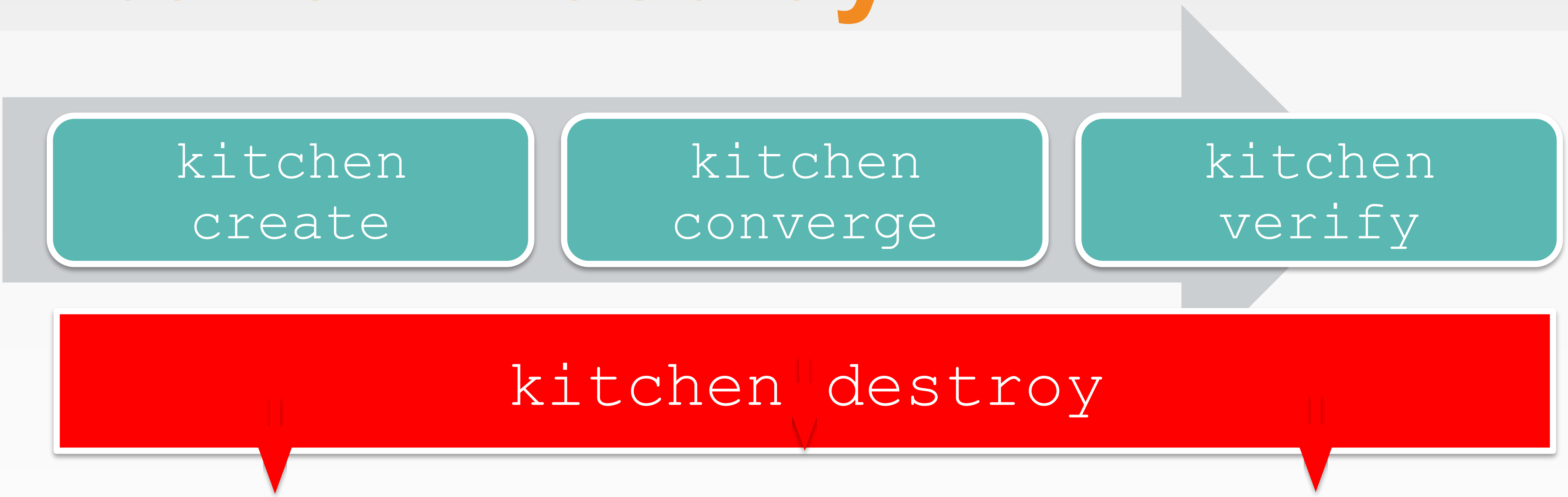
Lab: Running the Test

```
> kitchen verify
```

```
-----> Starting Kitchen (v3.0.0)
-----> Verifying <default-windows-2012r2>...
...
       Loaded tests from
{:path=>"C:.Users.Administrator.cookbooks.workstation.test.integration.default"}
...
Target:
winrm://Administrator@http://ec2-18-235-249-206.compute-1.amazonaws.com:5985/wsma
n:3389
  Registry Key System Policies


     [PASS]  EnableLUA should eq 0
     [PASS]  ConsentPromptBehaviorAdmin should eq 0
Test Summary: 2 successful, 0 failures, 0 skipped
```

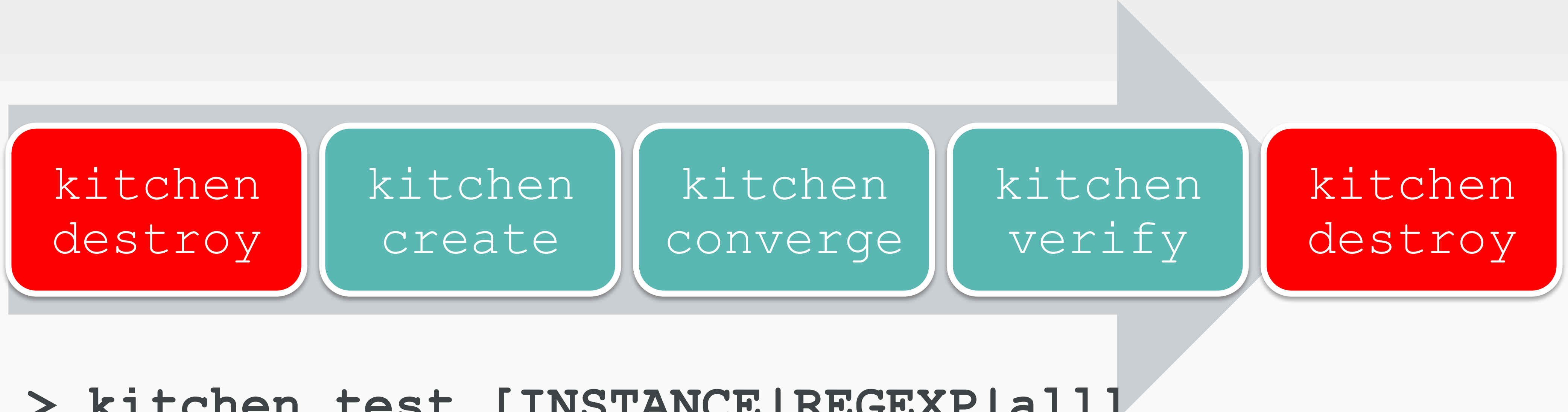
Kitchen Destroy



```
> 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.

Lab: kitchen destroy

We are done with Test Kitchen so please execute `kitchen destroy` to terminate the EC2 instance used in this module.

Lab: Run `kitchen destroy` to Clean Everything Up

```
> kitchen destroy
```

```
-----> Starting Kitchen (v3.0.0)
-----> Destroying <default-windows-2012r2>...
       EC2 instance <i-042b15bbb360a9a59> destroyed.
       Finished destroying <default-windows-2012r2> (0m1.36s).
-----> Kitchen is finished. (0m9.44s)
```

Important: Please be sure to run `kitchen destroy` to help us save instance costs.

Chef Intermediate

You can learn more about using Test Kitchen in the ***Chef Intermediate*** class.

There are also some Learn Chef Rally modules that cover Test Kitchen that you can take at any time.

https://learn.chef.io/#/

Review Questions

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

1. Where would you define additional platforms?

1. Why would you define a new test suite?

Q&A

What questions can we help you answer?

- Test Kitchen
- kitchen commands
- kitchen configuration
- InSpec

CHEF™