DevOps: Module 7, Lesson 6   
Test Kitchen: Integration Testing Using Test Kitchen Lab

## Overview

In this lab, you will test your Apache webserver in a CentOS 6.4 Docker container.

## Objectives

In this hands-on lab you will learn how to:

* Install the ChefDK and Docker from the command line
* Configure Test Kitchen
* Write several integration tests
* Run Test Kitchen

## Prerequisites

The following are required to complete this hands-on lab:

* A Microsoft Azure subscription
* A web browser
* A terminal application (if you are using a Mac or Linux workstation) or PuTTY (for a Windows workstation) to connect to the Linux VM
* Completion of the [Module 7 Lesson 4 Lab](https://github.com/MSFTImagine/computerscience/tree/master/Complimentary%20Course%20Content/Module7/Labs) 'Launching A Web Server With Chef’

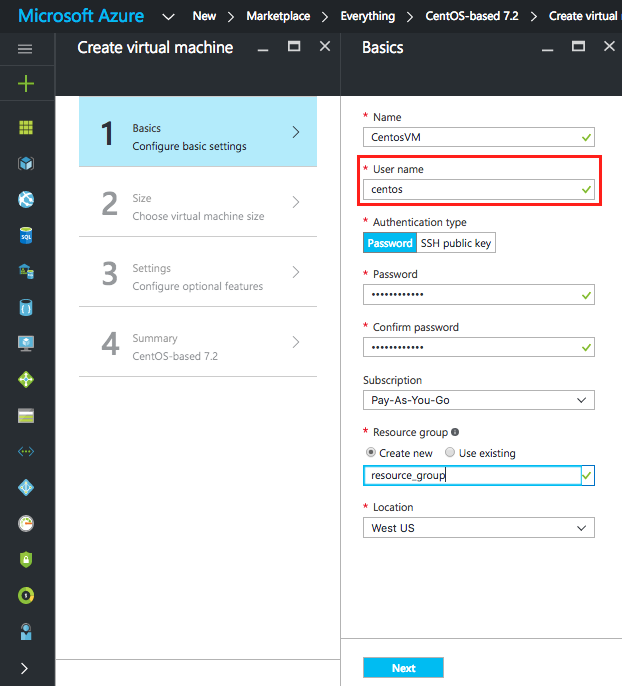
**Note:** The Azure portal is continually improved and changed. The steps in this exercise reflect the user interface of the Microsoft Azure portal at the time of writing, but may not match the latest design of portal.

## Exercises

This hands-on lab includes the following exercises:

* Exercise 1: Install Docker
* Exercise 2: Configure Test Kitchen
* Exercise 3: Write a test for Apache
* Exercise 4: Run the tests on Apache

## Exercise 1: Install ChefDK and Docker from the Command Line

1. The first task you have to launch a virtual CentOS 7.X image. Please refer to the [Module 7 Lesson 4 Lab](https://github.com/MSFTImagine/computerscience/tree/master/Complimentary%20Course%20Content/Module7/Labs) on how to launch a Linux instance. Take special not of your user name.
2. Find the IP Address of your new VM
3. Connect to the VM with a PuTTY for Windows or Terminal for Mac
4. Install the ChefDK using OmniTruck

$curl -L https://www.chef.io/chef/install.sh | sudo bash -s -- -P chefdk

Note: if the terminal hangs here, hit the <enter> key and input the ‘chef’ user password if asked

1. Set up Docker and its permissions
2. First update all the package on the CentOS machine

$ sudo yum update –y

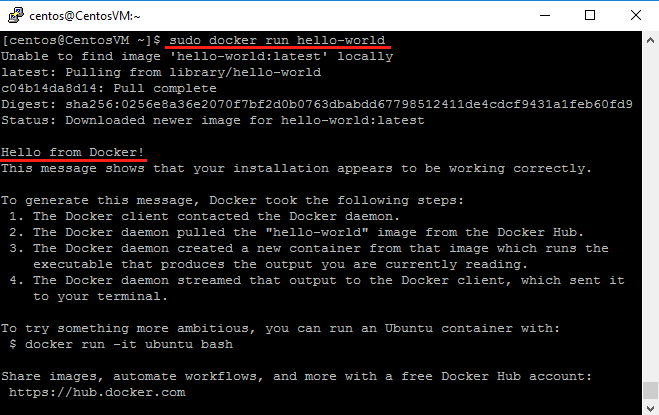
(the –y option answers ‘yes’ to any questions from yum)

1. Install Docker

$ curl -fsSL https://get.docker.com/ | sh

1. Start the Docker service

$ sudo service docker start

1. Verify that Docker was installed correctly  
   $ sudo docker run hello-world  
   
2. Create the Docker group (note this group might already exist from the installation)

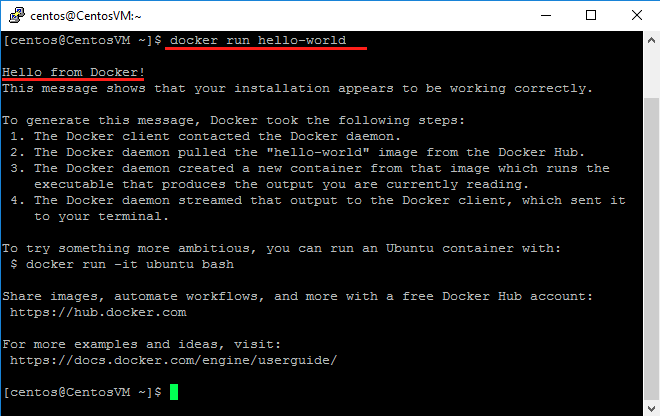
$ sudo groupadd docker

1. Add the centos user (use your user name) to the Docker group

$ sudo usermod -aG docker centos

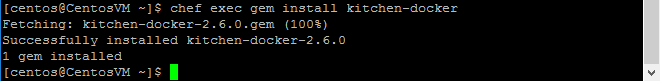
1. LOG OUT AND LOG BACK IN of your terminal or PuTTY session for the changes to take effect.
2. Now that the chef user is in the Docker group, we should be able to run:

$ docker run hello-world

If this returns successfully without running as sudo (see step 4), you were successful!

1. Install the kitchen-docker gem

$ chef exec gem install kitchen-docker



## Exercise 2: Configure Test Kitchen

Now that you’ve installed the ChefDK and Docker, let’s configure Test Kitchen

1. Create a ‘cookbooks’ directory
   1. Log into the Linux virtual machine using SSH from either Terminal (on a Mac) or PuTTY (from Windows)
   2. From the home directory, create a ‘cookbooks’ directory  
      **$mkdir cookbooks**
   3. This is where cookbooks will be stored
2. Create a new cookbook

From the home directory, use the **chef generate** command to create the apache cookbook. This will create the specific cookbook directory and its needed subdirectories and files:  
**$chef generate cookbook cookbooks/apache**  
Note: this assumes you are running the command from the home directory, hence the path of **cookbooks**/apache

1. Change directory to the Apache cookbook

$ cd ~  
$ cd cookbooks/apache

1. Edit the .kitchen.yml file as such:

---

driver:

name: docker

binary: docker

use\_sudo: false

provisioner:

name: chef\_zero

verifier:

name: inspec

platforms:

- name: centos-6.4

suites:

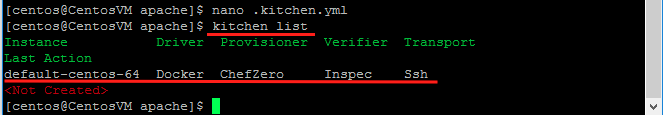
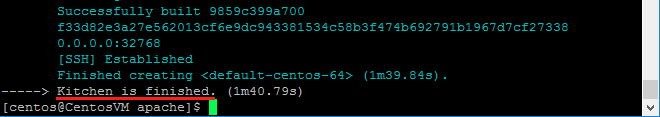
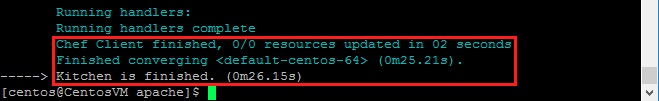
- name: default

run\_list:

- recipe[apache::default]

attributes:

---

1. Run $kitchen list to test the configuration. You should see the following:  
   
2. Run $kitchen create to see if you can successfully create the environment.  
   
3. Run $kitchen converge to see if Chef can be installed and if the recipes can be applied to the environment.  
   

## Exercise 3: Write an Integration Test

Now that you’ve configured Test Kitchen, let’s write a test.

1. Edit cookbooks/apache/test/recipes/default\_test.rb
2. Replace the text in the file with the following:

require 'spec\_helper'

describe 'apache::default' do

it "has httpd package installed" do

expect(package('httpd')).to be\_installed

end

it "displays home page" do

expect(command("curl http://localhost").stdout).to match /Hello/

end

end

## Exercise 4: Run Test Kitchen

Now that you’ve set everything up, let’s implement our integration test

1. In the /cookbooks/apache directory Run the test:

$kitchen verify

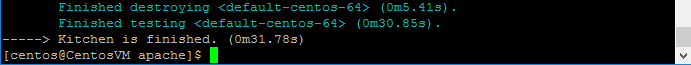
1. Look for the tests passing../../../../../Module%207%20Screenshots/Screen%20Shot%202016-10-10%20at%203.56.31%20PM.pn  
   Note: two tests, one testing that apache as installed, the second testing that the content “Hello” was found in the HTML output from the command ‘curl localhost’.   
   “0 failures” means all your tests passed.
2. Once they pass, run the entire test cycle:

$kitchen test

This will:

* 1. destroy any existing kitchen
  2. create a new kitchen
  3. install chef, copy the cookbooks over and run the recipes
  4. verify the tests
  5. destroy the kitchen
  6. report the results

Note that at the end of $kitchen test, the kitchen is destroyed.



1. Your integration testing is complete!

\*\*\*Don’t forget to delete your VM’s in order to avoid wasting your Azure credit\*\*\*

## Summary

In this hands-on lab, you learned how to:

* Install the ChefDK and Docker from the command line
* Configure Test Kitchen
* Write several integration tests
* Run Test Kitchen