**Dissection of a Chef Recipe or two for windows**

Working with chef one of the first things I needed to do was get to grips with the semantics of ruby.  I did a bit of speed reading, did a few simple ruby programs and I keep a copy of the little book of ruby handy for reference purposes so am becoming more comfortable with that.

What I found though is that it wasn’t problems with getting to grips with ruby and I’m definitely a newbie there but the actual understanding of the Chef recipe DSL. I had a look at the wiki and although it does give some guidance I was thinking that the best way to help someone get started quickly was to walk through a couple of example recipes.

There is a fair amount of information on using Chef with Linux targets so I’ll focus on using Windows as the target as I believe Chef has as much to offer the windows system administrator as it does for Linux sysadmins.

Before you start writing recipes the first thing you need to understand is the anatomy of a cookbook

Taking the definitions from the Opscode Wiki:

Cookbooks contain:

* [Attributes](http://wiki.opscode.com/display/chef/Attributes) that are values on [Node](http://wiki.opscode.com/display/chef/Nodes) to set default values used elsewhere in the cookbook.
* [Definitions](http://wiki.opscode.com/display/chef/Definitions) that allow you to create reusable collections of one or more [Resources](http://wiki.opscode.com/display/chef/Resources).
* [Files](http://wiki.opscode.com/display/chef/Files) that are transferred to your Chef-administered machines via [Cookbook File](http://wiki.opscode.com/display/chef/Resources#Resources-CookbookFile)resource.
* [Libraries](http://wiki.opscode.com/display/chef/Libraries) that extend Chef or provide helpers with Ruby code.
* [Recipes](http://wiki.opscode.com/display/chef/Recipes) that specify [Resources](http://wiki.opscode.com/display/chef/Resources) to manage, in the order they should be managed.
* [Lightweight Resources and Providers (LWRP)](http://wiki.opscode.com/display/chef/Lightweight+Resources+and+Providers+%28LWRP%29) that allow you to create your own custom resources and providers.
* [Templates](http://wiki.opscode.com/display/chef/Templates) that are rendered on Chef-configured machines with your dynamically substituted values. Think config files on steroids, then read [ERB templates](http://wiki.opscode.com/display/chef/Resources#Resources-Template).
* [Metadata](http://wiki.opscode.com/display/chef/Metadata) that tells Chef about your recipes, including dependencies, supported platforms and more.

Cookbooks are arranged in the following folder structure:

attributes/

defintions/

files/

libraries/

metadata.rb

providers/

README.rdoc

recipes/

resources/

templates/

Well I don’t know about you but as a newbie based on the above it’s a bit daunting to try and understand how it all fits together and hunting through the various pages on the Opscode wiki to understand can be slightly frustrating to say the least so I hope a walkthrough of a few simple recipes will help you get started.

Example 1:

**#**

**# Cookbook Name:: myapp**

**# Recipe:: deploymsi**

**#**

**# Copyright 2011, @devopscloud**

**#**

**# All rights reserved – Do Not Redistribute**

**#**

**msifile = File.basename(“myapp.msi”)**

**dir = “buildoutput”**

**drive=”c:”**

**msifiledst = “#{drive}\\#{dir}\\#{msifile}”**

**execute “install #{msifiledst}” do**

**command “msiexec /qn /i #{msifiledst} TARGETENV=DEV”**

**only\_if { File.exists?(msifiledst) }**

**end**

This example does what you think it does it installs an MSI on the target node.

How does it work:

Firstly we define a number of variables to allow us to identify the msi.

All the grunt work is defined in the execute resource:

**execute “install #{msifiledst}” do**

**command “msiexec /qn /i #{msifiledst} TARGETENV=DEV”**

**only\_if { File.exists?(msifiledst) }**

**end**

The resource  type is:**execute**.

The resource name is : **install #{msifiledst} = Install c:\buildoutput\myapp.msi**

It calls the command prompt and then runs msiexec but only if the msi actually exists which is what the only\_if (File.exists?..  bit of the recipe does.

Tip  the ‘\\’ to allow you to  use ‘\’  not an issue with Linux nodes but useful when working with windows nodes.

Building upon the above simple example we’ll now introduce something new in the next example:

Example 2:

**#**

**# Cookbook Name:: myapp**

**# Recipe:: deploywebapp**

**#**

**# Copyright 2011,  @devopscloud**

**#**

**# All rights reserved – Do Not Redistribute**

**#**

**msi = File.basename(“myWebapp.msi”)**

**dir = “buildoutput”**

**drive=”c:”**

**dst = “#{drive}\\#{dir}\\#{msi}”**

**template “C:/chef/tmp/appool.ps1” do**

**source “appool.ps1.erb”**

**end**

**execute “install #{dst}” do**

**command “msiexec /qn /i #{dst} TARGETENV=DEV”**

**only\_if { File.exists?(dst) }**

**end**

**execute “updateappool” do**

**command “c:\\Windows\\System32\\WindowsPowerShell\\V1.0\\powershell.exec:\\chef\\tmp\\appool.ps1\””**

**action :run**

**cwd “c:/chef/tmp”**

**end**

This recipe installs an MSI as in example 1 but it then runs a powershell script that makes modifications to the appool. This recipe introduces the concept of templates. Templates are stored in the templates folder of your cookbook and stored as .erb files. In this example the erb file contains powershell script. So what does these two line mean?

**template “C:/chef/tmp/appool.ps1” do**

**source “appool.ps1.erb”**

This essentially equates to the following:  copy the   file appool.ps1.erb  to target node to the folder c:/chef/tmp  and name accordingly.

Later on in the recipe we actually run the powershell script. Easy huh   Smile

The key thing here really is that all the Powershell you inevitably use as a windows administrator is still reusable and I haven’t even started talking about providers as yet.

The examples above are simple and not exactly robust but they do stuff which is all we want chef recipes to do really.

Recipes are quite a huge topic and I have barely  scraped the surface with these two  simple examples.