Chef CookBooks

Mohanraj Shanmugam

Cookbooks

- A cookbook is the fundamental unit of configuration and policy distribution.
- A cookbook defines a scenario and contains everything that is required to support that scenario:
  - Recipes that specify the resources to use and the order in which they are to be applied
  - Attribute values
  - File distributions
  - Templates
  - Extensions to Chef, such as custom resources and libraries

Cook Book

- The chef-client uses Ruby as its reference language for creating cookbooks and defining recipes, with an extended DSL for specific resources.
- The chef-client provides a reasonable set of resources, enough to support many of the most common infrastructure automation scenarios;
- however, this DSL can also be extended when additional resources and capabilities are required.

Recipie

- A recipe is the most fundamental configuration element within the organization.
- Is authored using Ruby, which is a programming language designed to read and behave in a predictable manner
- Is mostly a collection of resources, defined using patterns having
  - resource names,
  - attribute-value pairs, and
  - actions);
- helper code is added around this using Ruby, when needed

Recipie

- Must define everything that is required to configure part of a system
- Must be stored in a cookbook
- May be included in a recipe
- May use the results of a search query and read the contents of a data bag
- May have a dependency on one (or more) recipes
- May tag a node to facilitate the creation of arbitrary groupings
- Must be added to a run-list before it can be used by the chef-client
- Is always executed in the same order as listed in a run-list

Recipie

- The chef-client will run a recipe only when asked.
- When the chef-client runs the same recipe more than once, the results will be the same system state each time.
- When a recipe is run against a system, but nothing has changed on either the system or in the recipe, the chef-client won’t change anything.

Attribute

- An attribute can be defined in a cookbook (or a recipe) and then used to override the default settings on a node.
- When a cookbook is loaded during a chef-client run, these attributes are compared to the attributes that are already present on the node.
- Attributes that are defined in attribute files are first loaded according to cookbook order.
- For each cookbook, attributes in the default.rb file are loaded first
- Then additional attribute files (if present) are loaded in lexical sort order
- When the cookbook attributes take precedence over the default attributes, the chef-client will apply those new settings and values during the chef-client run on the node.

Resource

- Piece of the system and its desired state
- A resource instructs the chef-client to complete various tasks like installing packages, running Ruby code, or accessing directories and file systems.
- The chef-client includes built-in resources that cover many common scenarios.

Resources - Package

Package that should be installed

```
package "mysql-server" do
  action :install
end
```

Resources - Service

Service that should be running and restarted on reboot

```
service "iptables" do
  action [ :start, :enable ]
end
```

Resources - Service

File that should be generated

```
file "/etc/motd" do
  content "Property of Chef Software"
end
```

Resources - Cron

Cron job that should be configured

```
cron "restart webserver" do
  hour '2'
  minute '0'
  command 'service httpd restart'
end
```

Resources - User

User that should be managed

```
user "nginx" do
  comment "Nginx user <nginx@example.com>"
  uid 500
  gid 500
  supports :manage_home => true
end
```

Resources – Registry Key

Registry key that should be created

```
registry_key
"HKEY_LOCAL_MACHINE\\SOFTWARE\\Mic
rosoft\\Windows\\CurrentVersion\\P
olicies\\System" do
  values [{
    :name => "EnableLUA",
    :type => :dword,
    :data => 0
    }]
  action :create
end
```

Resources

- Piece of the system and its desired state

- http://docs.chef.io/chef/resources.html

chef-apply

- chef-apply is an executable program that allows you to work with resources
- Is included as part of the ChefDK
- A great way to explore resources
- NOT how you'll eventually use Chef in production

Install telnet

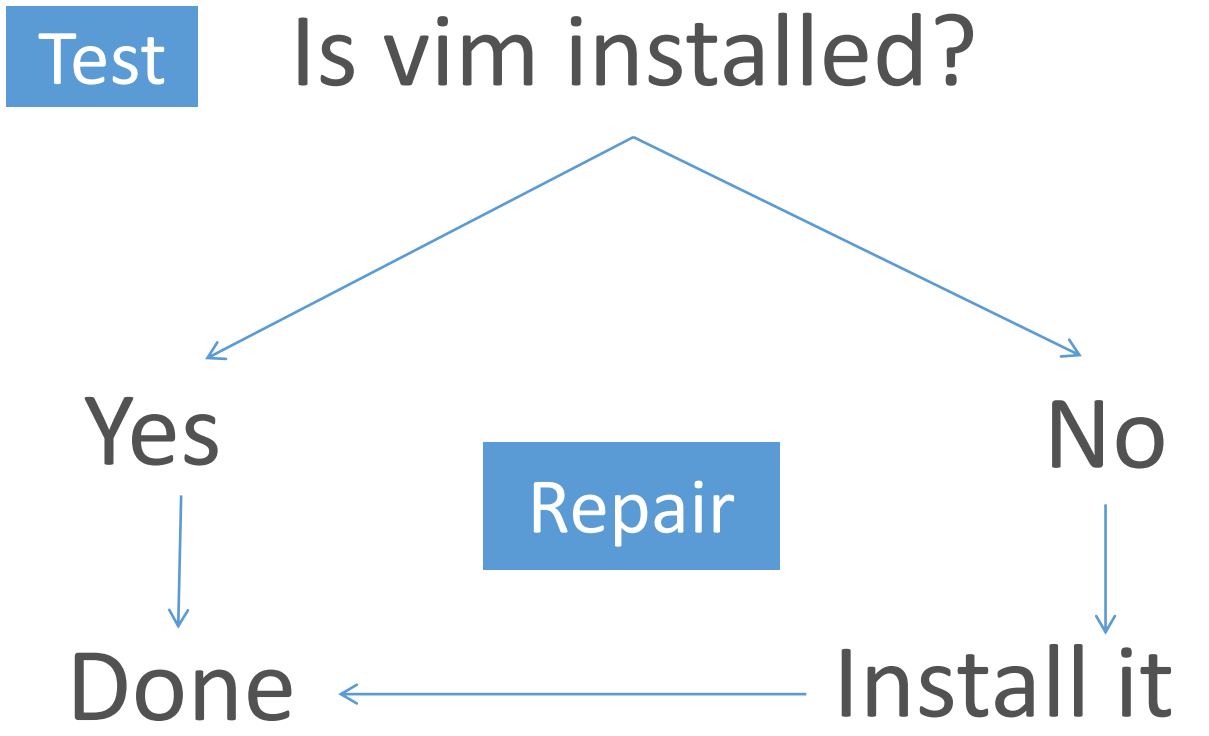
```
$ sudo chef-apply -e "package 'vim'"
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * yum_package[vim] action install
    - install version xxxxxx of package vim
```

Test and Repair

Resources follow a **test** and **repair** model

```
package "vim"
```



Templates

- A cookbook template is an Embedded Ruby (ERB) template that is used to dynamically generate static text files
- Templates may contain Ruby expressions and statements, and are a great way to manage configuration files.

```
<%= "my name is #{$ruby}" %>
```

- Use the **template** resource to add cookbook templates to recipes;
- place the corresponding Embedded Ruby (ERB) template file in a cookbook's /template directory
- To use a template, two things must happen:
  - A template resource must be added to a recipe
  - An Embedded Ruby (ERB) template must be added to a cookbook

Cookbook File

- A file distribution is a specific type of resource that tells a cookbook how to distribute files, including by node, by platform, or by file version.
- Use the cookbook_file resource to manage files that are added to nodes based on files that are located in the directory in a cookbook.

```
cookbook_file '/var/www/customers/public_html/index.php'
do
    source 'index.php'
    owner 'web_admin'
    group 'web_admin'
    mode '0755'
    action :create
end
```

Files

- Use the file resource to manage files directly on a node.

```
file '/var/www/customers/public_html/index.php' do
    content '<html>This is a placeholder for the home
page.</html>'
    mode '0755'
    owner 'web_admin'
    group 'web_admin'
end
```

Remote file

- A **remote_file** resource block manages files by using files that exist remotely.

```
remote_file '/var/www/customers/public_html/index.php' do
    source 'http://somesite.com/index.php'
    owner 'web_admin'
    group 'web_admin'
    mode '0755'
    action :create
end
```

Definition

- A definition behaves like a compile-time macro that is reusable across recipes.
- A definition is typically created by wrapping arbitrary code around resources that are declared as if they were in a recipe.
- A definition is then used in one (or more) actual recipes as if the definition were a resource.

Definition

- A definition:
  - Is not a resource or a custom resource
  - Is processed while the resource collection is compiled
  - Does not support common resource properties, such as notifies, subscribes, only_if, and not_if
  - Is defined from within the /definitions directory of a cookbook
  - Does not support why-run mode

Definition

- A definition has four components:
  - A resource name
  - Zero or more arguments that define parameters their default values; if a default value is not specified, it is assumed to be nil
  - A hash that can be used within a definition's body to provide access to parameters and their values
  - The body of the definition

```
define :host_porter, :port => 4000, :hostname => nil do
    params[:hostname] ||= params[:name]
directory '/etc/#{params[:hostname]}' do
        recursive true
    end
file '/etc/#{params[:hostname]}/#{params[:port]}' do
        content 'some content'
    end
end
```

Definition

which is then used in a recipe like this:

```
host_porter node['hostname'] do
  port 4000
end
host_porter 'www1' do
    port 4001
end
```

library

- A library allows arbitrary Ruby code to be included in a cookbook, either as a way of extending the classes that are built-in to the chef-client
- A library file is a Ruby file that is located within a cookbook's /libraries directory
- Because a library is built using Ruby, anything that can be done with Ruby can be done in a library file.

Custom Resource

- A custom resource:
  - Is a simple extension of Chef that adds your own resources
  - Is implemented and shipped as part of a cookbook
  - Follows easy, repeatable syntax patterns
  - Effectively leverages resources that are built into Chef and/or custom Ruby code
  - Is reusable in the same way as resources that are built into Chef
- A custom resource is defined as a Ruby file and is located in a cookbook's /resources directory
  - Declares the properties of the custom resource
  - Loads current properties, if the resource already exists
  - Defines each action the custom resource may take

Custom Resource

```
property :name, RubyType, default: 'value'
load_current_value do
    # some Ruby
end
action :name do
  # a mix of built-in Chef resources and Ruby
end
action :name do
  # a mix of built-in Chef resources and Ruby
end
```

Custom Resource

```
property :homepage, String, default: '<h1>Hello world!</h1>'
load_current_value do
    if ::File.exist?('/var/www/html/index.html')
        homepage IO.read('/var/www/html/index.html')
    end
end
action :create do
    package 'httpd'
service 'httpd' do
        action [:enable, :start]
    end
file '/var/www/html/index.html' do
        content homepage
    end
end
action :delete do
    package 'httpd' do
        action :delete
    end
end
```

Custom Resource

```
exampleco_site 'httpd' do
    homepage '<h1>Welcome to the Example Co. website!</h1>'
    action :create
end

exampleco_site 'httpd' do
    action :delete
end
```

Cookbook Directories and Metadata

- The cookbooks/ directory is used to store the cookbooks that are used by the chef-client when configuring the various systems in the organization.
- This directory contains the cookbooks that are used to configure systems in the infrastructure.
- Each cookbook can be configured to contain cookbook-specific copyright, email, and license data.

Cookbook Directories and Metadata

- Every cookbook requires a small amount of metadata.
- A file named metadata.rb is located at the top of every cookbook directory structure.
- The contents of the metadata.rb file provides hints to the Chef server to help ensure that cookbooks are deployed to each node correctly.
- A metadata.rb file is:
  - Located at the top level of a cookbook's directory structure
  - Compiled whenever a cookbook is uploaded to the Chef server or when the knife cookbook metadatasubcommand is run, and then stored as JSON data
  - Created automatically by knife whenever the knife cookbook create subcommand is run
  - Edited using a text editor, and then re-uploaded to the Chef server as part of a cookbook upload