Chef Resources

Chef's Fundamental Building Blocks

Objectives

After completing this module, you should be able to:

- Use Chef to install packages on your virtual workstation
- Use the **chef-apply** command
- Create a basic Chef recipe file
- Define Chef Resources

Choose an Editor

You'll need to choose an editor to edit files:

- emacs
- nano
- vi / vim

Linux Editor Reference

Tips for using these editors can be found below:

- Nano: (usually touted as the easiest editor to get started with editing through the command-line.)

```
OPEN FILE                $ nano FILENAME
WRITE (When exiting)     ctrl+x, y, ENTER
EXIT                     ctrl+x
```

- VIM: (Vim, like vi, is more complex because of its different modes.)

```
OPEN FILE                $ vim FILENAME
START EDITING            i
WRITE FILE               ESC, :w
EXIT                     ESC, :q
EXIT (don't write)       ESC, :q!
```

Group Exercise: Find an Editor

How about Nano?

```
$ which nano
/usr/bin/which: no nano in (/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

Group Exercise: Find an Editor

How about Nano?

```
$ which nano
/usr/bin/which: no nano in (/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

How about vim?

```
$ which vim
/usr/bin/which: no vim in (/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:
```

Group Exercise: Find an Editor

How about Nano?

```
$ which nano
/usr/bin/which: no nano in (/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

How about vim?

```
$ which vim
/usr/bin/which: no vim in (/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:
```

How about Emacs?

```
$ which emacs
/usr/bin/which: no emacs in (/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbi
```

Learning Chef

Learning Chef

- One of the best ways to learn a technology is to apply the technology in every situation that it can be applied.

Learning Chef

- One of the best ways to learn a technology is to apply the technology in every situation that it can be applied.

- A number of Chef tools are installed on the system so lets put them to use.

What is chef-apply?

What is chef-apply?

chef-apply is a command-line application that allows us to work with resources and recipes files.

What can chef-apply do?

```
$ sudo chef-apply --help
```

```
Usage: chef-apply [RECIPE_FILE | -e RECIPE_TEXT | -s] [OPTIONS]
        --[no-]color                 Use colored output, defaults to enabled
    -e, --execute RECIPE_TEXT        Execute resources supplied in a string
        --force-formatter            Use formatter output instead of logger output
        --force-logger               Use logger output instead of formatter output
    -F, --format FORMATTER           output format to use
    -j JSON_ATTRIBS,                 Load attributes from a JSON file or URL
        --json-attributes
    -l, --log_level LEVEL            Set the log level (debug, info, warn, error,
        --minimal-ohai               Only run the bare minimum ohai plugins chef n
        --[no-]profile-ruby          Dump complete Ruby call graph stack of entire
    -s, --stdin                      Execute resources read from STDIN
    -v, --version                    Show chef version
    -W, --why-run                    Enable whyrun mode
    -h, --help                       Show this message
```

Resources

Resources

- A resource is a statement of configuration policy.

Resources

- A resource is a statement of configuration policy.
- It describes the desired state of an element of your infrastructure and the steps needed to bring that item to the desired state.

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

Example: Package

```
package 'httpd'
```

The package named `httpd` is installed.

https://docs.chef.io/resource_package.html

Example: Service

```
service 'ntp' do
  action [ :enable, :start ]
end
```

The service named **ntp** is enabled (start on reboot) and started.

https://docs.chef.io/resource_service.html

Example: File

```
file '/etc/motd' do
  content 'This computer is the property ...'
end
```

The file name `/etc/motd` is created with content '**This company is the property ...**'

https://docs.chef.io/resource_file.html

Example: File

```
file '/etc/php.ini.default' do
  action :delete
end
```

The file name `/etc/php.ini.default` is deleted.

https://docs.chef.io/resource_file.html

Group Exercise: Using the –e Execute Option

```
$ sudo chef-apply --help
```

```
Usage: chef-apply [RECIPE_FILE | -e RECIPE_TEXT | -s] [OPTIONS]
        --[no-]color                 Use colored output, defaults to enabled
    -e, --execute RECIPE_TEXT        Execute resources supplied in a string
        --force-formatter            Use formatter output instead of logger output
        --force-logger               Use logger output instead of formatter output
    -F, --format FORMATTER           output format to use
    -j JSON_ATTRIBS,                 Load attributes from a JSON file or URL
        --json-attributes
    -l, --log_level LEVEL            Set the log level (debug, info, warn, error,
        --minimal-ohai               Only run the bare minimum ohai plugins chef n
        --[no-]profile-ruby          Dump complete Ruby call graph stack of entire
    -s, --stdin                      Execute resources read from STDIN
    -v, --version                    Show chef version
    -W, --why-run                    Enable whyrun mode
    -h, --help                       Show this message
```

Group Exercise: Install nano, emacs or vim

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

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * yum_package[nano] action install
    - install version 2.0.9-7.el6 of package nano
```

Group Exercise: Did I install my editor?

```
$ which nano
/bin/nano
```

Group Exercise: Test and Repair

- What would happen if you ran the installation command again?

Group Exercise: Test and Repair

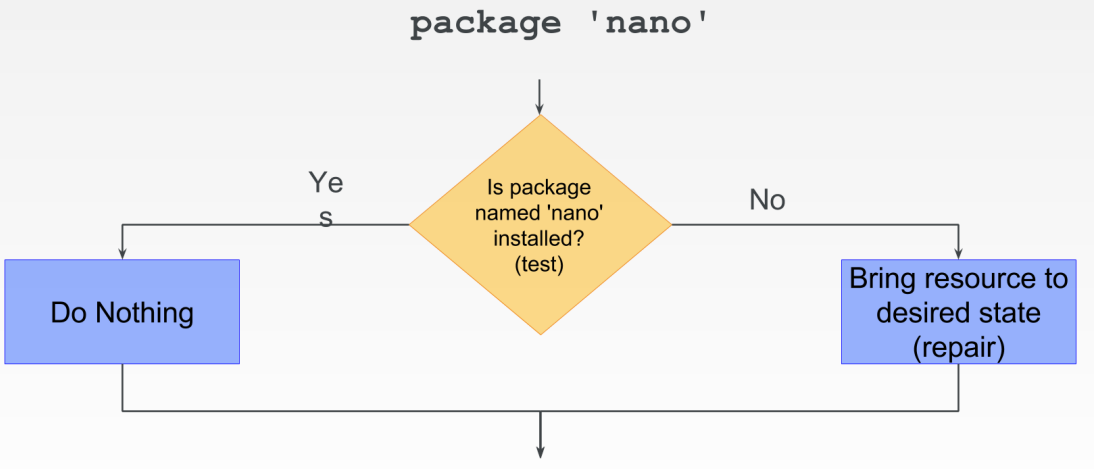
- What would happen if you ran the installation command again?

- What would happen if the package were to become uninstalled?

Test and Repair

- chef-apply takes action only when it needs to. Think of it as test and repair.
- Chef looks at the current state of each resource and takes action only when that resource is out of policy.

Test and Repair



CHEF

Group Exercise: Hello, World?

"I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application."

Group Exercise: Hello, World?

"I heard Chef is written in Ruby. If that's the case its required that we write a quick "Hello, world!" application."

Objective:

- Create a recipe file that defines the policy,
- The file named `hello.txt` is created with the content "Hello, world!".

Group Exercise: Create and Open a Recipe File

```
$ cd ~
$ nano hello.rb
```

Group Exercise: Create a Recipe File Named hello.rb

Edit `~/hello.rb`

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The file named `hello.txt` is created with the content 'Hello, world!'

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

Group Exercise: Can chef-apply Run a Recipe File?

```
$ sudo chef-apply --help
```

```
Usage: chef-apply [RECIPE_FILE | -e RECIPE_TEXT | -s] [OPTIONS]
        --[no-]color                 Use colored output, defaults to enabled
    -e, --execute RECIPE_TEXT        Execute resources supplied in a string
        --force-formatter            Use formatter output instead of logger output
        --force-logger               Use logger output instead of formatter output
    -F, --format FORMATTER           output format to use
    -j JSON_ATTRIBS,                 Load attributes from a JSON file or URL
        --json-attributes
    -l, --log_level LEVEL            Set the log level (debug, info, warn, error,
        --minimal-ohai               Only run the bare minimum ohai plugins chef n
        --[no-]profile-ruby          Dump complete Ruby call graph stack of entire
    -s, --stdin                      Execute resources read from STDIN
    -v, --version                    Show chef version
    -W, --why-run                    Enable whyrun mode
    -h, --help                       Show this message
```

Group Exercise: Apply a recipe file

```
$ sudo chef-apply hello.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * file[hello.txt] action create
    - create new file hello.txt
    - update content in file hello.txt from none to 315f5b
    --- hello.txt    2016-04-11 15:01:29.875000726 +0000
    +++ ./.hello.txt20160411-7152-1d3f0yn    2016-04-11 15:01:29.875000726 +0000
    @@ -1 +1,2 @@
    +Hello, world!
```

Group Exercise: What does `hello.txt` say?

```
$ cat hello.txt
Hello, world!
```

Group Exercise: Test and Repair

- What would happen if you ran the command again?

Group Exercise: Test and Repair

- What would happen if you ran the command again?

- What would happen if the file contents were modified?

Group Exercise: Test and Repair

- What would happen if you ran the command again?
- What would happen if the file contents were modified?
- What would happen if the file were removed?

Group Exercise: Test and Repair

- What would happen if you ran the command again?
- What would happen if the file contents were modified?
- What would happen if the file were removed?
- What would happen if the file permissions (mode), owner, or group changed?
  - Have we defined a policy for these attributes?

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with ATTRIBUTES

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with ATTRIBUTES

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with ATTRIBUTES

Example Resource Types:

cron, directory, dsc_resource, dsc_script, env, file, git, group, ifconfig, link, log, mdadm, mount, package, reboot, registry_key, remote_directory, remote_file, route, script, service, template, user, windows_package, windows_service and more...

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with ATTRIBUTES

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with ATTRIBUTES

Resource Definition

```
file 'hello.txt' do
  content 'Hello, world!'
end
```

The TYPE named NAME should be **ACTION'd** with ATTRIBUTES

*SO WHAT IS **ACTION'd**?*

Lab 2.1: The file Resource

- Read https://docs.chef.io/resource_file.html
- Discover the file resource's:
  - default action.
  - default values for mode, owner, and group.
- Update the file policy in `hello.rb` as follows:
  - The file named 'hello.txt' should be created with...
    - the content set to 'Hello, world!',
    - the mode set to '0644',
    - the owner is 'root',
    - and the group is 'root'.

Lab 2.1: The Updated file Resource

`~/hello.rb`

```
file 'hello.txt' do
  content 'Hello, world!'
  mode '0644'
  owner 'root'
  group 'root'
  action :create
end
```

- The default mode is set by the POSIX Access Control Lists.
- The default owner is the current user (could change).
- The default group is the POSIX group (if available).
- The default action is to create (not necessary to define it).

Lab 2.2: workstation Setup Recipe

Create a recipe file named `setup.rb` that defines the policy:

- The package named **nano** is installed.
- The package named **tree** is installed.
- The file named `/etc/motd` is created with the content 'Property of ...'.

Use **chef-apply** to apply the recipe file named `setup.rb`

Lab 2.2: workstation Setup Recipe File

`~/setup.rb`

```
package 'nano'
package 'vim'
package 'emacs'

package 'tree'

file '/etc/motd' do
  content 'Property of ...'
end
```

- The package named **nano** is installed.
- The package named **tree** is installed.
- The file named `/etc/motd` is created with the content 'Property of ...'.

Lab 2.2: Apply the Setup Recipe

```
$ sudo chef-apply setup.rb
```

```
Recipe: (chef-apply cookbook)::(chef-apply recipe)
  * yum_package[nano] action install
    - install version 2.0.9-7.el6 of package nano
  * yum_package[vim] action install
    - install version 7.4.629-5.el6 of package vim-enhanced
  * yum_package[emacs] action install
    - install version 23.1-28.el6 of package emacs
  * yum_package[tree] action install
    - install version 1.5.3-3.el6 of package tree
  * file[/etc/motd] action create
    - update content in file /etc/motd from e3b0c4 to d100eb
    --- /etc/motd    2013-06-07 09:31:32.000000000 -0500
    +++ /etc/.chef-motd20160404-20599-3b66qa    2016-04-04 08:01:19.673005668 -050
    @@ -1 +1,2 @@
    +Property of ...
```

Discussion

- What is a resource?

Discussion

- What is a resource?

- What are some other possible examples of resources?

Discussion

- What is a resource?
- What are some other possible examples of resources?
- How did the example resources we wrote describe the desired state of an element of our infrastructure?

Discussion

- What is a resource?
- What are some other possible examples of resources?
- How did the example resources we wrote describe the desired state of an element of our infrastructure?
- What does it mean for a resource to be a statement of configuration policy?

Q&A

What questions can we answer for you?

- chef-apply
- Resources
- Resource - default actions and default attributes
- Test and Repair

On to Module 3