Chef Resources

Chef's Fundamental Building Blocks

CHEF

Objectives

After completing this module, you should be able to:

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

GL: Time for Some Fun!

The workstation needs a little personal touch; something that makes it a little more fun.

Objective:

- ❑ Write a recipe that installs the 'cowsay' package
- ❑ Apply the recipe to the workstation
- ❑ Use 'cowsay' to say something

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.

Choose an Editor

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

Tips for using these editors can be found below in your participant guide.

emacs

nano

vi / vim

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' do
  action :install
end
```

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 computer 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

CHEF

GL: Use Your Editor to Open the Recipe

```
$ nano setup.rb
```

GL: Update the Setup Recipe

~/setup.rb

```
package 'cowsay' do
  action :install
end
```

GL: Time for Some Fun!

The workstation needs a little personal touch; something that makes it a little more fun.

Objective:

- ✓ Write a recipe that installs the 'cowsay' package
- ❑ Apply the recipe to the workstation
- ❑ Use 'cowsay' to say something

chef-client

chef-client is an agent that runs locally on every node that is under management by Chef.

When a chef-client is run, it will perform all of the steps that are required to bring the node into the expected state.

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

CHEF

--local-mode (or -z)

chef-client's default mode attempts to contact a Chef Server and ask it for the recipes to run for the given node.

We are overriding that behavior to have it work in a local mode.

GL: Apply the Setup Recipe

```
$ sudo chef-client --local-mode setup.rb
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-02-19T13:08:13+00:00] WARN: Node ip-172-31-12-176.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[nano] action install
    - install version 3.03-8.e16 of package cowsay
Running handlers:
Running handlers complete
Chef Client finished, 1/1 resources updated in 38 seconds
```

GL: Time for Some Fun!

The workstation needs a little personal touch; something that makes it a little more fun.

Objective:

- ✓ Write a recipe that installs the 'cowsay' package
- ✓ Apply the recipe to the workstation
- ❑ Use 'cowsay' to say something

GL: Run cowsay with a Message

```
$ cowsay moo
```

```
 _____
< moo >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

GL: Time for Some Fun!

The workstation needs a little personal touch; something that makes it a little more fun.

Objective:

✓ Write a recipe that installs the 'cowsay' package
✓ Apply the recipe to the workstation
✓ Use 'cowsay' to say something

Discussion

1. What would happen if you applied the recipe again?

2. What would happen if the package were to become uninstalled?

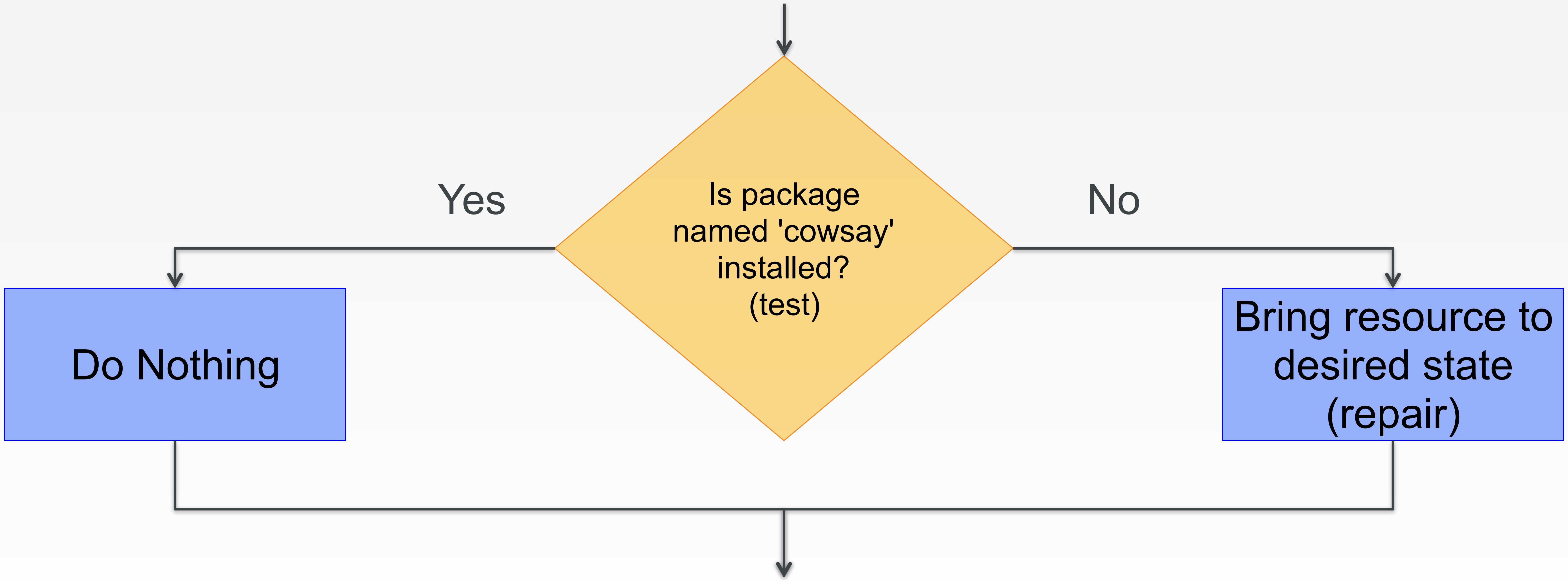
Test and Repair

`chef-client` 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

```
package 'cowsay'
```


Is package named 'cowsay' installed? (test)
Yes
No
Do Nothing
Bring resource to desired state (repair)

GL: 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 that writes out a file with the contents "Hello, world!"
- ❑ Apply that recipe to the workstation
- ❑ Verify the contents of the file

GL: Create and Open a Recipe File

```
$ nano hello.rb
```

GL: Create a Recipe File Named hello.rb

```
~/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

GL: 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 that writes out a file with the contents "Hello, world!"
- ❑ Apply that recipe to the workstation
- ❑ Verify the contents of the file

GL: Apply the Recipe File

```
$ sudo chef-client --local-mode hello.rb
```

```
Starting Chef Client, version 12.5.1
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Compiling Cookbooks...
[2016-02-19T13:08:13+00:00] WARN: Node ip-172-31-12-176.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/chef/hello.rb
  * file[hello.txt] action create
    - create new file hello.txt
    - update content in file hello.txt from non to 315f5b
    +++ ./.hello.txt20160224-8559-19kqial
        2016-02-24 16:51:04.400844959 +0000
    @@ -1 +1,2 @@
    +Hello, world!
```

GL: 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 that writes out a file with the contents "Hello, world!"
- ✓ Apply that recipe to the workstation
- ❑ Verify the contents of the file

GL: What Does hello.txt Say?

```
$ cat hello.txt
```

```
Hello, world!
```

GL: 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 that writes out a file with the contents "Hello, world!"
- ✓ Apply that recipe to the workstation
- ✓ Verify the contents of the file

Discussion

What would happen if the 'hello.txt' file contents were modified?

Test and Repair

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**

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**

Lab: The `file` Resource

- ❑ **Read** https://docs.chef.io/resources.html
- ❑ **Discover the file resource's:**
  - default action.
  - default values for **`mode`**, **`owner`**, and **`group`**.

- ❑ **Update the `file` policy in "hello.rb" to:**

  The file named 'hello.txt' should be created with the content 'Hello, world!', mode '0644', owner is 'root', and group is 'root'.

Lab: 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: The `file` Resource

✓ **Read** https://docs.chef.io/resources.html

✓ **Discover the file resource's:**

- default action.
- default values for **`mode`**, **`owner`**, and **`group`**.

✓ **Update the `file` policy in "hello.rb" to:**

The file named 'hello.txt' should be created with the content 'Hello, world!', mode '0644', owner is 'root', and group is 'root'.

Lab: Workstation Setup

- ❑ Create a recipe file named "setup.rb" that defines the policy:
  - The package named 'cowsay' is installed.
  - The package named 'tree' is installed.
  - The file named '/etc/motd' is created with the content 'Property of ...'.
- ❑ Use chef-client to apply the recipe file named "setup.rb"

Lab: Workstation Setup Recipe File

~/setup.rb

```
package 'cowsay' do
  action :install
end

package 'tree' do
  action :install
end

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

The package named 'cowsay' is installed.

The package named 'tree' is installed.

The file named '/etc/motd' is created with the content 'Property of ...'.

GL: Apply the Recipe File

```
$ sudo chef-client --local-mode setup.rb
```

```
Converging 3 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[cowsay] action install (up to date)
  * 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       2010-01-12 13:28:22.000000000 +0000
    +++ /etc/.motd20160224-8754-1xczeyn 2016-02-24 16:57:57.203844958 +0000
    @@ -1 +1,2 @@
    +Property of ...
Running handlers:
Running handlers complete
Chef Client finished, 2/3 resources updated in 17 seconds
```

Lab: Workstation Setup

✓ Create a recipe file named "setup.rb" that defines the policy:
  - The package named 'cowsay' is installed.
  - The package named 'tree' is installed.
  - The file named '/etc/motd' is created with the content 'Property of ...'.

✓ Use chef-client to apply the recipe file named "setup.rb"

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?

CHEF™