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

CONCEPT

Resources

A resource is a statement of configuration policy.

Describes the desired state of an element of your infrastructure

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

CONCEPT

Resource Definition

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

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

CONCEPT

Resource Definition

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

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

CONCEPT

Resource Definition

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

The TYPE named NAME should be ACTION'd with ATTRIBUTES

CONCEPT

Resource Definition

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

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

Example: ‘package’ resource

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

The package named 'httpd' should be installed.

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

CHEF

Example: ‘service’ resource

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

The service named 'ntp' should be enabled (start on reboot) and started.

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

CHEF

Example: ‘file’ resource

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

The file name '/etc/motd' should exist with content 'This company is the property ...'

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

Example: ‘file’ resource

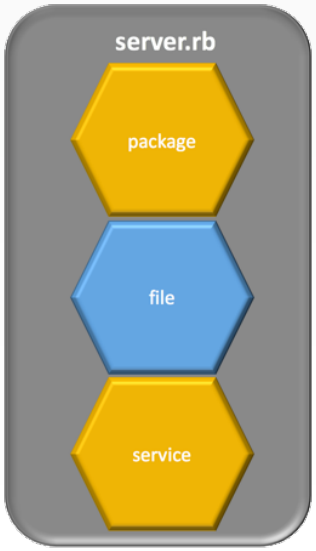
```
file '/etc/php.ini.default' do
  action :delete
end
```

The file name '/etc/php.ini.default' should be deleted if it exists

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

CONCEPT

Recipes



Resources are included in recipes.

Recipes are Ruby files (.rb extension) in which you include the code that defines your infrastructure.

CHEF

Lab: Workstation Setup

Objective:

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

Lab: Is the node already in the desired state?

```
$ which tree
```

```
/usr/bin/which: no tree in
(/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/home/chef/bin)
```

```
$ cat /etc/motd
```

Lab: Use Your Editor to Open the Recipe

```
$ [EDITOR NAME] setup.rb
```

Use the editor of your choice to create a file called setup.rb – nano, vi, vim, emacs

Lab: Final ‘setup.rb’ Recipe

~/setup.rb

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

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  action :create
end
```

SAVE FILE

CONCEPT

chef-client

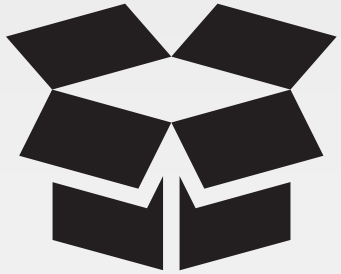
chef-client is an agent that runs locally on every node that is under management by Chef.

When a chef-client is run, it will execute the resources in the recipe to bring the node into the expected state.

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

CONCEPT

--local-mode



chef-client's default mode attempts to contact a Chef Server and ask it for the recipes to run for the given node. More on this later in the course.

We'll override that behavior to have it work in a local mode – i.e. feed it the recipes to run from local machine

'local mode' is often referred to as 'chef zero'

CHEF

Lab: Apply the Recipe File

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

```
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * 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, 2/2 resources updated in 17 seconds
```

Lab: Is the node in the desired state?

```
$ which tree
```

```
/usr/bin/tree
```

```
$ cat /etc/motd
```

```
Property of [INSERT DESIRED TEXT]
```

Lab: Setting Resource Attributes

We'll use the docs to discover the file resource's default values for:

- `mode`,
- `owner`
- `group`

Update the `file` resource in "setup.rb" to do the following:

The file named '/etc/motd' should be created with the content '**Property of [INSERT DESIRED TEXT]**', mode '**0644**', owner is '**chef**', and group is '**chef**'.

More information at https://docs.chef.io/resources.html

Lab: Final ‘setup.rb’ Recipe

~/setup.rb

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

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  action :create
  mode '0644'
  owner 'chef'
  group 'chef'
end
```

SAVE FILE

Lab: Rerun chef-client

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

```
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create
    - change owner from 'root' to 'chef'
    - change group from 'root' to 'chef'

Running handlers:
Running handlers complete
Chef Client finished, 1/2 resources updated in 08 seconds
```

Test and Repair

1. What would happen if you ran the chef-client command again?

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

3. What would happen if the /etc/motd file was manually edited or owner & group changed?

Test and Repair

1. What would happen if you ran the chef-client command again?

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

3. What would happen if the /etc/motd file was manually edited or owner & group changed?

4. Try it – set owner & group to ‘root’

Lab: Final ‘setup.rb’ Recipe

~/setup.rb

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

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  action :create
  mode '0644'
  owner 'root'
  group 'root'
end
```

SAVE FILE

Lab: Rerun chef-client

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

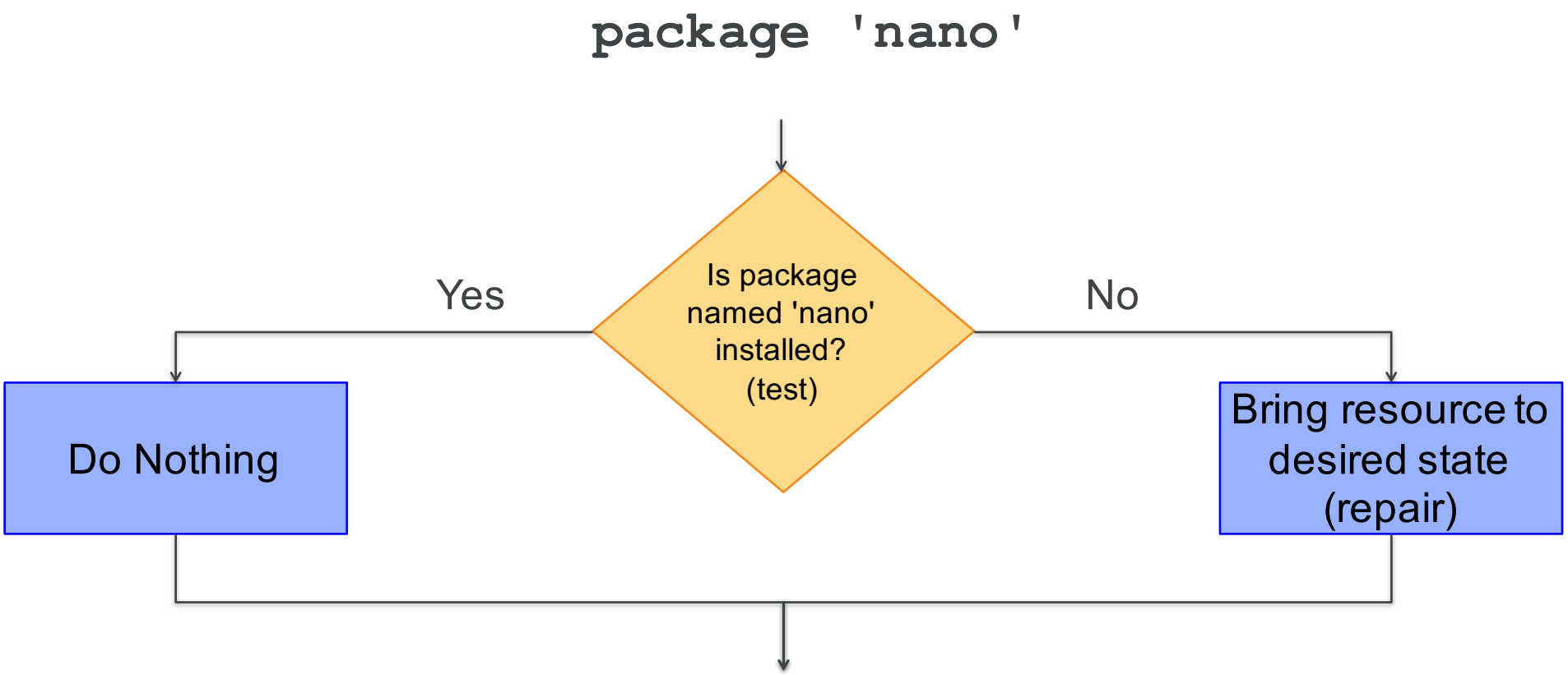
```
Converging 2 resources
Recipe: @recipe_files::/home/chef/setup.rb
  * yum_package[tree] action install (up to date)
  * file[/etc/motd] action create
    - change owner from 'root' to 'root'
    - change group from 'root' to 'root'

Running handlers:
Running handlers complete
Chef Client finished, 1/2 resources updated in 08 seconds
```

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



Remove default actions

~/setup.rb

```
package 'tree'

file '/etc/motd' do
  content 'Property of [INSERT DESIRED TEXT]'
  mode '0644'
  owner 'root'
  group 'root'
end
```

Default actions can be omitted

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?

Q&A

What questions can we answer for you?

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

CHEF™