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-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

DOCS

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

Chef Resources

- Chef **resource** is a statement of configuration policy that describes the desired state of a node.
- It’s the lowest level entity and the building block of Chef

Common Resource Examples

- Cron
- Execute
- File
- Git
- Package
- Python
- Script
- Ruby
- Service
- Template
- User
- many more (https://docs.chef.io/resources.html)

Chef Resources Syntax

```
type 'name' do                 # which resource 'name' to use
  attribute 'value'            # attributes you want to pass
  action :type of action       # what action you want to take
end                            # end
```

**

Action *- There's always a default action in every resource. If you do not specify an action, default action will be run*
Attributes *– List of attributes defined for each resource and the list is different for every resource*

Example: Package

```
package apache2' do
  action :install
end
```

The package named apache2' is installed.

Test :–

http://<IP>:80

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

Example: Service

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

The package named ntp' is installed and started

Test :–

ntpq -pn

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 ...'

Test:-

cat /etc/motd

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

GL: Use Your Editor to Open the Recipe

```
$ nano moo.rb
```

GL: Update the Moo Recipe

~/moo.rb

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

```
Test:-
       moo cow moos
```

CHEF

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 perform all of the steps that are required to bring the node into the expected state.

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

CHEF

CONCEPT

--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.

CHEF

GL: Apply the Setup Recipe

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

```
Starting Chef Client, version 12.13.37
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks...
[2016-08-22T20:20:45+00:00] WARN: Node ip-172-31-9-151.ec2.internal has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/chef/moo.rb
  * yum_package[cowsay] action install
    - install version 3.03-8.el6 of package cowsayRunning handlers:

Running handlers complete
Chef Client finished, 1/1 resources updated in 01 minutes 25 seconds
```

GL: Run cowsay with a Message

```
$ cowsay will moo for food
```

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

CHEF

Discussion

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

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

CONCEPT

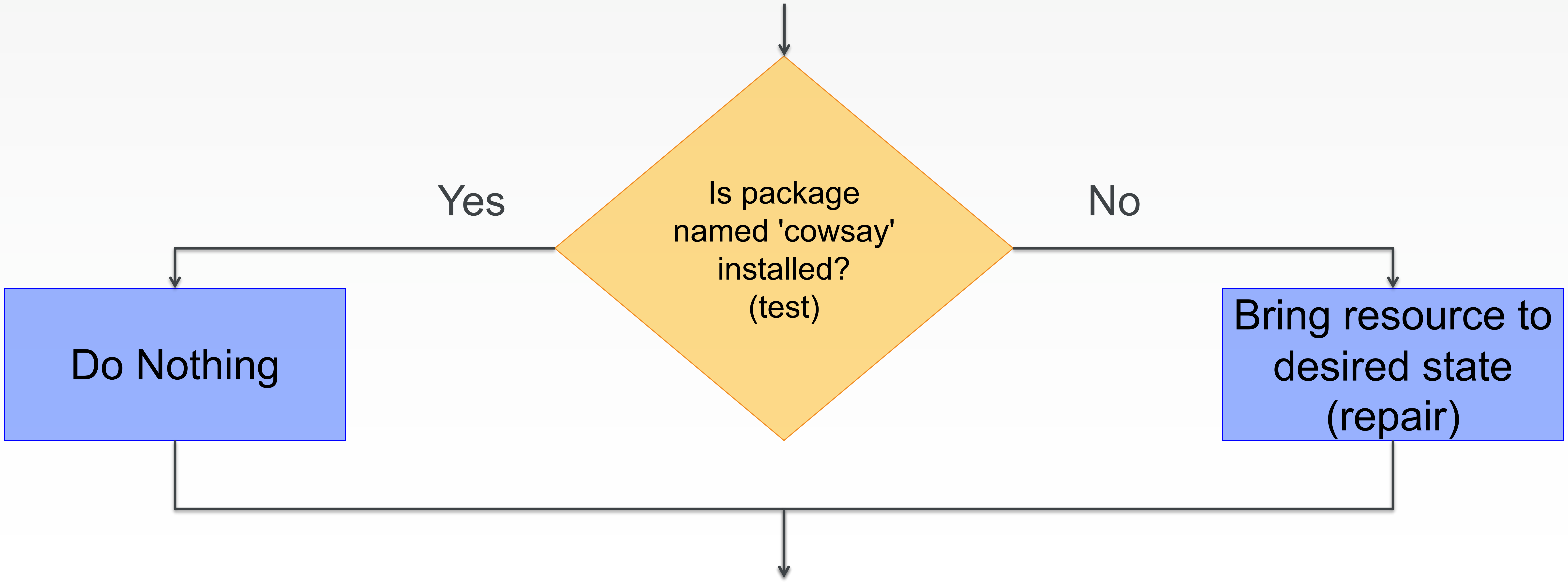
Test and Repair

`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!’

Test:-

cat /hello.txt

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

CHEF™

GL: Apply the Recipe File

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

```
Starting Chef Client, version 12.13.37
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: What Does hello.txt Say?

```
$ cat /hello.txt
```

```
Hello, world!
```

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 properties?

CONCEPT

Resource Definition

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

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

CONCEPT

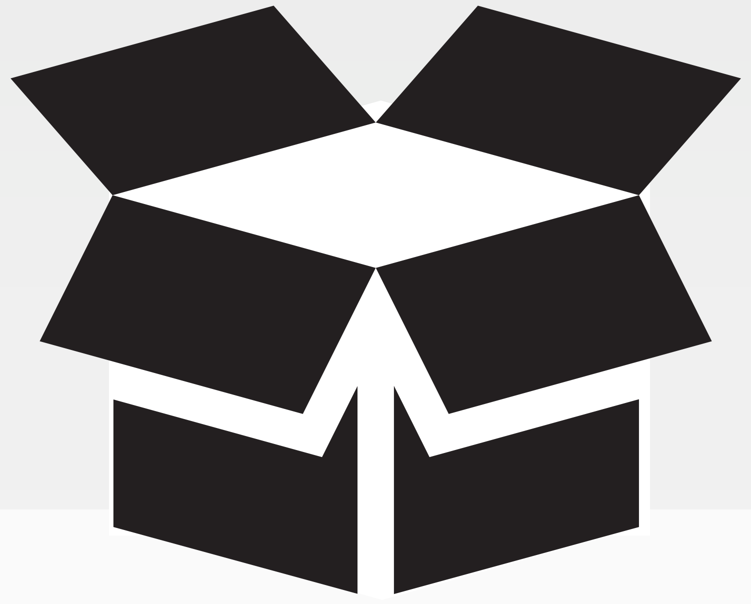
Resource Definition

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

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

CONCEPT

Resource Definition



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

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

CONCEPT

Resource Definition

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

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

CONCEPT

Resource Definition

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

?

The TYPE named NAME should be ACTION'd with PROPERTIES

GL: 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'.

GL: 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).

More on Chef Resources

Chef Resources - Cron

Manages cron entries for time-based job scheduling

```
cron 'cookbooks_report' do
  action :create   (or delete or nothing)
  minute '0'
  hour '0'
  weekday '1'
  user 'getchef'
  mailto 'sysadmin@example.com'
  home '/srv/supermarket/shared/system'
  command %W{
    cd /srv/supermarket/current &&
    env RUBYLIB="/srv/supermarket/current/lib"
    RAILS_ASSET_ID=`git rev-parse HEAD` RAILS_ENV="#{rails_env}"
    bundle exec rake cookbooks_report
  }.join(' ')
end
```

https://docs.chef.io/resource_reference.html#cron-resource

Chef Resources - Execute

Execute a single command. Since they are environment independent,
Use not_if and only_if for running specific commands in specific environments

```
execute 'myclass_run' do
  command 'java -cp myjar.jar com.mypackage.myClass'
  action :run    (or nothing)
end
```

https://docs.chef.io/resource_reference.html#execute-resource

Chef Resources - File

Manage files directly on a node

```
file 'index.php' do
  content 'home page.'
  mode '0755'
  backup true, 3
  owner 'web_admin'
  group 'web_admin'
  action :create    (or delete or touch or create_if_missing)
end
```

https://docs.chef.io/resource_reference.html#file-resource

Chef Resources - Git

Manage source control resources that exist in a git repository

```
git "/path/to/check/out/to" do
  repository "git://github.com/source/path/project.git"
  reference "master"
  action :sync (or checkout or export or nothing)
end
```

https://docs.chef.io/resource_reference.html#git-resource

Chef Resources - Package

Manage packages

Install single package

```
package 'tree' do
  action :install
  source "/home/packages/tree.deb"
  version '1.6.0-5.8.amzn1'
end
```

Install multiple packages

```
package %w(tree wget) do
  version [ '1.1.1', '1.1.1']
end
```

Install package using case statement

```
case node[:platform]
when 'ubuntu','debian'
  package 'apache2' do
    action :install
  end
when 'centos','redhat','rhel','fedora','amazon','oracle'
  package 'httpd' do
    action :install
  end
end
```

https://docs.chef.io/resource_reference.html#package-resource

Chef Resources - Python

Execute scripts using the Python interpreter

Run python using "execute" resource

```
filename = "test.py"
execute 'execute_file' do
 cwd '/home/ec2-user/code'
 command "python #{filename}"
end
```

test.py used in the above recipe

```
file = open('sampletext.txt', 'w')
file.write('content by Chef Execute')
file.close()
```

Run python using "python" resource

```
python 'execute' do
  code <<-EOH
file = open('sampletext2.txt', 'w')
file.write('content by Chef Python')
file.close()
  EOH
  action :run
end
```

https://docs.chef.io/resource_reference.html#python-resource

CHEF

Chef Resources - Script

Execute scripts using a specified interpreter, such as Bash, csh, Perl, Python, or Ruby.

This resource is the base resource for several other resources used for scripting on specific platforms.

```
extract_path = ’/foo’
script 'extract_module' do
  interpreter "bash"
  cwd ::File.dirname(src_filepath)
  code <<-EOH
    mkdir -p #{extract_path}
    EOH
 action :run  (or nothing)
end
```

https://docs.chef.io/resource_reference.html#script-resource

CHEF

Chef Resources - Service

Manage services

Single action

```
service "tomcat" do
  action :nothing  (or start or stop or restart or enable or disable)
end
```

Multiple actions

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

https://docs.chef.io/resource_reference.html#service-resource

Workstation Setup

- ❑ 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 to apply the recipe file named "setup.rb"

Workstation Setup Recipe File

~/setup.rb

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

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

The package named 'tree' is installed.

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

CHEF

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:
30 minutes

https://github.com/shekhar2010us/chef-essentials-repo-15/blob/master/labs/chapter%202.md

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-client
- Resources
- Resource - default actions and default properties
- Test and Repair

CHEF™