Chef

Module-2

Agenda

1. **Ruby Language**
2. **Authoring Cookbooks**
   1. Recipes
   2. Resources
3. **Cookbook Ecosystem**
   1. Attributes
   2. Templates
   3. Roles

Chef

Just Enough Ruby

Ruby Programming Language

1. **Written by Yukihiro Matsumoto**
2. **Since 1993**
3. **Features**
    - 3.1. Open Source
    - 3.2. Object Oriented
    - 3.3. Interpreted
    - 3.4. Server-side scripting
    - 3.5. HTML Embeddable
    - 3.6. Language for languages (DSL)
    - 3.7. Case-senstive
    - 3.8. Dynamic
4. **Close to C and Smaltalk**
5. **Comes also as part of Chef**
6. `ruby -c rkt.rb`

Ruby: Basics

1. **Comments**
   1.1. `# this is a comment`
   1.2. `=begin`
   1.3. `this is a block comment`
   1.4. `=end`
2. **Constants**
   2.1. `PI = 3.14`
3. **Variables**
   3.1. `area – local variable`
   3.2. `$locale – global variables`
   3.3. `@area – instance variable`
   3.4. `@@uuid – class variable`

Ruby: Decisions

```ruby
x = 1
if x > 2
   puts "x is greater than 2"
elsif x <= 2 and x!=0
   puts "x is 1"
else
   puts "I can't guess the number"
end
```

Ruby: Decisions

```ruby
$age =  5
case $age
when 0 .. 2
   puts "baby"
when 3 .. 6
   puts "little child"
when 7 .. 12
   puts "child"
when 13 .. 18
   puts "youth"
else
   puts "adult"
end
```

Ruby: Loops

```ruby
$i = 0
$num = 5
begin
   puts("Inside the loop i = #$i" )
   $i +=1
end while $i < $num
```

```ruby
for i in 0..5
   puts "Value of local variable is #{i}"
end
```

Ruby: Strings

```ruby
x, y, z = 12, 36, 72
puts "The value of x is #{ x }."
puts "The sum of x and y is #{ x + y }."
puts "The average was #{ (x + y + z)/3 }."
```

Ruby: Arrays and Hashes

```
nums = [1, 2, 3, 4, 5]
nums = nums + 6
puts "#{nums[2]}"
digits =(0..9)
%w(a b c )


H ={“a" => 100, "b" => 200}
puts "#{H['a']}"
H.keys
H.values
```

Ruby: Classes and Objects

```ruby
class Customer
   @@no_of_customers = 0
   def initialize(id, name, addr)
      @cust_id = id
      @cust_name = name
      @cust_addr = addr
   end

   def hello
      puts “Hello #{@cust_name}!”
   end
end

c = Customer.new("1", "abc", "xyx")
c.hello
```

Exercises

1. Write a Ruby script to print the following
   1.1. Maximum value of an array
   1.2. Minimum value of an array
   1.3. Average value of an array
2. Write a Ruby script to sort an array
3. Write a Ruby script to print factorial
4. Write and use a class ArrayAnalyzer with methods for min, max and average

Chef

Authoring Cookbooks

Cookbook Fundamentals

1. **Chef uses cookbooks to bring a node into a specific state**
    1.1. They are the fundamental unit of configuration and policy details
2. **Cookbooks are organised in a directory structure**
    2.1. recipes collections of resources
    2.2. attributes: key-value settings
    2.3. files: static files to be placed on the node
    2.4. templates: code to generate files dynamically
    2.5. libraries: code to extend Chef
    2.6. metadata.rb: dependency details of cookbooks

Basic Operations

1. **Generating a cookbook**
    1.1. `cd /path/to/chef-repo/cookbooks`
    1.2. `chef generate cookbook rkt-cb`
2. **Author the cookbook**
    2.1. `rkt-cb/recipes/default.rb`
    2.2. `attributes, templates and etc.,`
3. **Inspect the cookbook**
    3.1. `cookstyle rkt-cb`
4. **Test the cookbook**
5. **Upload the cookbook**
    5.1. `knife cookbook upload rkt.rb`
6. **Update the run-list**
    6.1. `knife node run_list add rkt-node ‘recipe[rkt-cb::default]'`
7. **Run the cookbook**
    7.1. `knife ssh -i rkt.pem ‘name:rkt-node' 'sudo chef-client' -x root`

Recipe

1. **Collection of ordered resources**
   1.1. type: around 100 predefined types
      1.1.1. *package, template, service, file, log, route and et.,*
   1.2. name: unique within the recipe
   1.3. parameters: pre-defined based on resource type
   1.4. action: pre-defined based on resource type
   1.5. notifications: pre-defined
2. **Resources are declarative**

```
type  'name' do
  parameter 'value'
  parameter 'value'
  action :type | [type, type, ...]
  notifies :type, type, ...
end
```

Resource: `package`

1. **Manages a package on a node**
2. **Properties**
   1. `options` - install operations
   2. `source` - path to the package on the local system
   3. `version` - version to be installed or upgraded
   4. `release` - specific release like stable
   5. `retries` - number of retries
   6. `retry_delay` - delay from the last attempt
   7. `timeout` - in seconds
3. **Actions**
   1. `:install` - installs the package (default)
   2. `:purge` and `:remove` - removes the package
   3. `:upgrade` - upgrades to the specified version

Resource: package

```
package("mongo-10gen-server") do
 action [:install]
 retries 0
 retry_delay 2
 package_name "mongo-10gen-server"
 options --smallfiles
end
```

Resource: `service`

1. **Manages a service on a node**
2. **Properties**
    1. `start_command` - command to start
    2. `reload_command` - command to reload
    3. `restart_command` - command to restart
    4. `stop–command` - command to stop
    5. `init_command` - overrides start/restart
    6. `timeout` - in seconds
3. **Actions**
    1. `:nothing` - do nothing (default)
    2. `:disable` or `:enable`
    3. `:start, :restart` or `:stop`

Resource: `service`

```
service("ngnix") do
 action [:enable, :start]
 retries 0
 retry_delay 2
 options --smallfiles
 service_name "ngnix"
end
```

Resource: `directory`

1. **Manages a directory on a node**
2. **Properties**
    1. `path` - defaults to name
    2. `mode` - rex as a string
    3. `owner` - directory owner
    4. `group` - directory group
    5. `recursive` - create or delete recursively
    6. `inherits` - rights from parent, on Windows
3. **Actions**
    1. `:nothing` - do nothing
    2. `:create` - creates directory (default)
    3. `:delete` - deletes directory

Resource: `directory`

```
directory '/etc/apache2' do
  owner 'root'
  group 'root'
  mode '0755'
  action :create
end
```

Other Popular Resources

1. apt_package, dmg_package, homebrew_package
2. bash, csh, ksh
3. file, cookbook_file, mount
4. cron, log, ohai
5. perl, python, ruby
6. git, subversion
7. template
8. windows_*

Illustrations

```
package %w{package-a package-b package-c package-d} do
  action :upgrade
end
```

Illustrations

```
package 'curl'
  case node[:platform]
  when 'redhat', 'centos'
    package 'zlib-devel'
    package 'openssl-devel'
    package 'libc6-dev'
  when 'ubuntu', 'debian'
    package 'openssl'
    package 'pkg-config'
     package 'subversion'
   end
end
```

Illustrations

```
if node['authorization']['sudo']['include_sudoers_d']
  directory '/etc/sudoers.d' do
    mode        '0755'
    owner       'root'
    group       'root'
    action      :create
  end

  cookbook_file '/etc/sudoers.d/README' do
    source      'README'
    mode        '0440'
    owner       'root'
    group       'root'
    action      :create
  end
end
```

Exercises

1. Develop a cook book to install MySql and create database schema on it
2. Develop a cookbook to install MongoDB on a docker on an ubuntu machine
3. Develop a cookbook to install Java Web Application stack on an ubuntu machine

Chef

The Eco System

Role

1. **Mechanism to group several nodes with similar cookbooks**
2. **Under chef-repo**
   - 2.1. `roles/web-servers.rb`
3. **Define Role**
   - 3.1. `name "web-servers"`
   - 3.2. `run_list "recipe[apache]"`
4. **Upload Role to the Server**
   - 4.1. `knife role from file web-servers.rb`
5. **Export the editor for Knife in knife.rb**
   - 5.1. `knife[:editor]="/usr/bin/vim"`
6. **Assign Role to a Node**

```
6.1.knife node edit rkt-webserver
6.2."run_list": [
6.3.  "recipe[starter::default]",
6.4.  “role[web-servers]"
6.5.]
```

Attributes

1. **Key-Value Settings**
2. **Sources**
   - 2.1. Nodes (collected by Ohai at the start of each Chef Infra Client run)
   - 2.2. Attribute files (in cookbooks)
   - 2.3. Recipes (in cookbooks)
   - 2.4. Environments
   - 2.5. Roles
3. **Types**
   - 3.1. `default`, `force_default`, `normal`, `override`, `force_override`, `automatic`

Attributes

1. **Automatic Attributes**
   1.1. `node['platform']`
   1.2. `node[‘ipaddress']`

2. **Defining Attributes in attributes/default.rb**
   2.1. `default['server']['port'] = [ '80' ]`

3. **Referring to attribute**
   3.1. `node['server']['port']`

Template

1. **Mechanism to generate files dynamically**
   1.1. rkt-cb/`templates/default/report.erb`
2. **Define Template**
   ```
   <%- 4.times do %>
   <%= @hi %>, <%= @world %> from <%= @from %>!
   <%- end %>
   ```
3. **Use the Template in recipe**
   ```
   template '/tmp/message' do
      source 'report.erb'
      variables(
         hi: 'Tesing',
         world: 'Welt',
         from: node['fqdn']
      )
   end
   ```

Exercises

1. Create a cookbook for cron based on configurable interval
2. Create an apache cookbook with configurable port
3. Develop html using a template
4. Prepare a role for web servers
5. Add cron and apache cookbooks to role
6. Prepare the infrastructure with the above role

Thank You