Chef

Module-3

Agenda

1. Search
2. Data bags
3. Notifications
4. LWRP
5. Ohai Plugin

Search

Search

1. **Indexes on the Chef server can be searched**
    1.1. **node, client, environment, role, <data_bag>**
2. **Chef uses Apache Solr for searching**
    2.1. **Use * for multiple characters**
    2.2. **Use `?` for single character**
3. **Command Line Syntax**
    3.1. `knife search <index> <solr-query>`
    3.2. **Use `-a` option for projection**
    3.3. **Use `OR` or `AND` for multiple queries**

Search: illustrations

1. **Searching node index on command line**
2. **List all nodes**
    2.1. `knife search node "*:*"`
3. **List all nodes with specified ip address**
    3.1. `knife search node "ip*:54.*"`
4. **List only the platform details of nodes**
    4.1. `knife search node "ip*:54.*" -a platform`
5. **List nodes on multiple clauses**
    5.1. `knife search node "ip*:53* or name:glarimy*"`
    5.2. `knife search node "ip*:53* and name:glarimy*"`

Search: illustrations

1. **Searching node index in recipe**

```
1.1.nodes = search("node", "*:*")

1.3.for node in nodes
1.3.1.   log node.to_s
1.3.2.   log node['ipaddress']
1.4.end
```

Data Bags

Data Bags

1. A global repository of attributes for all the nodes on the Chef-Server
2. A bag of similarly structured key-values data
3. Typically used for non-public data
4. Can also be encrypted
5. A mechanism for sharing data among nodes
6. Data is in JSON format

Data Bags

1. **Create data_bags folder**
    1.1. `chef-repo> mkdir data_bags`
2. **Create a data bag (users)**
    2.1. `chef-repo/data_bags> mkdir users`
3. **Create data in the data bag (krishna and vamsi)**
    3.1. `touch chef-repo/data_bags/users/krishna.json`
    3.2. `touch chef-repo/data_bags/users/vamsi.json`
4. **Export the data bag to the server**
    4.1. `knife data_bag create users`
5. **Export the data to the server**
    5.1. `knife data_bag from file users krishna.json`
    5.2. `knife data_bag from file users vamsi.json`
6. **Verify the data bag on the server**
    6.1. `knife search users "*:*:`

Data Bags: Illustration

1. **vamsi.json**

```
{
  "id": "vamsi",
  "comment": "Vamsi Kishore",
  "uid": 2001,
  "gid": 0,
  "home": "/home/vamsi",
  "shell": "/bin/bash"
}
```

2. **krishna.json**

```
{
  "id": "krishna",
  "comment": "Krishna Mohan Koyya",
  "uid": 2002,
  "gid": 0,
  "home": "/home/krishna",
  "shell": "/bin/bash"
}
```

Data Bags: Illustration

1. **Search databags in recipes**

```
search("users", "*:*").each do |user_data|
  #TODO
end
```

2. **Use the data**

```
search("users", "*:*").each do |user_data|
  user user_data["id"] do
    uid user_data["uid"]
    gid user_data["gid"]
    comment user_data["comment"]
    home user_data["home"]
    shell user_data["shell"]
  end
end
```

Notifications

Notifications

1. **A notification is a property on a resource**
2. **Resources take actions based on the notification**
   2.1. `notifies` **- informs other resources**
   2.2. `subscribes` **- listens to other resources**
3. **Syntax**
   3.1. `notifies :action, ‘resource[name]’, :timer`
   3.2. `subscribes :action, ‘resource[name]’, :timer`
   3.3. **Timers:**
      3.3.1. `:before` **- before the current resource**
      3.3.2. `:immediate` **- immediately**
      3.3.3. `:delayed` **- at the end of chef-client run (default)**

Notifications

```
execute 'first' do
  command 'echo hello'
  action :nothing
end

execute 'second' do
  command 'echo how are you?'
  action :nothing
end

execute 'third' do
  command 'echo see you later!'
  action :nothing
end

execute 'all' do
  command 'echo friend'
   notifies :run, 'execute[first]', :before
   notifies :run, 'execute[second]', :immediately
   notifies :run, 'execute[third]', :delayed
end
```

Notifications

```
execute 'third' do
  command 'echo see you later!'
  action :nothing
  subscribes :run, 'execute[second]', :immediately
end

execute 'second' do
  command 'echo how are you?'
  action :nothing
  subscribes :run, 'execute[first]', :immediately
end

execute 'first' do
  command 'echo hello'
end
```

LWRP

Light Weight Resource Providers

1. **A custom resource**
    1.1. Is a simple extension of Chef Infra Client
        1.1.1. that adds your own resources
    1.2. Is implemented and shipped as part of a cookbook
    1.3. Follows easy, repeatable syntax patterns
    1.4. Leverages existing resources and/or Ruby code
    1.5. Is reusable in the same way as other resources
2. **Chef Infra Client includes built-in resources**
    2.1. to manage files
    2.2. to manage packages
    2.3. to manage templates
    2.4. to manage services
3. **New resources can be added on top of them**

Light Weight Resource Providers

1. **A custom resource is defined as a Ruby file**
    1.1. Located in a cookbook's `/resources` directory
    1.2. Declares the properties of the custom resource
    1.3. Loads current state of properties
        1.3.1. if the resource already exists
    1.4. Defines each action the custom resource
2. **Name defaults to cookbook-name_resource-file-name**
    2.1. Can be overridden using `resource_name`
3. **Local properties can be accessed via `new_resource`**

Light Weight Resource Providers

1. **Create cookbook with a desired name**
   1.1. `chef-repo/cookbooks> chef generate cookbook lwrp`
2. **Create `/resources` folder under cookbook**
   2.1. `chef-repo/cookbooks/lwrp> mkdir resources`
3. **Create a file with desired name under `/resources`**
   3.1. `chef-repo/cookbooks/lwrp/resources> touch prints.rb`
4. **Develop the resource**
5. **Use the resource in a recipe**

Light Weight Resource Providers

1. **chef-repo/cookbooks/lwrp/resources/prints.rb**

```
1.1.resource_name :prints

1.3.actions :perform
1.4.attribute :message, kind_of: String
1.5.attribute :capitals, [true, false], default: true

1.7.action 'perform' do
    1.7.1.  log "doing the work”

    1.7.3.  if new_resource.capitals then
        1.7.3.1.    log "#{new_resource.message.upcase}"
    1.7.4.  else
        1.7.4.1.    log "#{new_resource.message.downcase}"
    1.7.5.  end

    1.8.1.  execute 'print' do
        1.8.1.1.    command "echo #{new_resource.message}"
    1.8.2.  end
1.9.end
```

Light Weight Resource Providers

1. **chef-repo/cookbooks/lwrp/recipes/default.rb**

```
1.1.log 'start'

1.3.prints 'Hello Krishna' do
  1.3.1.   message 'Hello World'
  1.3.2.   action :perform
1.4.end

1.6.log 'done'
```

Ohai

Ohai

1. **Runs on chef-client**
2. **Queries the current state of Node object**
   2.1. `chef-node> ohai`
3. **Plug-ins fetches different prices of data**
4. **Examples**
   4.1. network
   4.2. os
   4.3. hostname
   4.4. java
   4.5. nodes
   4.6. docker
   4.7. ec2

Ohai Plug-in: os

```
1.Ohai.plugin(:Java) do
2.  provides "languages/java"
3.  depends "languages"

5.  def get_java_info
    5.1.TODO
6.  end
7.  def has_real_java?
    7.1. TODO
8.  end

10.  def on_darwin?
    10.1.TODO
11.  end

13.  collect_data do
    13.1.TODO
14.  end
15.end
```

Ohai Plug-in: os

```
def get_java_info
   so = shell_out("java -mx64m -version")
   if so.exitstatus == 0
      java = Mash.new
      so.stderr.split(/\r?\n/).each do |line|
        case line
        when /(?:java|openjdk) version \"([0-9\.\_]+)\"/
          java[:version] = $1
       when /^(.+Runtime Environment.*) \((build)\s*(.+)\)$/
          java[:runtime] = { "name" => $1, "build" => $3 }
        when /^(.+ (Client|Server) VM) \(build\s*(.+)\)$/
          java[:hotspot] = { "name" => $1, "build" => $3 }
        end
      end

      languages[:java] = java unless java.empty?
  end
  rescue Ohai::Exceptions::Exec
    logger.trace('Failed "java -mx64m -version". Skipping plugin')
end
```

Ohai Plug-in: os

```
1.    def has_real_java?
2.      return true unless on_darwin?
3.
4.      shell_out("/usr/libexec/java_home").status.success?
5.    end
6.
7.    def on_darwin?
8.      RUBY_PLATFORM.downcase.include?("darwin")
9.    end
10.
11.   collect_data do
12.     get_java_info if has_real_java?
13.   end
```

Exercise

1. **Develop and use an LWRP to install MySql and initialise the database with employee data.**
    1.1. **Employee data comes from a data bag**
    1.2. **The LWRP provider and user must be two different cook books**
    1.3. **Constraints:**
        1.3.1. **LWRP Name: hr**
        1.3.2. **LWRP usage:**

```
1.3.2.1. hr “bengaluru” do
    1.3.2.1.1. source <data_bag name>
    1.3.2.1.2. pwd <mysql_init_password>
1.3.2.2. end
```

Thank You