LAB 4: Basic Commands of Chef

Recipe 1

Step 1: To generate the Recipe

```
cd test-cookbook
chef generate recipe recipe1
cd ..
vi test-cookbook/recipes/recipe1.rb
(Press i)
package 'finger' do
  action :install
end

file '/myfile2' do
  content "This is my second file"
  action :create
  owner 'root'
  group 'root'
end
```

Step 2: To verify the recipe

```
chef exec ruby -c test-cookbook/recipe/recipe1.rb
```

Step 3: To apply that recipe locally

We run chef-client to apply recipe to bring node into desired state, we call this process as "Convergence".

```
chef-client –zr " recipe[test-cookbook::recipe1]"
```

Step 4: Verification

```
ls /
which finger
```

Recipe 2

Step 1: To generate the Recipe

```
cd test-cookbook
chef generate recipe recipe2
cd ..
vi test-cookbook/recipes/recipe2.rb
(Press i)
package 'httpd' do
  action :install
end

file '/var/www/html/index.html' do
  content "Hello Dear Students!!!"
  action :create
end

service 'httpd' do
  action [:enable, :start]
end
(Press esc)
:wq!
```

Step 2: To verify the recipe

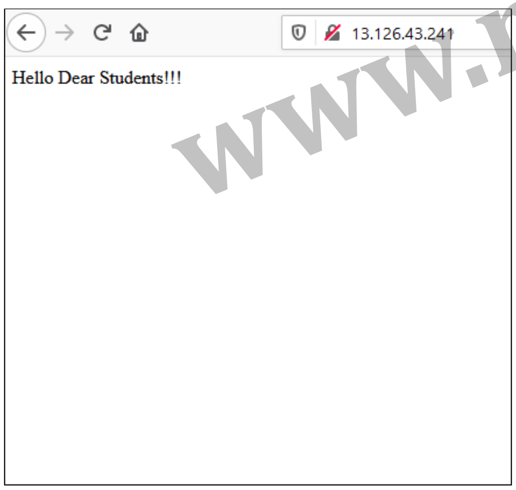
```
chef exec ruby -c test-cookbook/recipe/recipe2.rb
```

Step 3: To apply that recipe locally

```
chef-client –zr " recipe[test-cookbook::recipe2]"
```

Step 4: Verification

```
Paste Public IP in the Browser
```



Hello Dear Students!!!

Recipe 3

Step 1: To generate the Recipe

```
cd test-cookbook
chef generate recipe recipe3
cd ..
vi test-cookbook/recipes/recipe3.rb
(Press i)
package ‘httpd’ do
  action :install
end

file ‘/var/www/html/index.html’ do
  content “<h1>Hello Dear Students!!!</h1>
  <img src=‘rst.png’>”
  action :create
end

remote_file ‘/var/www/html/rst.png’ do
  source “https://rst-bucket-1.s3.amazonaws.com/WhatsApp+Image+2020-03-
13+at+22.12.27.jpeg”
end

service ‘httpd’ do
  action [:enable, :start]
end
(Press esc)
:wq!
```

Step 2: To verify the recipe

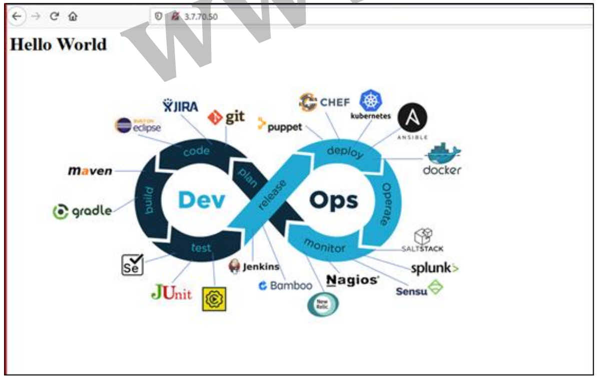
```
chef exec ruby -c test-cookbook/recipe/recipe3.rb
```

Step 3: To apply that recipe locally

```
chef-client –zr “ recipe[test-cookbook::recipe3]”
```

Step 4: Verification

```
Paste Public IP in the Browser
```



Recipe 4

We have a web application to be deployed into 1000 nodes & we need to know some details of each server. Because we need to mention that in configuration file of each node. This information is varied from system to system. These details we call “Attributes” Chef-client tool gathers these Attributes from ohai store and puts in configuration files, instead of hard coding these attributes, we mention as variables

Step 1: To generate the Recipe

```
cd test-cookbook
chef generate recipe recipe4
cd ..
vi test-cookbook/recipes/recipe4.rb
(Press i)
file ‘/robofile’ do
  content “This is to get Attributes
  HOSTNAME: #{node[‘hostname’]}
  IPADDRESS: #{node[‘ipaddress’]}
  CPU: #{node[‘cpu’][‘0’][‘mhz’]}
  MEMORY: #{node[‘memory’][‘total’]}”
  owner ‘root’
  group ‘root’
  action :create
end
(Press esc)
:wq!
```

Step 2: To verify the recipe

```
chef exec ruby -c test-cookbook/recipe/recipe3.rb
```

Step 3: To apply that recipe locally

```
chef-client –zr “ recipe[test-cookbook::recipe3]”
```

Step 4: Verification

```
ls /
cat /robofile
```

Recipe 5

To execute Linux commands

```
execute “run a script” do
  command <<-EOH
  mkdir /Panchanandir
  touch /Panchananfile
  EOH
end
```

Recipe 6

We can create users and groups

```
user “Ram” do
  action :create
end

group “devops” do
  action :create
  members ‘Ram’
  append true
end
```

Recipe 7

To deal with multiple resources at a time

```
[‘httpd’,’mariadb-server’,’unzip’,’git’,’vim’].each do |p|
  package p do
    action :install
  end
end

[‘Ram’,’Shyam’,’Ajay’,’Vijay’,’Rohan’,’Sohan’].each do |q|
  user q do
    action :create
  end
end
```