CHEF - Configuration(Each and every minute detail of machine like Server,Storage etc) Management (Like Delete,Update,Create etc..) Tool

Two Types of Configuration Management Tools
• Push Based : Server pushes configuration to the nodes(Ansible,Salt Stack)
• Pull Based : Nodes check with the server.Periodically and fetches the configuration from it.-(Chef,Puppet)

CHEF :

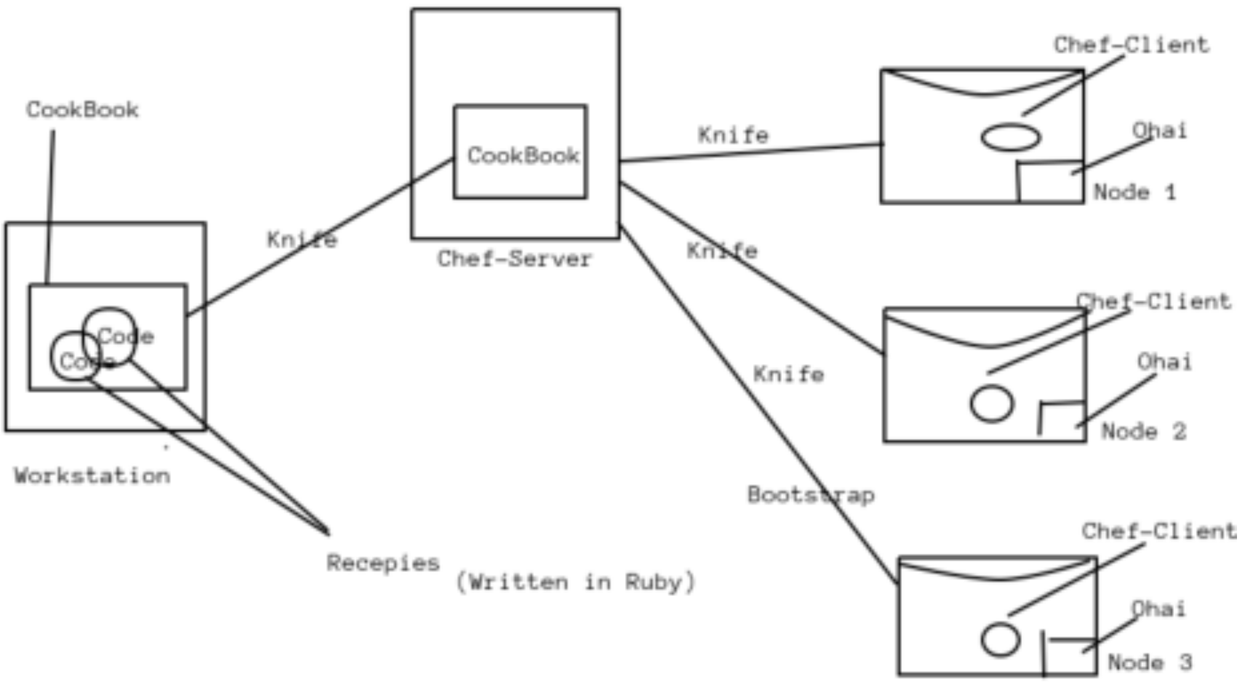
• Chef is a company and the name of a configuration management tool written in Ruby and Erlang
• Founded by Adam Jacobs in year 2009
• Actual name was “Marionette” later renamed to Chef
• On April 2,2019 the company announced that all their products are now open source under the Apache 2.0 License
• Chef is used by Facebook,AWS Opsworks,Hp Public Cloud etc
• Chef is a Admin. tool whatever system admins use to do manually,now we are automating all those tasks by using chef

Configuration Management : It is a method through which we automate admin tasks
• Config management tool turns our code into Infrastructure
• So our code would be repeatable testable and versionable

Advantages of CM Tools:

• Complete Automation
• Increase Uptime
• Improve Performance
• Ensure Compliance
• Prevent Errors
• Reduce Costs

Chef-Architecture or Process :



Components of Chef :

Workstation : Where we write code

- Workstations are personal computers or Virtual Servers where all configuration code is created,tested or changed
- Devops engineer actually sits here and write codes.This code is called recipe.A collection of recepies are called CookBooks
- Workstation communicate with the chef server using knife
- Knife is a command line tool that uploads the cookbook to the server

Chef-Server: Where we store code

- The chef-server is a middle-man between workstation and the nodes
- All cookbooks are stored here
- Server may be hosted locally or remote

Node : Where we apply code

- Nodes are the systems that require the configuration
- Ohai fetches the currents state of the node its located in
- Node communicates with the chef-server using the chef-client
- Each node can have a different configuartion required
- Chef-client is installed on every node

Knife : Tool to establish communication among workstation,server and node knife is a command-line tool that runs on workstation

Chef-Client : Tools runs on every chef node to pull code from chef server

• Chef client will
☐ gather current system configuration
☐ download the desired system configuration from the chef server
☐ Configure the node such that it adhere to the policy

Ohai : Maintain Current state information of the chef-node

Idempitency : Tracking the state of system resources to ensure that the changes should not reapply repeatedly

Chef-SuperMarket : Where we get custom Code

Creating Cookbooks and Recipes:

• First of all create one linux machine in AWS
• Now use putty and access the machine
☐ Login as -- ec2 user
☐ sudo su
☐ yum update -y
• Now go to google & search www.chef.io
• Go to downloads→ chef workstation
• Enter name,email,company -- It automatically starts downloading → go to Downloads & copy the URL
• Now in linux-machine
☐ wget <url>
☐ ls → it shows chef .rpm package
☐ yum install <chef-workstation> -y
☐ which chef
☐ chef --version

Cookbook : It is a collection of recipes and some other files and folders

Inside Cookbook :

chefignore : Like .gitignore
kitchen.yml : for testing cookbook
Metadata.rb : name,version,author etc of cookbook
readme.md : information about usgae of cookbook
recipe : where we write code
spec : for unit test
test : for integration test

Lab :

☐ which chef
☐ mkdir cookbooks
☐ ls
☐ cd cookbooks/
☐ chef generate cookbook test-cookbook
☐ yum install tree -y
☐ tree
☐ cd test-cookbook
☐ chef generate recipe <name>
☐ tree
☐ cd ..
☐ vi test-cookbook/recipes/test-recipe.rb

☐ add following code :
```
     File '/myfile' do
     Content 'Welcome to TG'
      action :create
       end
```
☐ chef spec ruby -c test-cookbook/recipes/test-recipe.rb
☐ chef-client -zr "recipe[test-cookbook:test-recipe]"
☐ ls /

Creating and Writing Second Recipe :

☐ cd test-cookbook
☐ chef generate recipe recipe2
☐ cd ..
☐ vi test-cookbook/recipes/recipe2.rb
☐ add following code :
```
     package 'tree' do
     action install
     end

      File '/myfile2' do
     Content 'second project code'
      action :create
      owner :root
      group:root
       end
```
☐ chef-client -zr "recipe[test-cookbook::recipe2]"
☐ cat /myfile2
☐ yum remove tree -y
☐ chef-client -zr "recipe[test-cookbook::recipe2]"

Deploying an Apache Server

☐ ls
☐ chef generate cookbook apache-cookbook
☐ ls
☐ cd apache-cookbook
☐ chef generate recipe apache-recipe
☐ tree
☐ cd ..
☐ ls
☐ vi apache-cookbook/recipes/apache-recipe.rb
☐ add following :
```
       package "httpd" do
       action install
       end

        File '/var/www/html/index.html' do
        content 'Welcome to TG'
        option : create
        end

        Service 'httpd' do
        action [:enable,:start]
        end
```

☐ chef exec ruby -c apache-cookbook/recipes/apache-recipe.rb
☐ chef-client -zr “recipe[apache-cookbook::apache-recipe]”

Resource : It is the basic components of a recipe used to manage the infrastructure with different kind of states.There can be multiple resources in a recipe,which will help infrastructure
Ex:
Package : Manages the package on a node
Service : Manages the services on a node
User : Manages the users on the node
Group : Manages groups
Template : Manages the files with embedded ruby template
Cookbook-File : Transfers the files from the files subdirectory in the cookbook to a location on the node
File : Manages the content of a file on the node
Execute : Executes a command on the node
Cron : Edits an existing cron file on the node
Directory : Manages the directory on the node

CHEF Attributes :

Attributes : It is a Key value pair which represents a specific details about a node

• Attributes are used by chef-client

• Attribute are used to determine :
☐ The current state of the node
☐ What the state of the node was at the end of the previous chef-client run
☐ What the state of the node should be at the end of the current chef-client

• Types of Attributes :
☐ Default
☐ Force_Default
☐ Normal
☐ Override
☐ Force_Override
☐ Automatic

• Attributes are defined by
☐ Node(Collected by ohai at the start of each chef-client run)
☐ Cookbooks(Attribute Files)
☐ Roles
☐ Environment

Note:Attributes defined by Ohai have the highest priority,followed by attributes defined in a recipe then attributes defined in an attribute files

Lab :

☐ login into Aws Linux Machine
☐ sudo su
☐ ls
☐ ll
☐ ohai

☐ ohai ipaddress
☐ ohai memory/total
☐ ohai cpu/0/mhz
☐ ls
☐ cd Cookbooks/
☐ ls
☐ cd apache-cookbook/
☐ tree
☐ chef generate recipe recipe-new
☐ cd ..
☐ vi apache-cookbook/recipes/recipes-new.rb
☐ add following :

```
File ‘/basicinfo’ do
Content "This is to get Attributes
HOSTNAME:  #{node['hostname']}
IPADDRESS:  #{node['ipadress']}
CPU:  #{node['cpu']['0']['mhz']}
MEMORY:  #{node['memory']['total']}"
owner ‘root’
group ‘root’
action  :'create
end
```

☐ chef-client -zr “ ”recipe[apache-cookbook::recipe3]"
☐ ls /
☐ cat /basicinfo

• Executing Linux Commands:

☐ Login to AWS Linux Machine
☐ sudo su
☐ cd cookbooks
☐ ls
☐ vi test-cookbook/recipes/test-recipes.rb
☐ add following

```
 execute "run a script" do
    command <<-EOH
    mkdir /rajputdir
    touch /rajputfile
    EOH
  end
```

☐ chef exec ruby -c test cookbook/recippes/test-recipe.rb
☐ chef-client -zr “recipe[test-cookbook::test-recipe]”
☐ ls /
☐ vi test-cookbook/recipes/test-recipe.rb
☐ add following

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

☐ chef-client -zr “recipe[test-cookbook::test-recipe]”
☐ vi test-cookbook/recipes/test-recipe.rb
☐ add following

```
group “technicalguftugu” do
action :create
members ‘rajput’
append true
```

        end
☐ chef-client -zr “recipe[test-cookbook::test-recipe]”
☐ cat /etc/group

• We run chef client to apply recipe to bring node into desired state.This process is known as Convergence

Runlist :

• To run the recipes in a sequence order that we mention in a run list
• With this process,we can run multiple recipes,but the condition is,there must be only one recipe from one cookbook
☐ chef-client -zr “recipe[test-cookbook::test-recipe],recipe[apache-cookbook::apache-recipe]”

How to Include Recipes:
• To call recipe/recipes from another recipe with in same cookbook
• To run multiple recipes from same cookbook
• Here comes the default recipe into action(we can use any recipe)
• We can run any no. of recipes with this command,but all must be from same cookbook
☐ vi test-cookbook/recipes/default.rb
☐ add following
      include-recipe “test-cookbook::test-recipe”
      include-recipe “test-cookbook::recipe2”
☐ chef-client -zr “recipe[test-cookbook::default]"

• Combining previous 2 concepts to run multiple recipes from multiple cookbooks simultaneously

☐ chef-client -zr “recipe[test-cookbook::default],recipe[apache-cookbook::default]”
                    (OR)
☐ chef-client -zr “recipe[test-cookbook],recipe[apache-cookbook]”

Chef Server and Node:

• Chef server is going to be a mediator for the code or cookbooks
• Firstly create one account in chef-server
• Then,attach our workstation to the chef-server
• Now upload cookbooks from workstation to chef server
• Now attach nodes to chef server via bootstrap process
• Apply cookbooks from chef server to Node

Lab :
☐ Login to amazon linux machine using putty
☐ ls
☐ cd cookbooks/
☐ ls

☐ Open google chrome→ search manage.chef.io
☐ Create one account
☐ Go to chef account → click on organisation→ starter kit→ download starter kit
☐ Open the downloaded content → unzip → chef-repo
☐ Now download winscp → login with ec2 credentials

☐ Now drag & drop chef folder from window to linux

☐ Now open workstation in AWS again
☐ ls
☐ cd ..
☐ ls
☐ cd chef-repo/
☐ ls -a
☐ cd .chef/
☐ ls
☐ cat config.rb
☐ cd ..
☐ knife ssl check

Bootstrap a Node:
• Attaching a node to chef server is called Bootstrapping(Both workstation and node should be in same az)
• Now onwards,we have to be inside chef-repo directory to run any command
• Two actions will be done while bootstrapping:
☐ Adding node to chef-server
☐ Installing chef package

• Create one linux machine(node 1) ,launch in same AZ
• Advance Details
```
#!/bin/bash
sudo su
yum update -y
```
• Now go to chef-workstation
• cd chef-repo
• Paste node-key.pem in chef-repo folder from local pc
• knife bootstrap <private ip of node> --ssh-user ec2-user --sudo -i node-key.pem -N node1
• knife node list

• under chef-repo do ls
• cd ..
• ls
• mv cookbook/test-cookbooks chef-repo/cookbooks
• mv cookbook/apache-cookbooks chef-repo/cookbooks
• rm -rf cookbooks/ → inside ec2-user
• cd chef-repo
• ls
• ls cookbooks/

Uploading apache-cookbook into chef-server:

☐ knife cookbook upload apache-ccokbook
☐ knife cookbook list
☐ knife node run_list set node1 “recipe[apache-cookbook::apache-recipe]”
☐ knife node show node1

Accessing Node1:

☐ sudo su
☐ chef-client

Inside chef-repo :

☐ vi cookbooks/apache-cookbbok/recipes/apache-recipe.rb → change some content and save
☐ knife cookbook upload apache-cookbook
☐ Now run chef-client command inside node1 again

Automating chef-client
☐ inside node1
☐ vi /etc/crontab
☐ add following:
    * * * * * root chef-client
• Now in chef-repo
☐vi cookbooks/apache-cookbooks/recipes/apache-recipe.rb
☐ knife cookbook upload apache-ccokbook
☐ Open browser and check updated content

Managing Multiple Nodes:
• Now,create another linux machine(node2)
• Advance details
```
 !#/bin/bash
 sudo su
 yum update -y
 echo “* * * * * root chef-client” >> /etc/crontab
```
• Now in workstation run bootstrap command
• Now attach cookbook to node run list
• Now in browser check node 2 webpage

Commands to delete and clean chef-server:

☐ knife cookbook list
☐ knife cookbook delete <cookbook name> -y
☐ knife node list
☐ knife node delete <node name> -y
☐ knife client list
☐ knife client delete <client name> -y
☐ knife role list
☐ knife role delete <role name> -y

Lab :
• In Workstation
☐ sudo su
☐ ls
☐ cd chef-repo
☐ knife node list
☐ knife node delete node1 -y
☐ knife node delete node2 -y
☐ knife node list
☐ knife cookbook list
☐ knife cookbook delete apache-cookbook -y
☐ knife client list
☐ knife client delete node1 -y
☐ knife client delete node2 -y

Chef-role:

☐ inside chef-repo do ls
☐ cd roles/
☐ ls
☐ vi devops.rb
☐ add following :

```
 name “devops”
 description “web server role”
  run_list “recipe[apache-cookbook::apache-recipe]”
```

☐ now inside chef-repo
☐ knife role from file roles/devops.rb
☐ knife role list

• Now create 2 instances as node1 and node2 in same az as of workstation
☐ knife bootstrap <private ip of node> --ssh-user ec2-user --sudo -i node-key.pem -N node1
☐ knife bootstrap <private ip of node> --ssh-user ec2-user --sudo -i node-key.pem -N node2
• Now connect these nodes to role
☐ knife node list
☐ knife node run_list set node1 “role[devops]”
☐ knife node run_list set node2 “role[devops]”
☐ knife node show node1
☐ knife cookbook upload apache-cookbook
• check with public ip of any node
☐ vi cookbooks/apache-cookbook/recipes/apache-recipe.rb
☐ edit content and save
☐ knife cookbook upload apache-cookbook
☐ cat cookbooks/apache-cookbook/recipes/-recipe3.rb
☐ vi roles/devops.rb
☐ add following :

```
  name “devops”
  description “web server role”
  run_list “recipe[apache-cookbook::recipe3]”
```

☐ knife role from file roles/devops.rb

• Now access any node via putty & check

• Again in workstation
☐ vi roles/devops.rb
☐ add following :

```
  name “devops”
  description “web server role”
  run_list “recipe[apache-cookbook]”
```

☐ knife role from file roles/devops.rb
☐ vi cookbooks/apache-cookbook/recipes/-recipe3.rb
☐ add follwoing

```
user “bhupinder”
file"/bhupinderfile
```

☐ vi cookbooks/apache-cookbook/recipes/apache-recipe.rb
☐ edit content and save
☐ knife cookbook upload apache-cookbook
☐ vi roles/devops.rb
☐ add following :

```
  name “devops”
  description “web server role”
  run_list “recipe[apache-cookbook],recipe[test-cookbook]”
```

☐ knife role from file roles/devops.rb
☐ knife cookbook upload test-cookbook

• inside chef-repo
☐ vi cookbooks/test-cookbook/recipes/test-recipe.rb
☐ add following:
```
    %w(httpd mariadb-server unzip git vim) .each do |p|
     package p do
     action :install
      end
      end
```
☐ knife cookbook upload test-cookbook
• Now inside any node search git or vim(which git or which vim) etc to check it is working properly

~~~~~~
~~~~~~