CHEF

History:

- Chef was created by Adam Jacob as a tool for his consulting company, whose business model was to build end-to-end server/deployment tools.
- Chef is a configuration management tool for dealing with machine setup on physical servers, virtual machines and in the cloud.
- Prior to 2013, it was called as “Opscode”, and renamed as Chef in 2013.

Advantages of CM tool, Advantages of chef:

Advantages of CM tool:

- Increased efficiency with a defined configuration process that provides control and improves visibility with tracking.
- Cost reduction by having detailed knowledge of all the elements of the configuration which allows for unnecessary duplication to be avoided.
- Faster restoration of your service if a process failure occurs. If you know the required state of the configuration then recovering the working configuration will be much quicker and easier.
- More efficient change management that reduces the risk of product incompatibility or problems.
- Your business will have greater agility and faster problem resolution, giving a better quality of service for your customers.

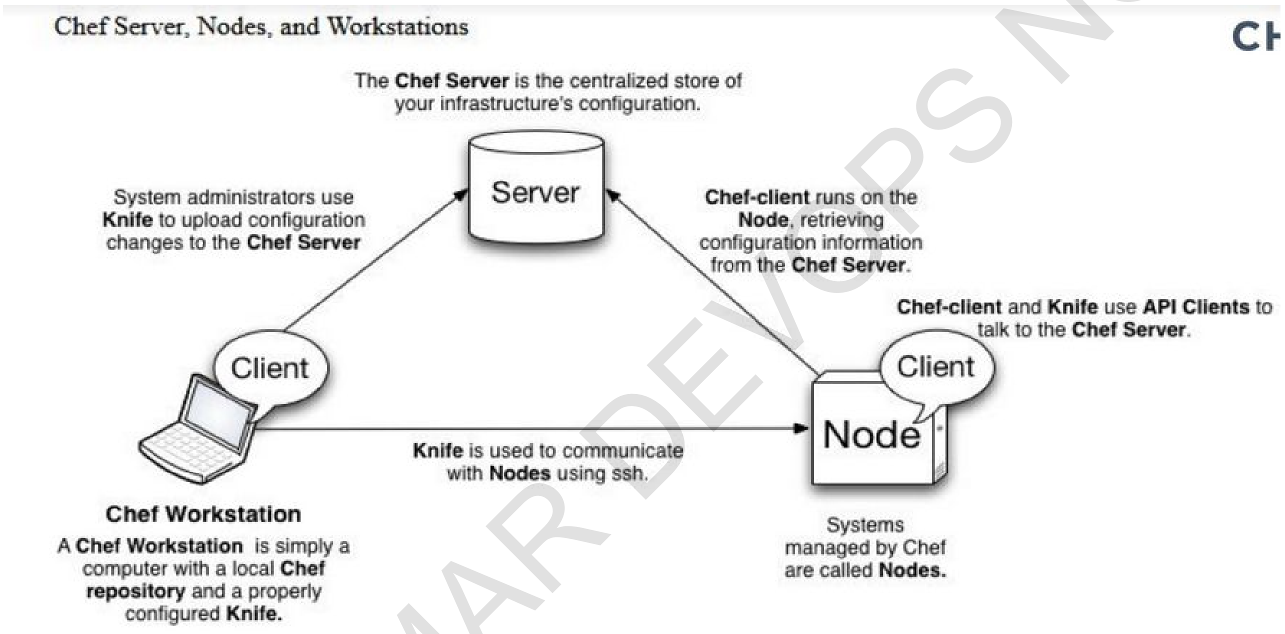
Advantages of chef:

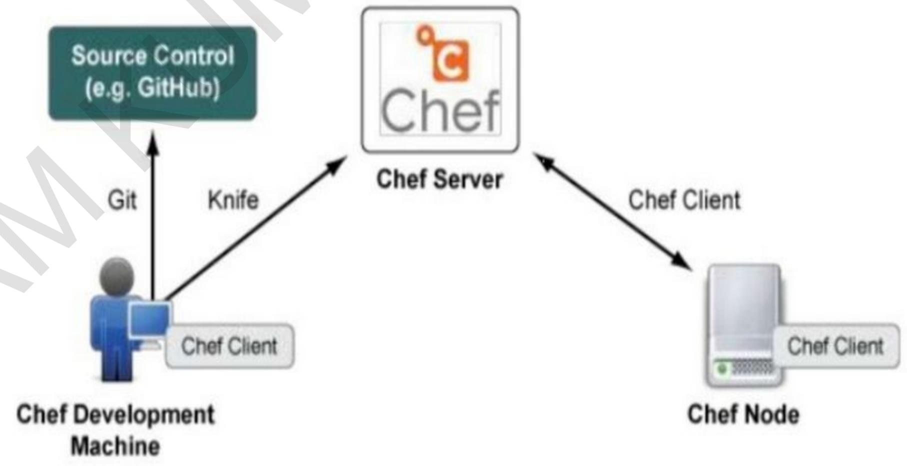
- Chef is a powerful configuration management tool that manages and helps in automating the infrastructure with the advanced software methodology .
- It helps in managing the timeline for release. It really helps in deploying the project with the automation.
- Extendable with scripting language, easy configuration files
- Easy to implement, learn and use on a daily basis for Configuration Management.

- chef offers a CLI and GUI to report on fleets of deployed assets and configurations assigned to them.
- good to register nodes, then install cookbooks from supermarkets. ready to use cookbooks simply deploy them on registered nodes.

Chef Architecture, Chef Workstation:

Chef Architecture:


Chef Server, Nodes, and Workstations
The Chef Server is the centralized store of your infrastructure's configuration.
Server
System administrators use Knife to upload configuration changes to the Chef Server
Chef-client runs on the Node, retrieving configuration information from the Chef Server.
Chef-client and Knife use API Clients to talk to the Chef Server.
Client
Client
Node
Knife is used to communicate with Nodes using ssh.
Chef Workstation
A Chef Workstation is simply a computer with a local Chef repository and a properly configured Knife.
Systems managed by Chef are called Nodes.



<u>Chef Workstation:</u>

CHEF WORKSTATION SETUP ON RHEL

1. Make sure chef workstation name is changed to required hostname

```
#hostnamectl set-hostname <new_hostname>
```

2. Download the Chef Development kit from Chef Official website

3. Verify the package that was downloaded using

```
# rpm -ivh --test <download_chefdk package>
```

if verify is ok, then go ahead and install it.

```
# rpm -ivh <download_chefdk package>
```

4. Execute below command :

```
# chef verify
```

5. Execute below command :

```
# which ruby
```

6. setup ruby path variables

```
# echo 'eval "$(chef shell-init bash)"' >> ~/.bash_profile
# . ~/.bash_profile
# which ruby
```

7. Install git on chef workstation, before creating chef-repo

```
# yum install -y git-all
```

8. Create a chef-repo in user's home directory path

```
# chef generate repo chef-repo
```

9. verify the components inside of chef-repo

```
# ls -al ~/chef-repo/
```

10. Now, configure git global settings

```
# git config --global user.name "user_name"

# git config --global user.email "user_emailaddress"
```

11. enter into chef-repo that was created

```
# cd ~/chef-repo/
```

12. now, initiate the git init inside of chef-repo folder to start tracking

```
# git init
```

13. Now, create a hidden directory called ".chef" under the chef-repo directory.

● This hidden directory will hold the RSA keys that we created on the Chef server.

```
# mkdir -p ~/chef-repo/.chef
```

14. Since this hidden directory stores the RSA keys, it should not be exposed to the public.

● To do that we will add this directory to ".gitignore" to prevent uploading the contents to GitHub.

```
# echo '.chef' >> ~/chef-repo/.gitignore
```

15. Now, Add and commit all existing files.

```
# cd ~/chef-repo/   # git add .    # git commit -m "initial commit" .
```

Now, check status

```
# git status
```

<u>Chef Server, Chef Nodes:</u>

<u>Chef server setup on RHEL:</u>

1. Make sure chef workstation name is changed to required hostname

```
#hostnamectl set-hostname <new_hostname>
```

2. Download the Server Package from Chef Official website

3. Verify the package that was downloaded using

```
    # rpm -ivh --test <downloaded_server_ package>
```

if verify is ok, then go ahead and install it.

```
# rpm -ivh <downloaded_server_ package>
```

4. Execute below command to reconfigure

```
    # chef-server-ctl reconfigure
```

5. Execute below command to check status:

```
    # chef-server-ctl status
```

6. Create a Chef user use below command

```
  # chef-server-ctl user-create <user_name> <user__firstname> <user_lastname>
<email> password -f <path>
```

Example -

```
# chef-server-ctl user-create admin vikram kumar thumuvikram@gmail.com
password -f /etc/chef/admin.pem
```

7. Create organisation in chef

```
# chef-server-ctl org-create <org_shortname> <org_fullname> --assocation_user
<user_name> -f <path/org_shortname-validator.pem>
```

Example

```
chef-server-ctl org-create vkt "vkt technologies" --association_user admin -f
/etc/chef/vkt-validator.pem
```

8. Firewall setup for port 80 and 443

```
 # firewall-cmd --permanent --zone public --add-service http

 # firewall-cmd --permanent --zone public --add-service https

 # firewall-cmd --reload
```

<u>Setup Chef Node:</u>

● Login into chef node

● Create user “devops” using below command

```
# useradd devops
```

● Make sure that user id devops have sudo access. We can provide sudo access to devops using below command

```
# usermod -G wheel devops
```

● Install chef client package on chef node

BOOTSTRAP CHEF NODE

● Perform below steps from chef workstation
```
# knife bootstrap <IP address of Chef node>  -x root -P pass --sudo
```
● Verify bootstrap using below command
```
# knife node list
```
● Verify by using listing node details
```
# knife client show <node_name>
```

Knife:

POST SETUP ACTIONS ON CHEF-WORKSTATION

● We need to create a knife.rb file on chef-workstation after <username>.pem and <short_org_name> transferred to chef-workstation from chef-server

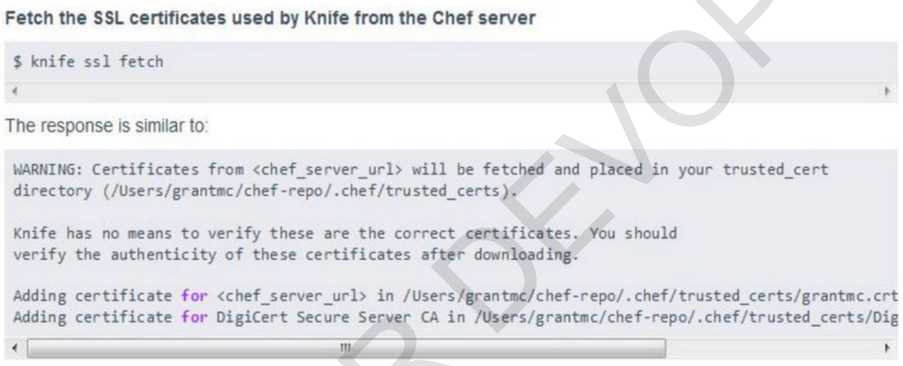
STEPS:

1. Login into chef-workstation
2. Browse to location ~/chef-repo/.chef
3. Make sure that knife.rb file is created with code

```
current_dir = File.dirname(__FILE__)
log_level                :info
log_location             STDOUT
node_name                 "<user_name>"
client_key               "#{current_dir}/<user_name>.pem"
Validation_client_name    "<short_org_name>-validator"
validation_key           "#{current_dir}/<short_org_name>-validator.pem"
chef_server_url
"https://<chef_server_host_name>/organizations/<short_org_name>"
syntax_check_cache_path  "#{ENV['HOME']}/.chef/syntaxcache"
```

cookbook_path ["#{current_dir}/../cookbooks"]
● Now, save the file knife.rb

KNIFE SSL FETCH
● PURPOSE - Use the knife ssl fetch subcommand to copy SSL certificates from an HTTPS server to the trusted_certs_dir directory that is used by knife and the chef-client to store trusted SSL certificates. When these certificates match the hostname of the remote server, running knife ssl fetch is the only step required to verify a remote server that is accessed by either knife or the chef-client.
● Once, knife.rb is created go ahead and execute below command On chef-workstation
# knife ssl fetch
Note : make sure that before executing above command pwd path should be pointing to .chef folder on workstation

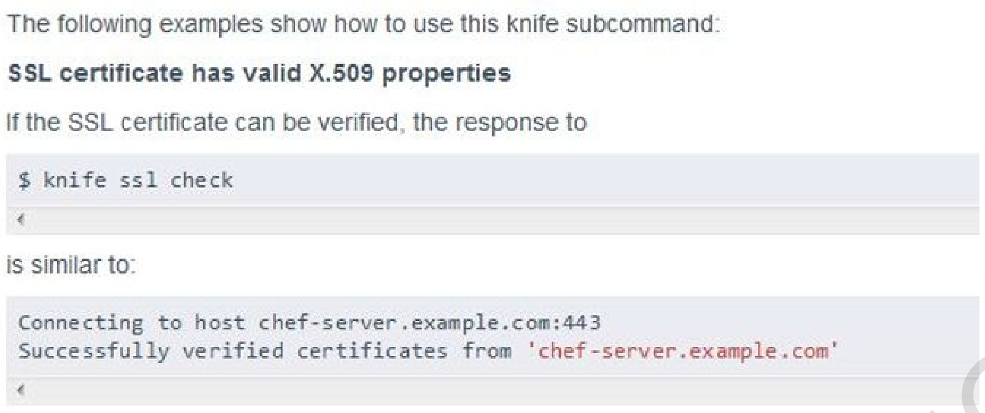

Fetch the SSL certificates used by Knife from the Chef server

```
$ knife ssl fetch
```

The response is similar to:

```
WARNING: Certificates from <chef_server_url> will be fetched and placed in your trusted_cert
directory (/Users/grantmc/chef-repo/.chef/trusted_certs).

Knife has no means to verify these are the correct certificates. You should
verify the authenticity of these certificates after downloading.

Adding certificate for <chef_server_url> in /Users/grantmc/chef-repo/.chef/trusted_certs/grantmc.crt
Adding certificate for DigiCert Secure Server CA in /Users/grantmc/chef-repo/.chef/trusted_certs/Dig
```

KNIFE SSL CHECK
● Use the knife ssl check subcommand to verify the SSL configuration for the Chef server or a location specified by a URL or URI. Invalid certificates will not be used by OpenSSL.
● When this command is run, the certificate files (*.crt and/or *.pem) that are located in the /.chef/trusted_certs directory are checked to see if they have valid X.509 certificate properties. A warning is returned when certificates do not have valid X.509 certificate properties or if the /.chef/trusted_certs directory does not contain any certificates.
● Command
# knife ssl check


The following examples show how to use this knife subcommand:

SSL certificate has valid X.509 properties

If the SSL certificate can be verified, the response to

```
$ knife ssl check
```

is similar to:

```
Connecting to host chef-server.example.com:443
Successfully verified certificates from 'chef-server.example.com'
```

Chef-client:

A 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, including:

- Registering and authenticating the node with the Chef server
- Building the node object
- Synchronizing cookbooks
- Compiling the resource collection by loading each of the required cookbooks, including recipes, attributes, and all other dependencies
- Taking the appropriate and required actions to configure the node
- Looking for exceptions and notifications, handling each as required

Ohai:

Ohai is a tool that is used to detect attributes on a node, and then provide these attributes to the chef-client at the start of every chef-client run

● OHAI comes with Chef Development Kit

● We can run OHAI commands on any node where chef-client is installed and ohai gets installed as part of chef client installation

● OHAI will return data inform of JSON (JavaScript Object Notation)

● OHAI gathers node information from System Hardware Inventory

● OHAI Runs when we use either chef-client or Client-apply commands

OHAI COMMANDS

```
# ohai

# ohai <key_pair>
```

Example : #ohai ipaddress

```
#ohai hostname

#ohai memory

#ohai memory/total
```

Idempotency:

- It means a recipe can run multiple times on same system and results will be identical.
- It brings a chef-node to desired state as per IAAC

Cookbooks:

● It is the fundamental unit of configuration and policy distribution

● Each cookbook defines a scenario

● Contains all of the components that are required to complete or support a scenario

Example :

● Everything that is needed to configure and install SQL etc

● For more information on Cookbooks, please visit below link

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

CHEF HELP

● To know more about chef commands, use below command that shows help information

# chef --help

● We can also use --help option on sub commands

# chef generate --help

● Below command shows help information on chef cookbook

# chef generate cookbook --help

HOW TO CREATE A COOKBOOK IN CHEF

STEP 1 : Make sure that we are located in chef-repo

STEP 2 : cd into cookbooks folder in chef-repo

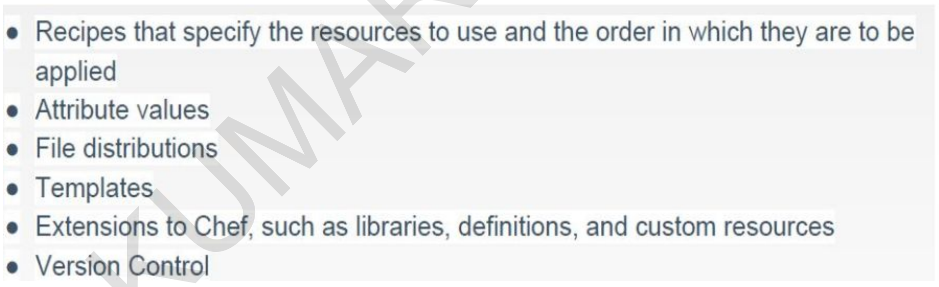
#cd cookbooks

STEP 3: Go ahead and create a cookbook using below command

# chef generate cookbook <cookbook_name>

Example : # chef generate cookbook workstation

COOKBOOK - COMPONENTS -



- Recipes that specify the resources to use and the order in which they are to be applied
- Attribute values
- File distributions
- Templates
- Extensions to Chef, such as libraries, definitions, and custom resources
- Version Control

COOKBOOK - COMMON COMPONENTS

- README
- Metadata
- Recipes
- Testing directories (spec + test)

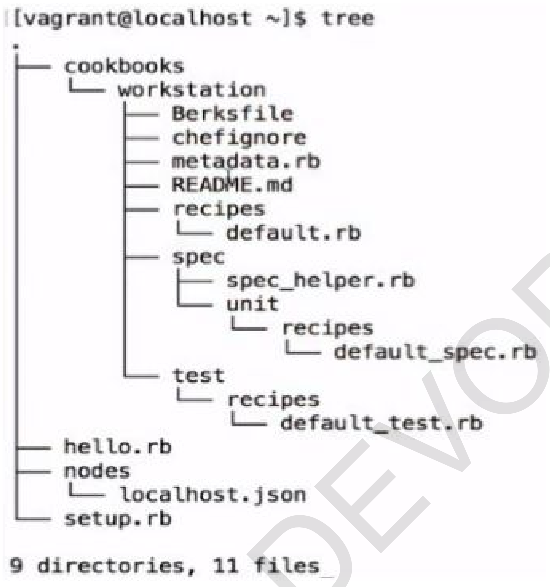
COOKBOOK COMPONENTS

● Berksfile - Manages dependencies

● Chefignore - similar to git ignore, used to ignore files And directories from chef server

● Metadata.rb files contains common information of Cookbook like maintainer, Versioner, description, long descriptions etc

● README.md files explains about how cookbook works



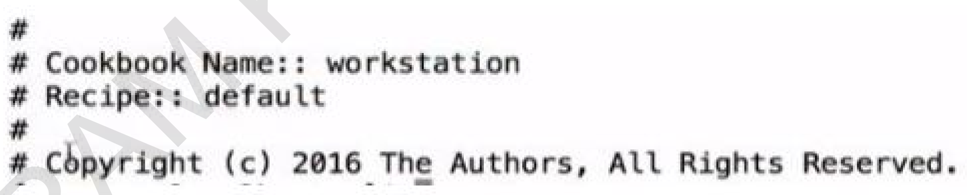
```
[vagrant@localhost ~]$ tree
.
├── cookbooks
│   └── workstation
│       ├── Berksfile
│       ├── chefignore
│       ├── metadata.rb
│       ├── README.md
│       ├── recipes
│       │   └── default.rb
│       ├── spec
│       │   ├── spec_helper.rb
│       │   └── unit
│       │       └── recipes
│       │           └── default_spec.rb
│       └── test
│           └── recipes
│               └── default_test.rb
├── hello.rb
├── nodes
│   └── localhost.json
└── setup.rb

9 directories, 11 files
```

COOKBOOK COMPONENT - RECIPES

● Cookbook generates a default.rb file under recipes folder

● All cookbook recipes files must be defined inside a Recipes folder

● Below is the information of default.rb file



```
#
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2016 The Authors, All Rights Reserved.
```

CHEF COOKBOOK UNIT TESTING

● Spec folder inside of cookbook is for chef cookbook unit testing

● Unit testing is done using rspec utility of chef

CHEF COOKBOOK INTEGRATION TESTING

● "Test" folder inside of chef cookbook for integration testing

● Integration testing is done using a framework called as "Test Kitchen"

NOTE - Generally, test and spec folders are included in chefignore file

TRACK CHANGES IN CHEF COOKBOOK USING GIT

● We need to track chef cookbooks in a single alone units , it means apply git on individual cookbooks

Step 1 : Install git on Chef Workstation PC

Step 2 : go into target Chef cookbook folder

Step 3 : then # git init

This will initiate the tracking of cookbook

Step 4 : Configure git user.name and user.email on the cookbook using git git config command

PROCEDURES ON MAJOR OR MINOR CHANGES IN COOKBOOK

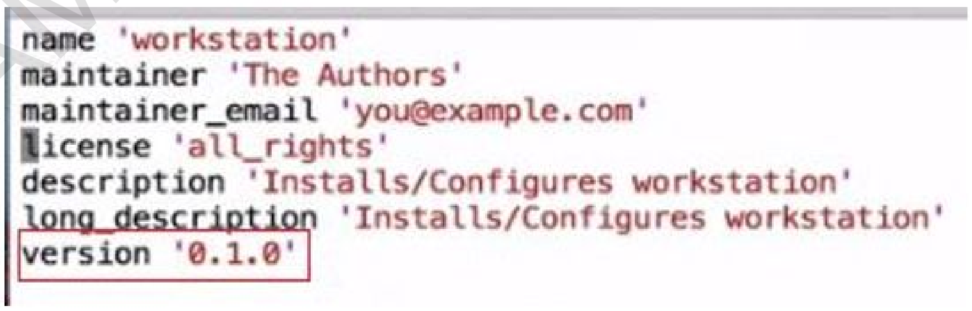
● Any Changes in Cookbook either major or minor, then need to update metadata.rb file

● Open Metadata.rb file and update its Semantic versioning information

● Semantic version format is

Major.minor.patch version

Example : 1.0.5



```
name 'workstation'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'all_rights'
description 'Installs/Configures workstation'
long_description 'Installs/Configures workstation'
version '0.1.0'
```

Recipes:

- A Chef cookbook consists of recipes that a node's desired state.
- Recipes work as a collection of resources that determine the configuration or policy of a node, with resources being a configuration element of the recipe.
- For a node to run a recipe, it must be on that node's run list.

Q. How to apply chef-client single recipe in local mode?

chef-client --local-mode RECIPE_FILE

Berks file:

- The goal of Berkshelf is to manage a cookbook outside of Chef repository in isolation.
- Berkshelf is a Command Line tool which can act as a source code management tool or a package manager like gem, apt, yum, etc.
- It replaces portions of knife like generating, uploading & downloading Cookbooks.
- A Berksfile describes the set of sources and dependencies needed to use a cookbook. It is used in conjunction with the berks command.
- Berkshelf reads the dependencies from metadata as well, as long as you add the metadata line to the Berksfile.

Metadata:

- Every cookbook requires a small amount of metadata.
- Located at the top level of a cookbook's directory structure
- Compiled whenever a cookbook is uploaded to the Chef server or when the knife cookbook metadata subcommand is run, and then stored as JSON data
- Created automatically by knife whenever the knife cookbook create subcommand is run
- Edited using a text editor, and then re-uploaded to the Chef server as part of a cookbook upload

Ruby language:

Ruby is an interpreted, high-level, general-purpose programming language.

It was designed and developed in the mid-1990s by Yukihiro "Matz" Matsumoto in Japan.

Ruby is dynamically typed and uses garbage collection.

It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

According to the creator, Ruby was influenced by Perl, Smalltalk, Eiffel, Ada, Basic, and Lisp.

Deploying Apache web server:

- The Apache web server is easy to install. On my CentOS 6.x server, it just takes a simple yum command.
- It installs all the necessary dependencies if any are missing.
- I used the dnf command below on one of my Fedora virtual machines.
- The syntax for dnf and yum are the same except for the name of the command itself.

```
dnf -y install httpd
```

- All the configuration files for Apache are located in /etc/httpd/conf and /etc/httpd/conf.d. The data for the websites is located in /var/www by default, but you can change that if you want.

Note:

I use a virtual machine (VM) using Fedora 27 with Apache 2.4.29. If you have a different distribution or a different release of Fedora, your commands and the locations and content of the configuration files may be different. However, the configuration lines you need to modify are the same.

Run list:

- In local mode, we need to provide a list of recipes to apply to the system.This is called a run list.
- A run list is an ordered collection of recipes to execute, each recipe in the run list must be addressed with the format

recipe[COOKBOOK::RECIPE]

Q. How to apply chef-client using runlist on Multiple cookbook ?

● Syntax :

chef-client --local-mode --runlist

“recipe[cookbook_name::recipe_name],recipe[cookbook_name::recipe_nam

E],....n”

Note :

● We can also use -z instead of --local-mode

● We can also use -r instead of --runlist

Include_recipe:

Q. How to include methods in chef recipes ?

● Include method in recipes would help to call another recipe in a recipe

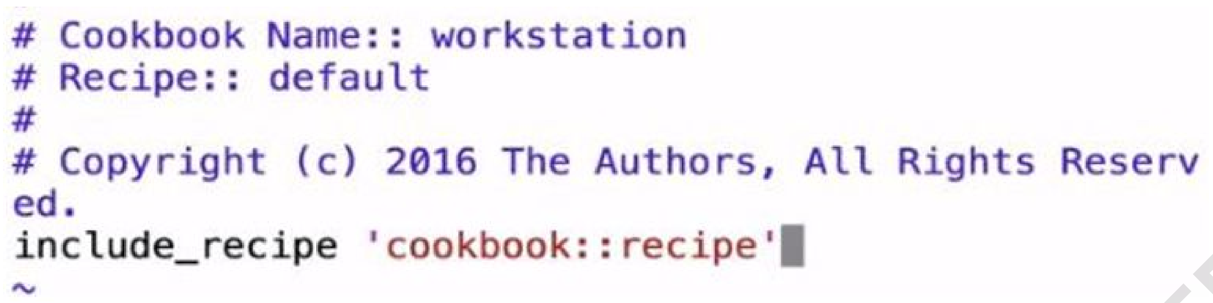
● Generally, include method should be invoked in default.rb file of a cookbook

● Any cookbook would contain default.rb recipe file under recipe folder of a cookbook

● Syntax is :

Include_recipe ‘cookbook_name::recipe’

Sample case study of include_recipe:



```
# Cookbook Name:: workstation
# Recipe:: default
#
# Copyright (c) 2016 The Authors, All Rights Reserv
ed.
include_recipe 'cookbook::recipe'
~
```

Attributes:

- Attributes define specific values about a node and its configuration and are used by the Chef client to apply those attributes to nodes via its attribute list.
- The chef client can receive attributes from nodes, attribute files, recipes, environments, and roles.
- Often attributes are used in conjunction with templates and recipes to define settings.

An attribute is a specific detail about a node. Attributes are used by Chef Infra Client to understand:

- The current state of the node
- What the state of the node was at the end of the previous Chef Infra Client run
- What the state of the node should be at the end of the current Chef Infra Client run

Chef Resources:

- Resources are written in Ruby and defined in recipe files.
- Resources must contain a type, a name, one or more properties, and one or more actions.
- Resources are the key components that make up any single recipe.

resource is a statement of configuration policy that:

- Describes the desired state for a configuration item
- Declares the steps needed to bring that item to the desired state
- Specifies a resource type—such as package, template, or service
- Lists additional details (also known as resource properties), as necessary

- Are grouped into recipes, which describe working configurations

Example: Package

```
Package 'httpd' do
    Action :install
end
```

The package name 'httpd' is installed

Example: Service

```
Service 'ntp' do
   Action [ :enable, :start]
end
```

The service named 'ntp' is enable and started

Example: File

```
File '/etc/motd' do
    Content 'this computer is the property ...'
end
```

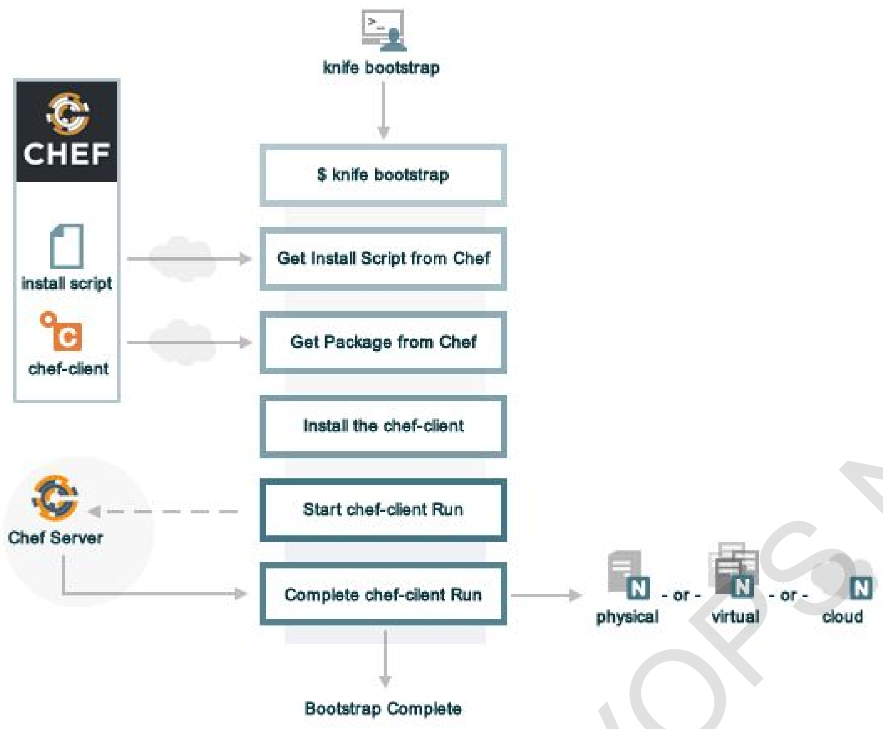
<u>Bootstrapping node:</u>

- Bootstrapping installs Chef Infra Client on a target system so that it can run as a client and sets the node up to communicate with a Chef Infra Server.

There are two ways to do this:

- Run the knife bootstrap command from a workstation.
- Perform an unattended install to bootstrap from the node itself, without requiring SSH or WinRM connectivity.

The following diagram shows the stages of the bootstrap operation



Wrapper cookbook:

- A wrapper cookbook is just a regular cookbook that includes recipes from other cookbooks.

Common use cases for wrapper cookbooks include:

- Modifying the behavior of a community cookbook from the Chef Supermarket
- Bundling several base cookbooks into a single cookbook
- Version controlling a node's run list and attribute definitions

To include another cookbook in your wrapper cookbook you must do a minimum of two things:

- Add dependencies to your wrapper cookbook's metadata.rb
- Add an include_recipe line to your wrapper cookbook's recipes/default.rb

Chef supermarket:

- Chef Supermarket is the site for community cookbooks. It provides an easily searchable cookbook repository and a friendly web UI.

- Cookbooks that are part of the Chef Supermarket are accessible by any Chef user.

There are two ways to use Chef Supermarket:

- The public Chef Supermarket is hosted by Chef Software and is located at Chef Supermarket.
- A private Chef Supermarket may be installed on-premise behind the firewall on the internal network. Cookbook retrieval from a private Chef Supermarket is often faster than from the public Chef Supermarket because of closer proximity and fewer cookbooks to resolve. A private Chef Supermarket can also help formalize internal cookbook release management processes (e.g. “a cookbook is not released until it’s published on the private Chef Supermarket”).

Calling Dependency cookbooks:

- Berkshelf is a dependency manager for Chef cookbooks.
- It's a pain to manually ensure that you have installed all the cookbooks that another cookbook depends on.
- You have to download each and every one of them manually only to find out that with each downloaded cookbook, you inherit another set of dependent cookbooks.
- And even if you use knife cookbook site install, which installs all the dependencies locally for you, your cookbook directory and your repository get cluttered with all those cookbooks. Usually, you don't really care about all those cookbooks and don't want to see or manage them.

Chef Roles:

- A role is a way to define certain patterns and processes that exist across nodes in an organization as belonging to a single job function.
- Roles in Chef are a logical way of grouping nodes.
- Each role consists of zero (or more) attributes and a run-list.
- Typical cases are to have roles for web servers, database servers, and so on.
- One can set a custom run list for all the nodes and override attribute value within roles.

- When a chef-client runs, it merges its own attributes and run-lists with those contained within each assigned role.

Clean up chef server:

- The cleanup-bifrost subcommand removes unused authorization objects from the authorization database (called bifrost).
- This subcommand has the following syntax: Copy. chef-server-ctl cleanup-bifrost OPTIONS.
- The cleanup-bifrost subcommand removes unused authorization objects from the authorization database (called bifrost).
- For most users, the unused authorization objects do not substantially affect the performance of Chef Infra Server; however in certain situations it can be helpful to clean them up.
- This command is primarily intended for use by Chef support.