**Chef is Pull based configuration management tool**

Chef is a **configuration management tool** used to automate infrastructure.

* **Chef** is a **configuration management tool** used in **DevOps** to **automate infrastructure**. Think of it like a **"robot chef"** in a restaurant kitchen that cooks (configures) everything exactly the same way every time.
* **Method to Automate the task.**
* **Use by Facebook,Aws,Hp.**

**Configuration Management (CM)** means:

* Setting up computers (servers) the way you want — automatically. (So instead of doing things **manually** on every computer (like installing software, setting settings, or creating files), you write a script once, and it takes care of everything — **the same way** on all systems.)
* **Configuration Management :** how your systems are set up."

**Configuration Management Tools: turns your code to infrastructure**

* Tools that help you automatically set up and manage computers or servers.
* They turn manual IT tasks (like installing software) into scripts that run on many machines.
* They help DevOps teams save time and avoid mistakes by automating setup tasks.
* Think of them as remote controls for managing thousands of computers at once.
* They let you treat your infrastructure like code — you write it, test it, and run it.

Ohia

Recipe 1 1

Recipe 1 1

Node 3 or Server

Node1

Node 2

Node 1

Work Station

Boostrap

Knife

Knife

Knife

Knife

Knife

Knife

Cookbook store here

Cookbook

Ohia

Ohia

Chef Server or Server

**Cookbook**: A collection of configuration files (recipes), Collection of Recipes. Cookbooks are collections of recipes.

**Chef Workstation:** Where you write the code.

* Workstations are personal Computers or Virtual Servers where all Configuration Code is Created, tested or Changed.
* The machine where the infrastructure code is written. This is where you develop your cookbooks and recipes before pushing them to the Chef Server.
* Devops engineer actually sits here and write Codes This Code is Called Recipe. A Collection of Recipes are Known as Cookbook
* Workstation Communicate with the Chef Server using knife.
* Knife is a Command line tool that uploads the Cookbook to the Server.

**Chef Server:** Where you upload the code.

* This is the central repository that stores the cookbooks, policies, and metadata used to manage infrastructure.
* Chef-Server is a middle-man between workstation and the nodes.
* All Cookbooks are stored here.
* Server may be hosted locally or remote.

**Nodes: Where you apply the code.**

* **Nodes are the systems that require the Configuration.**
* **Ohai fetches the Current State of the node its located in.**
* **Node Communicate with the Chef-server using the Chef-Client.**
* **Each node Can have a different Configuration Required.**
* **Chef-Client is installed on every node.**

The central hub where all configuration data, cookbooks, and recipes are stored. It acts as the source of truth for your infrastructure.

**Chef Client:** This is the agent installed on the servers and workstations that executes the code defined in the cookbooks and applies the desired state.

This is installed on every node (server) that you want to manage. It pulls configuration details from the Chef Server and applies them.

**Recipe** : The code we write in cookbook is called recipe(installation of windows code is called recipe)

**Knife**: Transfer things from workstation to chef server. Knife is **Chefs command-line tool** to interact with the Chef server.

**Cookbooks and Recipes:** Cookbooks are a collection of recipes, which are the actual code written to describe the desired state of your system.

**Run Lists:** A run list is an ordered list of recipes that the Chef client runs on a node.

**Bootstrap**: Chef server connect to node the process is called bootstrap.

**Ohai: Ohai is a tool for collecting system configuration data**, which it then provides to Chef Infra Client to use in cookbooks.

**Ohai is a tool that is used to collect system configuration data**, which is provided to the chef-client for use within cookbooks. Ohai is run by the chef-client.

The types of attributes Ohai collects include but aren’t limited to:

* Operating System
* Network
* Memory
* Disk
* CPU
* Kernel
* Host names
* Fully qualified domain names
* Virtualization
* Cloud provider metadata

**Idempotency:**  **idempotency is about detecting any changes and correcting only these changes**.

**Chef Supermarket: Where you get custom code.**

Cookbook is a collection of recipes and some other files and folders.

Inside Cookbook:

* Chefignore = Like gitignore.
* Kitchen yml= for testing Cookbook.
* Metadeta.rb = name, version, author etc of Cookbook.
* Readme md = information about usage of Cookbook.
* Recipe - Where you write Code
* Spec = for unit test
* Test= for integration test

**Adavantages of configuration management tool:**

* **Complete automation**
* **Increase uptime (can use server).**
* **Manually installing the software on a server typically takes around 1 hour. By using a configuration management tool, we’ve automated this process and reduced the installation time to less than 10 minutes. This not only accelerates deployment but also minimizes downtime, allowing the server to be up and running nearly 50 minutes faster than before.**
* **Decrease Downtime (server down).**
* **Improve performance.**
* **Ensure compliance (**We follow all rules and regulations during the work process**).**
* **Prevent Errors.**
* **Reduce Cost.**

Download & install Chef and create Cookbook, Recipes

Method 1: All details in txt file .

curl -LO https://omnitruck.chef.io/install.sh

sudo bash install.sh -P chef-workstation

* Method 2: wget <chef download link>
* Yum install <chef -workstation downloaded file> //paste the link here.
* mkdir cookbooks //make a folder name cookbooks.. on desktop
* cd cookbooks/ //go inside the cookbook.
* chef generate cookbook zee-cookbook //now generate a cookbook there you will lot of folders and recipe too
* cd zee-cookbook // go to cookbook //here name of the cookbook is zee-cookbook you can give your name also here
* yum install tree –y //it will install the package tree to see details of files.
* Tree //now you can use this command to check
* chef generate recipe zee-recipe //now generate recipe here you can give any name here I give zee-recipe.
* Cd .. //now go back
* vi zee-cookbook/recipes/zee-recipe.rb //you have to type this command in [cookbooks] only there.

file '/myfile' do

content 'Welcome to Technical Guftgu'

action :create

end

* Chef exec ruby -c zee-cookbook/recipes/zee-recipe.rb //now check the erroe use this command. //Syntax OK
* Chef-client -zr “recipe[zee-cookbook::zee-recipe]” //run the recipe
* Cat /myfile(xyz) (also try ls /) //(to check inside the file)

Now create more recipe like I created first : Apache server: [cookbooks]#chef generate cookbook apache-cookbook

The code recipe will be in the txt file : name is apache.txt

**What Are Attributes?Like ceo example**

* **Key-value pairs** that describe something about a node (e.g., ip\_address: 192.168.1.1)
* Used by the **Chef client** to manage the node's configuration.

**Why Are Attributes Used?**

To decide:

1. What is the **current state** of the node?
2. What was the **previous state** (from last Chef run)?
3. What should be the **desired state** after the current run?

**Types of Attributes (with Priority) A**ll **F**orceful **O**verrides **N**ormal **F**unctions **D**own

(Ohai defines automatic attributes — highest priority)

|  |  |  |
| --- | --- | --- |
| **Type** | **Priority** | **Notes** |
| **automatic** | Highest (1st) | Set by **Ohai** (system info) |
| **force\_override** | 2nd Highest | Rare, but powerful override |
| **override** | 3rd | Can override default values |
| **normal** | 4th | Set manually or by cookbooks |
| **force\_default** | 5th | Forces a default value |
| **default** | Lowest (6th) | Common, basic setting |

**Who Defines Attributes?**

* **Node**
* **Cookbooks**
* **Roles**
* **Environments**

RUNLIST:

* A **list of tasks** (recipes/roles) that Chef runs **in order** on a node.
* To run the recipe in a sequence order that we mention in a run list. With this process we can run multiple recipes but the condition is, they must be only one recipe from one cookbook.

[cookbooks]# chef-client -zr “recipe[**zee-cookbook**::default],**recipe[Apache-cookbook::default**]”

recipe[apache-cookbook::default] is the another cookbook and we will be able to run it

The zee-cookbook is the name of cookbook we created

The red part is to run the script

This will run the default recipe in the cookbook you can use another recipes as well like we can have :

[Zee-cookbook::test-recipe]

vi Zee-cookbook/recipes/default.rb

Write there

Default recipe

It contains the cookbook where I have to add 2 or recipes so I can add the names of the recipe here

ABC-cookbook: is the cookbook name which we create using chef generate cookbook it should be same which name you have created

Xyz is the recipe name

You can add as many as recipe here but the cookbook should be same

Like this

Include\_recipe “ABC-cookbook::XYZ-recipe”

Include\_recipe “ABC-cookbook::mudi-recipe”

Include\_recipe “ABC-cookbook::google-recipe”

Include\_recipe “ABC-cookbook::XYZ-recipe”

Include\_recipe “ABC-cookbook::1-recipe”

Include\_recipe “ABC-cookbook::2-recipe”

vi ABC-cookbook/recipes/default.rb

vi google-cookbook/recipes/default.rb

vi test-cookbook/recipes/default.rb

Include\_recipe “google-cookbook::recipe1”

Include\_recipe “google-cookbook::recipe2”

Include\_recipe “test-cookbook::google-recipe”

Include\_recipe “test-cookbook::google-recipe”

**[cookbooks]# chef-client -zr “recipe[ABC-cookbook::default],recipe[google-cookbook::default], recipe[test-cookbook::default]”**

**Connect workstation to chef server to node using chef-repo, bootstrap**

* Chef server is going to mediator between the code and cookbooks.
* Bootstrapping Attaching a node to chef server, by Bootstrap command, both workstation and node should be in same AZ. Two actions will be done while bootstrapping 1. adding node to chef server 2. installing chef package.
* Chef-repo It would be the main directory inside it you have to run any commons, it is also having cookbooks).
* Create chef manage account by “manage.chef.io” and download starter kit. Go to download and extract file chef-repo, after extracting we got more files inside chef-repo are (.chef ,cookbooks ,gitignore, README.md, roles)
* For sending chef-repo file to Linux machine we use the software called WinSCP. Drag Chef-repo from left window and drop to right Linux window. (by ls command in you can check Chef-repo is showing in your workstation or not)
* **Sudo su.**
* **Ls output:: chef-repo**
* **cd chef-repo**
* **ls-a**
* **cd .chef // output config.rb Wafzee.pem**
* **knife ssl check // to check workstation is connected with chef server. Connecting to host api.chef.io:443 Successfully verified certificates from ‘api.chef.io’**
* Create Linux machine (Node1) same AZ of workstation with new security group and new key pair name node1-key, save Private IP for further knife bootstrap commands.
* Attach Advance details
* [ #!/bin/bash

Sudo su

Yum update -y]

* With the help of WinSCP please transfer downloaded node1-key.pem to Chef-repo for bootstrap command.
* Now go to chef workstation and execute Bootstrap command to attach node1 to chef-server.
* [chef-repo] # knife bootstrap 172.31.10.120 -–ssh-user ec2-user -–sudo -i node 2key.pem -N node1
* // Node has been connected to server and node package has installed.
* [chef-repo] # knife node list



**Data Bags**

* Store global JSON-based data (e.g., passwords).
* Can be **encrypted** for sensitive info.
* **Commands:**
  + Create bag: knife data bag create my\_bag
  + Add item: knife data bag create my\_bag my\_item --json-file my\_item.json

**Environments**

* Manage different setups: **dev**, **test**, **prod**, etc.
* Define **attributes** & **run lists** per environment.
* **Command:**
  + knife environment create production

**Roles**

* Define configuration patterns (e.g., webserver).
* Include **run list** & **default attributes**.
* **Command:**
  + knife role create webserver
* Helps apply consistent config to multiple nodes.

**Chef Vault**

* Encrypted version of data bags.
* Securely share secrets between nodes.
* **Command:**
  + knife vault create vault\_name item\_name --search 'role:ROLE\_NAME' --admins 'ADMIN\_NAME'

**Berkshelf**

* Manages cookbook **dependencies**.
* Keeps all cookbooks in sync.
* **Install:**
  + gem install berkshelf
* **Berksfile Example:**

ruby

CopyEdit

source 'https://supermarket.chef.io'

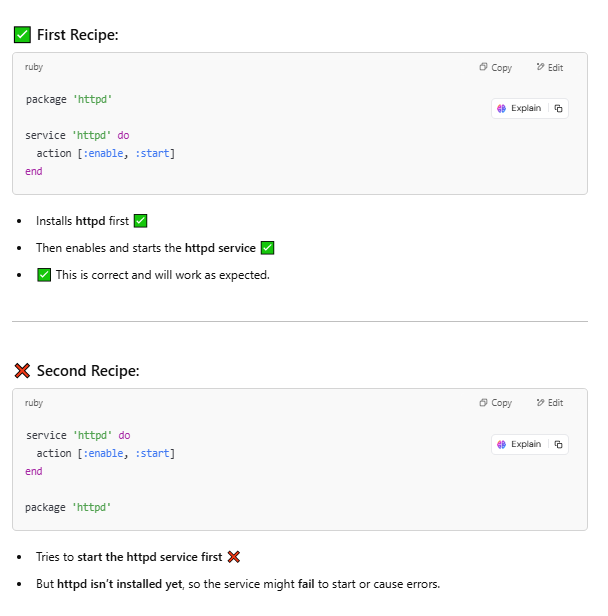
cookbook 'apache2'

cookbook 'mysql', '~> 8.0'

## What happens if you don’t specify a resource’s action in Chef?

👉 **Chef applies the default action of that resource.**

If you **don’t specify an action**, Chef automatically uses the **default** one for that resource.



**Write a service Resource that stops and then disables the httpd service from starting when the system boots in Chef.**

Use the below Resource to stop and disable the httpd service from starting when system boots.

|  |  |
| --- | --- |
| 1  2  3 | **service 'httpd' do**  **action [:stop, :disable]**  **end** |

**What information do you need in order to bootstrap in Chef?**

Just mention the information you need in order to bootstrap:

* Your node’s host name or public IP address.
* A user name and password you can log on to your node with.
* Alternatively, you can use key-based authentication instead of providing a user name and password.

### ****What is the command you use to upload a cookbook to the Chef server?****

You can directly mention the command to upload a cookbook to the Chef server **“knife cookbook upload”**.

### ****Where can you get reusable cookbooks that are written and maintained by the Chef community?****

You can directly answer this question by saying reusable Cookbooks are present at Chef Supermarket, [***https://supermarket.chef.io****.*](https://supermarket.chef.io./)