**What is Chef:-**

Chef is a configuration management tool that is written in Ruby. It's capable of managing both your on premise and cloud servers with ease.

You can easily manage up to 10000 nodes using chef. Replicating the infrastructure components is easy once we have them automated via chef.

Chef works with three core components: The Chef server, workstations, and nodes. The Chef server is the hub of Chef operations, where changes are stored for use. Workstations are static computers or virtual servers where all code is created or changed. There can been as many workstations as needed, whether this be one per person or otherwise. Finally, nodes are the servers that need to be managed by Chef – these are the machines that changes are being pushed to, generally a fleet of multiple machines that require the benefits of an automation program.

**The Chef Server:-**

The Chef server is the primary mode of communication between the workstations where your infrastructure is coded, and the nodes where it is deployed. All configuration files, cookbooks, metadata, and other information are stored on the server. The Chef server also keeps information regarding the state of all nodes at the time of the last [chef-client](https://www.linode.com/docs/applications/chef/beginners-guide-chef#chef-client) run.

Any changes made must pass through the Chef server to be deployed. Prior to accepting or pushing changes, it verifies that the nodes and workstations are paired with the server through the use of authorization keys, and then allows for communication between the workstations and nodes.

## Workstations:-

Workstations are where users create, test, and maintain cookbooks and policies that will be pushed to nodes. Cookbooks created on workstations can be used privately by one organization, or uploaded to the Chef Supermarket for others to use. Similarly, workstations can be used to download cookbooks created by other Chef users and found in the Supermarket.

Workstations are set up to use the *Chef Development Kit* (ChefDK), and can be located on virtual servers or on physical workstation computers. Workstations are set to interact with only one Chef server, and most work will be done in the chef-repo directory located on the workstation.

### chef-repo:-

The chef-repo directory is the specific area of the workstation where cookbooks are authored and maintained. The chef-repo is always version-controlled, most often through the use of Git, and stores information and history that will be used on nodes, such as cookbooks, environments, roles, and data bags. Chef is able to communicate with the server from the chef-repo and push any changes via the use of the knife command, which is included in the ChefDK.

Originally the chef-repo had to be pulled from GitHub using git commands, but that action is now integrated into Chef through the use of the chef generate repo chef-repo command.

### Knife:-

The knife command communicates between the chef-repo located on a workstation and the Chef server. knife is configured with the knife.rb file, and is used from the workstation:

~/chef-repo/.chef/knife.rb

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | log\_level :info  log\_location STDOUT  node\_name 'username'  client\_key '~/chef-repo/.chef/username.pem'  validation\_client\_name 'shortname-validator'  validation\_key '~/chef-repo/.chef/shortname.pem'  chef\_server\_url 'https://123.45.67.89/organizations/shortname'  syntax\_check\_cache\_path '~/chef-repo/.chef/syntax\_check\_cache'  cookbook\_path [ '~/chef-repo/cookbooks' ] |

## Nodes:-

A *node* is a system configured to run the chef-client. This can be any system, as long as it is being maintained by Chef.

Nodes are validated through the validator.pem and client.pem certificates that are created on the node when it is bootstrapped. All nodes must be bootstrapped over SSH as either the root user or a user with elevated privileges.

Nodes are kept up-to-date through the use of the chef-client, which runs a convergence between the node and the Chef server. What cookbooks and roles the node will take on depends on the run list and environment set for the node in question.

### chef-client

The chef-client checks the current configuration of the node against the recipes and policies stored in the Chef server and brings the node up to match. The process begins with the chef-client checking the node’s [run list](https://www.linode.com/docs/applications/chef/beginners-guide-chef#run-lists), loading the cookbooks required, then checking and syncing the cookbooks with the current configuration of the node.

The chef-client must be run with elevated privileges in order to properly configure the node, and should be run periodically to ensure that the server is always up to date – often this is achieved through a cron job or by setting up the chef-client to run as a service.

### Run Lists

Run lists define what cookbooks a node will use. The run list is an ordered list of all cookbooks and recipes that the chef-client needs to pull from the Chef server to run on a node. Run lists are also used to define [roles](http://docs.chef.io/server_manage_roles.html), which are used to define patterns and attributes across nodes.

### Ohai

Ohai collects information regarding nodes for the Chef server. It is required to be present on every node, and is installed as part of the bootstrap process.

The information gathered includes network and memory usage, CPU data, kernel data, hostnames, FQDNs, and other automatic attributes that need to remain unchanged during the chef-client run.

## Cookbooks

Cookbooks are the main component of configuring nodes on a Chef infrastructure. Cookbooks contain values and information about the *desired state* of a node, not how to get to that desired state – Chef does all the work for that, through their extensive libraries.

Cookbooks are comprised of recipes, metadata, attributes, resources, templates, libraries, and anything else that assists in creating a functioning system, with attributes and recipes being the two core parts of creating a cookbook. Components of a cookbook should be modular, keeping recipes small and related.

Cookbooks can and should be version controlled. Versions can help when using environments and allow for the easier tracking of changes that have been made to the cookbook.

### Recipes

Recipes are the fundamental part of cookbooks. Recipes are written in Ruby and contain information in regards to everything that needs to be run, changed, or created on a node. 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.

### Attributes

Attributes define specific values about a node and its configuration. These values are used to override default settings, and are loaded in the order cookbooks are listed in the run list. Often attributes are used in conjunction with templates and recipes to define settings.

### Files

These are static files that can be uploaded to nodes. Files can be configuration and set-up files, scripts, website files – anything that does not been to have different values on different nodes.

### Providers and Resources

Providers and resources are also used to define new functionality to use in Chef recipes. A *resource* defines a set of actions and attributes, whereas *provider* informs the chef-client how to commit each action.

### Templates

Templates are embedded Ruby files (.erb) that allows for content based on the node itself and other variables generated when the chef-client is run and the template is used to create or update a file.

Also gone through below chef Reference links

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Tutorial+Introduction>

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Tutorial+1>

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Tutorial+2>

<http://confluence.mpls.digitalriver.com/display/serv/Chef+-+Current+Environment+Details+at+DR>

<http://confluence.mpls.digitalriver.com/display/serv/Git+Commands+for+use+with+Chef+Development>

<https://www.linode.com/docs/applications/chef/beginners-guide-chef>

<https://www.linode.com/docs/applications/chef/deploy-a-chef-server-workstation-and-node-on-ubuntu-14-04/>

<https://www.linode.com/docs/applications/chef/creating-your-first-chef-cookbook>

**Installation of Chef:-**

Chef has three major components:

1. Workstation
2. Server
3. Nodes

Following are the steps to install Chef:

* 1. Install Chef DK (Development Kit) on Chef Workstation
  2. Setup a Chef Server
  3. Create a Recipe or a Cookbook / download a Cookbook from Chef Supermarket in Workstation
  4. Upload a Cookbook on the Chef Server
  5. Connect A Node To The Chef Server
  6. Deploy the Cookbook from the Server to the Node

## ****Install Chef DK (Development Kit):-****

## In my Chef Workstation I will install Chef DK. Chef DK is a package that contains all the development tools that you will need when coding Chef.

## Download the chef development kit.

## #wget <https://packages.chef.io/stable/el/6/chefdk-1.0.3-1.el6.x86_64.rpm>

## The package is now downloaded. Now install this package using rpm.

## #rpm -ivh chefdk-1.0.3-1.el6.x86\_64.rpm

## ****Setup Chef Server:-****

## I will use the hosted version of Chef Server on the cloud but you can use a physical machine as well. This Chef-Server is present at [manage.chef.io](http://manage.chef.io/)

## Over here, create an account if you don’t have one. Once you have created an account, sign-in with your login credentials.

## If you are signing in for the first time, the very first thing that you will be doing is creating an organization. Organization is basically a group of Machines that you will be managing with the Chef Server.

## First, I will go to the administration tab. Over there, I have already created an organization called edu. So I need to download the starter kit in my Workstation. This starter kit will help you to push files from the Workstation to the Chef Server. Click on the settings icon on the right hand side and click on Starter Kit.

## Now you will get an option to download the Starter Kit. Just click on it to download the Starter Kit zip file.

## Move this file to your root directory. Now unzip this file by using unzip command in your terminal. You will notice that it includes a directory called chef-repo.

## #unzip chef-starter.zip

## Now move this starter kit to the cookbook directory in chef-repo directory.

## Execute this:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **#mv starter /root/chef-repo/cookbooks** Download A Cookbook From Chef Supermarket In Workstation:- Chef Cookbooks are available in the Cookbook Supermarket, we can go to the Chef Supermarket. Download the required Cookbooks from[supermarket.chef.io](http://supermarket.chef.io/). I’m downloading one of the Cookbook to install Apache from there.  Execute this:   |  |  | | --- | --- | |  | **#cd chef-repo**  **#knife cookbook site download learn\_chef\_httpd** |  There is Tar ball downloaded for the Apache Cookbook. Now, I will extract the contents from this downloaded Tar file. For that, I will use tar command. Execute this:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | **#tar -xvf learn\_chef\_httpd-0.2.0.tar.gz**  All the required files are automatically created under this Cookbook. There is no need to make any modifications. Let’s check the Recipe description inside my recipe folder.  Now, I will just upload this cookbook to my Chef Server as it looks perfect to me  Upload A Cookbook In The Chef Server  **#cd /root/chef-repo/learn\_chef\_httpd/recipes**  **#cat default.rb**  In order to upload the Apache Cookbook that I have downloaded, first move this learn\_chef\_httpd file to the Cookbooks folder in the chef-repo. Then change your directory to cookbooks.  Execute this:   |  |  | | --- | --- | |  | **#mv /root/chef-repo/learn\_chef\_httpd /root/chef-repo/cookbooks**  **#cd /root/chef-repo/cookbooks** |   Now in this directory, execute the below command to upload the Apache Cookbook:  Execute this:   |  |  | | --- | --- | |  | **#knife cookbook upload learn\_chef\_httpd** |   Now, our final step is to add Chef Node. We’ve setup a Workstation, a Chef Server and we need to add our Nodes to the Chef Server for automation.  **5. Connect A Node To The Chef Server:-**  The terminal color of my Node machine is different from the Workstation so that you will be able to differentiate between both.  I just need the IP address of my Node for that I will execute the below command in my Node machine.  Execute this:   |  |  | | --- | --- | |  | **#ifconfig** |   I will add my Chef Node to the Server by executing Knife Bootstrap command in which I will specify the IP address of The Chef Node and its name. Execute the command shown below:  **Execute this:**  **#knife bootstrap 192.168.56.102 --ssh-user root --ssh-password edureka --node-name chefNode**  This command will also initialize the installation of the Chef-Client in the Chef Node. You can verify it from the CLI on the Workstation using the knife command,as shown below:  Execute this:   |  |  | | --- | --- | |  | **#Knife node list** |  6. Deploy The Cookbook From The Server To The Node Let’s see how we can add a Cookbook to the Node and manage its Run list from the Chef Server. Run List describes the order in which the Cookbooks should be executed. As you can see in the screenshot below, click the Actions tab and select the Edit Run list option to manage the Run list.  In the Available Recipes,  you can see our learn\_chef\_httpd Recipe, you can drag that from the available packages to the current Run List and save the Run list.  Now login to your Node and just run chef-client to execute the Run List.  **#chef-client**  After successfully run the chef client cookbooks and recipes have applied to node. | |