

## Chef Tutorial

Chef Tutorial is the second blog of Chef blog series. In my [previous blog](#), I have explained what is Chef, Configuration Management and how Chef achieves Configuration Management with the help of a use-case of Gannett.

In this Chef Tutorial following topics will be covered:

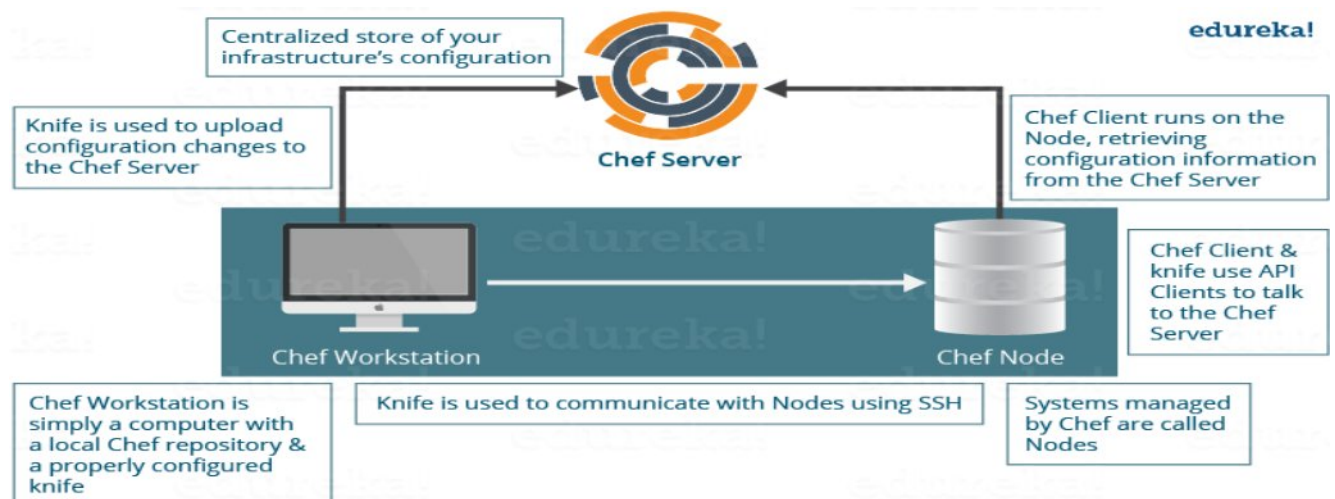
- [Chef Architecture](#)
- [Hands-on Demo](#)

I am sure after reading my [previous blog](#) you must be curious to know how exactly Chef works. The first section of this Chef Tutorial blog will explain you the Chef architecture in detail, which will clear all your doubts.

### Chef Tutorial – Chef Architecture

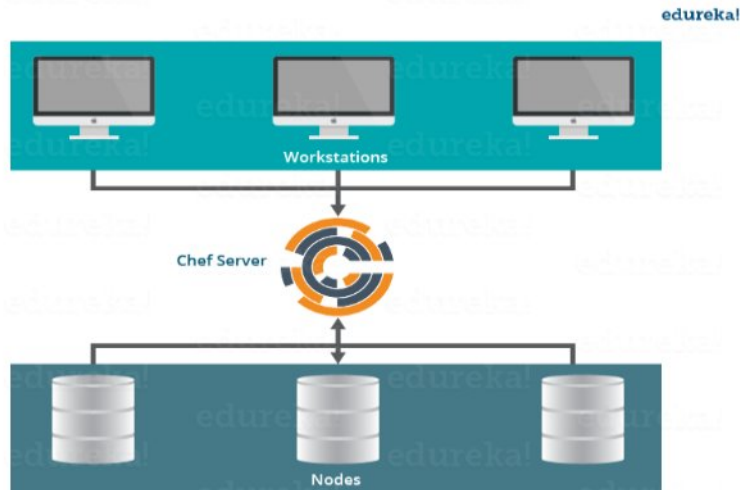
As shown in the diagram below, there are three major Chef components:

- Workstation
- Server
- Nodes



### Chef Tutorial – Workstation

The Workstation is the location from which all of Chef configurations are managed. This machine holds all the configuration data that can later be pushed to the central Chef Server. These configurations are tested in the workstation before pushing it into the Chef Server. A workstation consists of a command-line tool called Knife, that is used to interact with the Chef Server. There can be multiple Workstations that together manage the central Chef Server.



Workstations are responsible for performing the below functions:

- Writing Cookbooks and Recipes that will later be pushed to the central Chef Server
- Managing Nodes on the central Chef Server

Now, let us understand the above mentioned points one by one.

Writing Cookbooks and Recipes that will later be pushed to the central Chef Server

**Recipes:** A Recipe is a collection of resources that describes a particular configuration or policy. It describes everything that is required to configure part of a system. The user writes Recipes that describe how Chef manages applications and utilities (such as Apache HTTP Server, MySQL, or Hadoop) and how they are to be configured.

These Recipes describe a series of resources that should be in a particular state, i.e. Packages that should be installed, services that should be running, or files that should be written.

Later in the blog, I will show you how to write a Recipe to install Apache2 package on Chef Nodes by writing a ruby code in Chef Workstation.

**Cookbooks:** Multiple Recipes can be grouped together to form a Cookbook. A Cookbook defines a scenario and contains everything that is required to support that scenario:

- Recipes, which specifies 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

## Managing Nodes on the central Chef Server

The Workstation system will have the required command line utilities, to control and manage every aspect of the central Chef Server. Things like adding a new Node to the central Chef Server, deleting a Node from the central Chef Server, modifying Node configurations etc can all be managed from the Workstation itself.

Now let us see, what components of Workstation are required to perform the above functions.

Workstations have two major components:

**Knife utility:** This command line tool can be used to communicate with the central Chef Server from Workstation. Adding, removing, changing configurations of Nodes in a central Chef Server will be carried out by using this Knife utility. Using the Knife utility, Cookbooks can be uploaded to a central Chef Server and Roles, environments can also be managed. Basically, every aspect of the central Chef Server can be controlled from Workstation using Knife utility.

**A local Chef repository:** This is the place where every configuration component of central Chef Server is stored. This Chef repository can be synchronized with the central Chef Server (again using the knife utility itself).

### Chef Tutorial – Chef Server

The Chef Server acts as a hub for configuration data. The Chef Server stores Cookbooks, the policies that are applied to Nodes, and metadata that describes each registered Node that is being managed by the Chef-Client.

Nodes use the Chef-Client to ask the Chef Server for configuration details, such as Recipes, Templates, and file distributions. The Chef-Client then does as much of the configuration work as possible on the Nodes themselves (and not on the Chef Server). Each Node has a Chef Client software installed, which will pull down the configuration from the central Chef Server that are applicable to that Node. This scalable approach distributes the configuration effort throughout the organization.

### Chef Tutorial – Chef Nodes

Nodes can be a cloud based virtual server or a physical server in your own data center, that is managed using central Chef Server. The main component that needs to be present on the Node is an agent that will establish communication with the central Chef Server. This is called Chef Client.

Chef Client performs the following functions:

- It is responsible for interacting with the central Chef Server.
- It manages the initial registration of the Node to the central Chef Server.
- It pulls down Cookbooks, and applies them on the Node, to configure it.
- Periodic polling of the central Chef Server to fetch new configuration items, if any.

## Chef Tutorial – Advantages of Chef:

This Chef tutorial will be incomplete if, I don't include the key benefits of Chef:

- You can automate an entire infrastructure using Chef. All tasks that were manually being done, can now be done via Chef tool.
- You can configure thousands of nodes within minutes using Chef.
- Chef automation works with the majority of the public cloud offerings like [AWS](#).
- Chef will not only automate things, but will also keep the systems under consistent check, and confirm that the system is in fact configured the way it is required (Chef Agent/Client does this job). If somebody makes a mistake by modifying a file, Chef will correct it.
- An entire infrastructure can be recorded in the form of a Chef repository, that can be used as a blueprint to recreate the infrastructure from scratch.

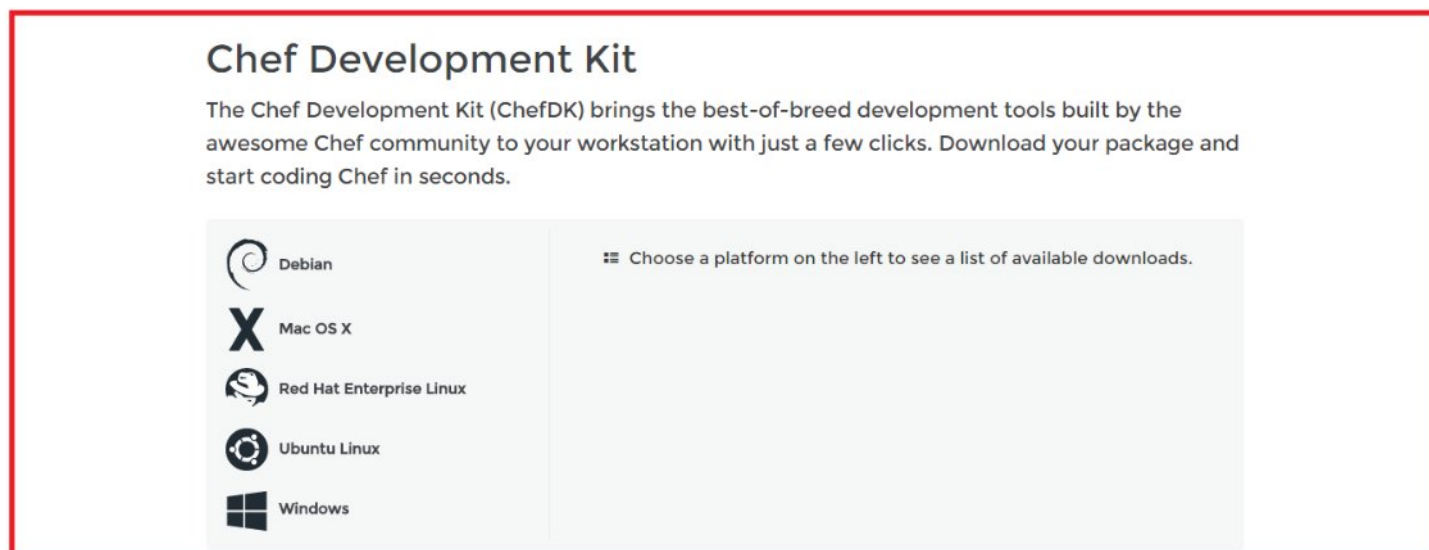
## Chef Tutorial – Hands-On

Here I will explain you how to create a Recipe, Cookbook and Template in Chef Workstation. I will also explain you how to deploy a Cookbook from Workstation to Chef-Client (Chef Node).

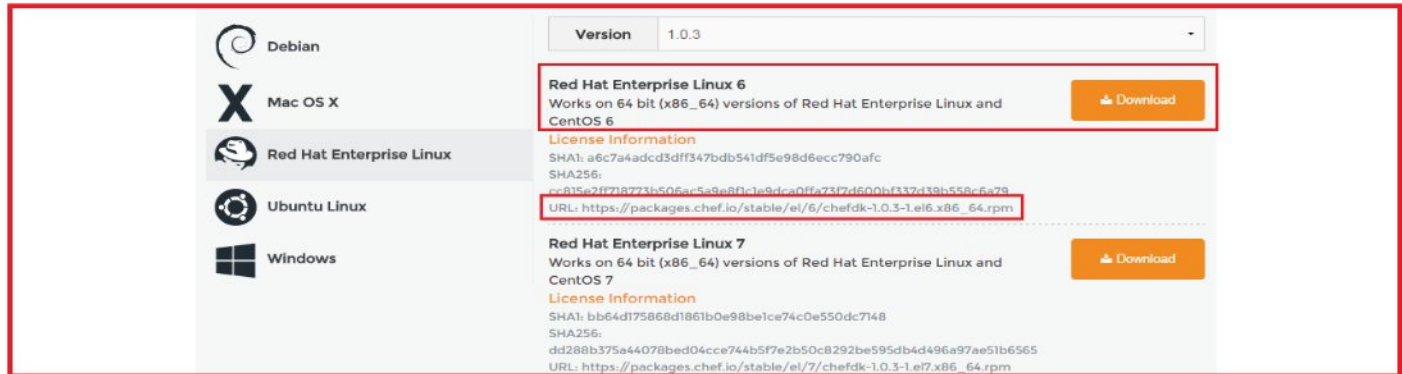
I am using two Virtual Images one for Chef Workstation and other for Chef Node. For Chef Server I will use the hosted version of Chef (on cloud). You can also use a physical machine for Chef Server as well.

Step 1: Install Chef DK (Development Kit) in your Chef Workstation.

Chef DK is a package that contains all the development tools that you will need when coding Chef. Here is the link to download [Chef DK](#).



Here, choose the operating system that you are using. I am using CentOS 6.8 .So, I will click on Red Hat Enterprise Linux.



Copy the link according to the version of CentOS that you are using. I am using CentOS 6, as you can see that I have highlighted in the above screenshot.

Go to your Workstation terminal and download the Chef DK by using wget command and paste the link.

Execute this:

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

```
[root@Workstation ~]# wget https://packages.chef.io/stable/el/6/chefdk-1.0.3-1.el6.x86_64.rpm
--2016-11-25 07:40:38-- https://packages.chef.io/stable/el/6/chefdk-1.0.3-1.el6.x86_64.rpm
Resolving packages.chef.io... 151.101.8.65
Connecting to packages.chef.io[151.101.8.65]:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://packages.chef.io/files/stable/chefdk/1.0.3/el/6/chefdk-1.0.3-1.el6.x86_64.rpm [following]
--2016-11-25 07:40:44-- https://packages.chef.io/files/stable/chefdk/1.0.3/el/6/chefdk-1.0.3-1.el6.x86_64.rpm
Reusing existing connection to packages.chef.io:443.
HTTP request sent, awaiting response... 200 OK
Length: 105276444 (100M) [application/x-rpm]
Saving to: "chefdk-1.0.3-1.el6.x86_64.rpm"

100%[=====>] 105,276,444 300K/s in 3m 19s
2016-11-25 07:44:15 (517 KB/s) - "chefdk-1.0.3-1.el6.x86_64.rpm" saved [105276444/105276444]
```

The package is now downloaded. It is time to install this package using rpm.

Execute this:

```
1 rpm -ivh chefdk-1.0.3-1.el6.x86_64.rpm
```

```
[root@Workstation ~]# rpm -ivh chefdk-1.0.3-1.el6.x86_64.rpm
warning: chefdk-1.0.3-1.el6.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID 83ef826a: NOKEY
Preparing... ##### [100%]
1:chefdk ##### [100%]
Thank you for installing Chef Development Kit!
[root@Workstation ~]#
```

Chef DK is now installed in my Workstation.

Step 2: Create a Recipe in the Workstation

Let us start by creating a Recipe in the Workstation and test it locally to ensure it is working. Create a folder named chef-repo. We can create our Recipes inside this folder.

Execute this:

```
1 mkdir chef-repo
2 cd chef-repo
```

```
[root@Workstation ~]# mkdir chef-repo
[root@Workstation ~]# cd chef-repo/
[root@Workstation chef-repo]#
```

In this chef-repo directory I will create a Recipe named edureka.rb. .rb is the extension used for ruby. I will use vim editor, you can use any other editor that you want like gedit, emacs, vi etc.

Execute this:

```
1 vim edureka.rb
```

Here add the following:

```
1 file '/etc/motd'
2 content 'Welcome to Chef'
3 end
```

```
file '/etc/motd' do
  content 'Welcome to Chef'
end
```

This Recipe edureka.rb creates a file named /etc/motd with content "Welcome to Chef".

Now I will use this Recipe to check if it is working.

Execute this:

```
1 chef-apply edureka.rb
```

```
[root@Workstation chef-repo]# chef-apply edureka.rb
Recipe: (chef-apply cookbook)::(chef-apply recipe)
 * file[/etc/motd] action create
  - update content in file /etc/motd from e3b0c4 to 40a30c
  --- /etc/motd                2010-01-12 13:28:22.000000000 +0000
  +++ /etc/.chef-motd20161125-5713-ba857q    2016-11-25 09:00:05.262933936 +0000
  @@ -1 +1,2 @@
  +Welcome to Chef
  - restore selinux security context
[root@Workstation chef-repo]#
```

So there is a file created in the chef-repo that has content Welcome to Chef.

Step 3: Modifying Recipe file to install httpd package

I will modify the Recipe to install httpd package on my Workstation and copy an index.html file to the default document root to confirm the installation. The default action for a package resource is installation, hence I don't need to specify that action separately.

Execute this:

```
1 vim edureka.rb
```

Over here add the following:

```
1 package 'httpd'
2 service 'httpd' do
3   action [:enable, :start]
4 end
5 file '/var/www/html/index.html' do
6   content 'Welcome to Apache in Chef'
7 end
```



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

file '/var/www/html/index.html' do
  content 'Welcome to Apache in Chef'
end
```

Now I will apply these configurations by executing the below command:

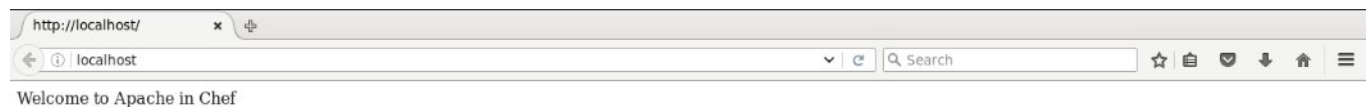
Execute this:

```
1 chef-apply edureka.rb
```

```
[root@Workstation chef-repo]# chef-apply edureka.rb
Recipe: (chef-apply cookbook)::(chef-apply recipe)
 * yum_package[httpd] action install (up to date)
 * service[httpd] action enable
   - enable service service[httpd]
 * service[httpd] action start
   - start service service[httpd]
 * file[/var/www/html/index.html] action create
   - create new file /var/www/html/index.html
   - update content in file /var/www/html/index.html from none to 152204
   -- /var/www/html/index.html      2016-11-25 10:02:52.399858608 +0000
   +++ /var/www/html/.chef-index20161125-6176-1w10rp.html      2016-11-25 10:02:52.399858608 +0000
   @@ -1 +1,2 @@
   +Welcome to Apache in Chef
   - restore selinux security context
[root@Workstation chef-repo]#
```

The command execution clearly describes each instance in the Recipe. It installs the Apache package, enables and starts the httpd service on the Workstation. And it creates an index.html file in the default document root with the content "Welcome to Apache in Chef".

Now confirm the installation of Apache2 by opening your web-browser. Type your public IP address or the name of your host. In my case, it is localhost.



Step 4: Now we will create our first Cookbook.

Create a directory called cookbooks, and execute the below command to generate the Cookbook.

Execute this:

```
1 mkdir cookbooks
2 cd cookbooks
3 chef generate cookbook httpd_deploy
```

httpd\_deploy is a name given to the Cookbook. You can give any name that you want.

```
[root@Workstation chef-repo]# mkdir cookbooks
[root@Workstation chef-repo]# cd cookbooks/
[root@Workstation cookbooks]# chef generate cookbook httpd_deploy
Generating cookbook httpd_deploy
- Ensuring correct cookbook file content
- Ensuring delivery configuration
- Ensuring correct delivery build cookbook content

Your cookbook is ready. Type `cd httpd_deploy` to enter it.

There are several commands you can run to get started locally developing and testing your cookbook.
Type `delivery local --help` to see a full list.

Why not start by writing a test? Tests for the default recipe are stored at:
test/recipes/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:
recipes/default.rb

[root@Workstation cookbooks]#
```

Let us move to this new directory httpd\_deploy.

Execute this:

```
1 cd httpd_deploy
```

Now let us see the file structure of the created Cookbook.

Execute this:

```
1 tree
[root@Workstation httpd_deploy]# tree
.
├── Berksfile
├── cheffignore
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
└── test
    └── recipes
        └── default_test.rb

6 directories, 8 files
[root@Workstation httpd_deploy]#
```

Step 5: Create a Template file.

Earlier, I created a file with some contents, but that can't fit with my Recipes and Cookbook structures. So let us see how we can create a Template for index.html page.

Execute this:

```
1 chef generate template httpd_deploy index.html
[root@Workstation httpd_deploy]# chef generate template httpd_deploy index.html
Recipe: code_generator::template
* directory[./httpd_deploy/templates/default] action create
  - create new directory ./httpd_deploy/templates/default
  - restore selinux security context
* template[./httpd_deploy/templates/index.html.erb] action create
  - create new file ./httpd_deploy/templates/index.html.erb
  - update content in file ./httpd_deploy/templates/index.html.erb from none to e3b0c4
    (diff output suppressed by config)
  - restore selinux security context
[root@Workstation httpd_deploy]#
```



```
[root@Workstation httpd_deploy]# tree
.
├── Berksfile
├── chefignore
├── httpd_deploy
│   └── templates
│       ├── default
│       └── index.html.erb
├── metadata.rb
├── README.md
├── recipes
│   └── default.rb
├── spec
│   ├── spec_helper.rb
│   └── unit
│       └── recipes
│           └── default_spec.rb
└── test
    └── recipes
        └── default_test.rb

9 directories, 9 files
```

Now if you see my Cookbook file structure, there is a folder created with the name templates with index.html.erb file. I will edit this index.html.erb template file and add my Recipe to it. Refer the example below:

Go to the default directory

Execute this:

```
1 cd /root/chef-repo/cookbook/httpd_deploy/templates/default
```

Over here, edit the index.html.erb template by using any editor that you are comfortable with. I will use vim editor.

Execute this:

```
1 vim index.html.erb
```

Now add the following:

```
1 Welcome to Chef Apache Deployment
Step 6: Create a Recipe with this template.
```

Go to the Recipes directory.

Execute this:

```
1 cd /root/chef-repo/cookbooks/httpd_deploy/recipes
```

Now edit the default.rb file by using any editor that you want. I will use vim editor.

Execute this:

```
1 vim default.rb
```

Over here add the following:

```
1
2 package 'httpd'
3 service 'httpd' do
4   action [:enable, :start]
5 end
6 template '/var/www/html/index.html' do
7   source 'index.html.erb'
8 end
```

```
# Cookbook Name:: httpd_deploy
# Recipe:: default
#
# Copyright (c) 2016 The Authors, All Rights Reserved.
package 'httpd'
service 'httpd' do
  action [:enable, :start]
end

template '/var/www/html/index.html' do
  source 'index.html.erb'
end
```

Now I will go back to my chef-repo folder and run/test my recipe on my Workstation.

Execute this:

```
1 cd /root/chef-repo
2 chef-client --local-mode --runlist 'recipe[httpd_deploy]'
```

```
[root@Workstation chef-repo]# chef-client --local-mode --runlist 'recipe[httpd_deploy]'
```

```
[2016-11-28T06:05:31+00:00] WARN: No config file found or specified on command_line, using command line options.
```

```
Starting Chef Client, version 12.16.42
```

```
Resolving cookbooks for run list: ["httpd_deploy"]
```

```
Synchronizing Cookbooks:
```

```
- httpd_deploy (0.1.0)
```

```
Installing Cookbook Gems:
```

```
Compiling Cookbooks...
```

```
Converging 3 resources
```

```
Recipe: httpd_deploy::default
```

```
* yum::package[httpd] action install (up to date)
```

```
* service[httpd] action enable (up to date)
```

```
* service[httpd] action start (up to date)
```

```
* template[/var/www/html/index.html] action create
```

```
- update content in file /var/www/html/index.html from 152204 to 748cbd
```

```
--- /var/www/html/index.html      2016-11-25 10:02:52.399858608 +0000
```

```
+++ /var/www/html/.chef-index20161128-22551-eiujfo.html  2016-11-28 06:05:51.806388896 +0000
```

```
@@ -1,2 +1,2 @@
```

```
-Welcome to Apache in Chef
```

```
+Welcome to Chef Apache Deployment
```

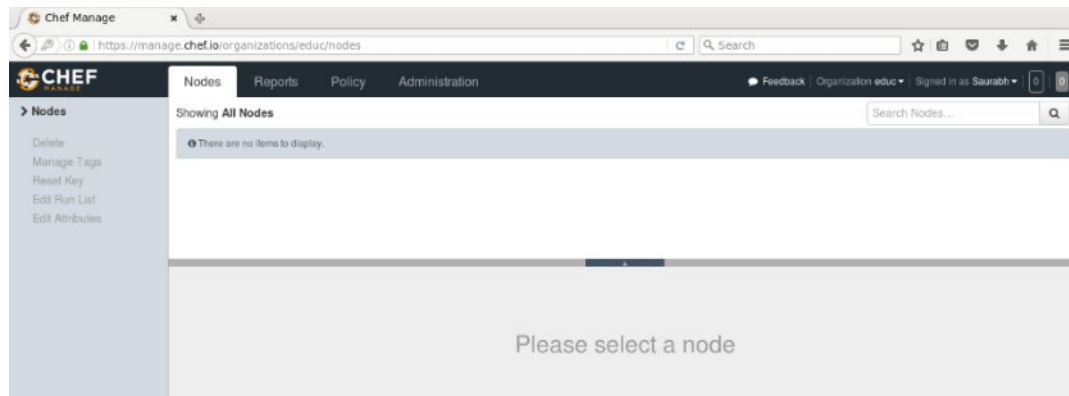
According to my Recipe, Apache is installed on my Workstation, service is being started and enabled on boot. Also a template file has been created on my default document root.

Now that I have tested my Workstation. It's time to setup the Chef Server.

## Step 7: 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)

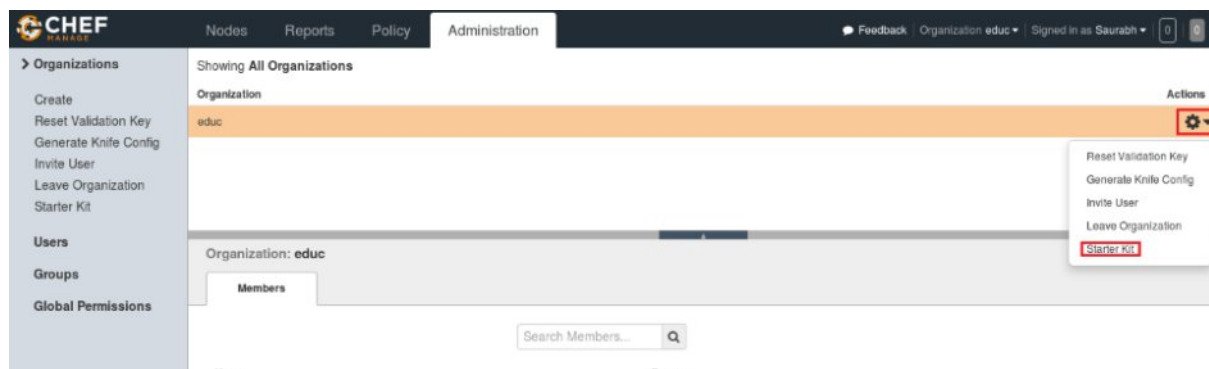
Over here create an account if you don't have one. Once you have created an account, sign-in with your login credentials.



This is how Chef Server looks like.

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.



When you click over there 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 zip file by using unzip command in your terminal. You will notice that it includes a directory called chef-repo.

Execute this:

```
1 unzip chef-starter.zip
```

```
[root@Workstation ~]# unzip chef-starter.zip
Archive:  chef-starter.zip
  inflating: chef-repo/cookbooks/chefignore
    creating: chef-repo/cookbooks/starter/
    creating: chef-repo/cookbooks/starter/files/
    creating: chef-repo/cookbooks/starter/files/default/
  inflating: chef-repo/cookbooks/starter/files/default/sample.txt
  inflating: chef-repo/cookbooks/starter/metadata.rb
    creating: chef-repo/cookbooks/starter/attributes/
  inflating: chef-repo/cookbooks/starter/attributes/default.rb
    creating: chef-repo/cookbooks/starter/templates/
  inflating: chef-repo/cookbooks/starter/templates/default/
  inflating: chef-repo/cookbooks/starter/templates/default/sample.erb
    creating: chef-repo/cookbooks/starter/recipes/
  inflating: chef-repo/cookbooks/starter/recipes/default.rb
  inflating: chef-repo/README.md
  inflating: chef-repo/.gitignore
    creating: chef-repo/.chef/
    creating: chef-repo/roles/
  inflating: chef-repo/.chef/knife.rb
  inflating: chef-repo/roles/starter.rb
  inflating: chef-repo/.chef/saurabh010.pem
```

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

Execute this:

```
1 mv starter /root/chef-repo/cookbook
```

Chef Cookbooks are available in the Cookbook Super Market, 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:

```
1 cd chef-repo
2 knife cookbook site download learn_chef_httpd
```

There is Tar ball downloaded for the Apache Cookbook. Now, we need to extract the contents from this downloaded Tar file. For that, I will use tar command.

```
1 tar -xvf learn_chef_httpd-0.2.0.tar.gz
```

```
[root@Workstation chef-repo]# tar -xvf learn_chef_httpd-0.2.0.tar.gz
learn_chef_httpd/
learn_chef_httpd/.kitchen.yml
learn_chef_httpd/Berksfile
learn_chef_httpd/chefignore
learn_chef_httpd/metadata.json
learn_chef_httpd/metadata.rb
learn_chef_httpd/README.md
learn_chef_httpd/recipes/
learn_chef_httpd/templates/
learn_chef_httpd/templates/default/
learn_chef_httpd/templates/default/index.html.erb
learn_chef_httpd/recipes/default.rb
[root@Workstation chef-repo]#
```

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 recipes folder.

Execute this:

```
1 cd /root/chef-repo/learn_chef_httpd/recipes
2 cat default.rb
```

```
[root@Workstation recipes]# cat default.rb
#
# Cookbook Name:: learn_chef_httpd
# Recipe:: default
# Copyright (C) 2014
#
#
package 'httpd'

service 'httpd' do
  action [:enable, :start]
end

template '/var/www/html/index.html' do
  source 'index.html.erb'
end

service 'iptables' do
  action :stop
end
[root@Workstation recipes]#
```

Now, I will just upload this cookbook to my Chef Server as it looks perfect to me.

Step 8: Upload Cookbook to the Chef Server.

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:**

```
1 mv /root/chef-repo/learn_chef_httpd /root/chef-repo/cookbooks
```

Now move to this cookbooks directory.

Execute this:

```
1 cd cookbooks
```

Now in this directory, execute the below command to upload the Apache Cookbook:

Execute this:

```
1 knife cookbook upload learn_chef_httpd
```

```
[root@Workstation cookbooks]# knife cookbook upload learn_chef_httpd
Uploading learn_chef_httpd [0.2.0]
Uploaded 1 cookbook.
[root@Workstation cookbooks]#
```

Verify the Cookbook from the Chef Server Management console. In the policy section, you will find the Cookbook that you have uploaded. Refer the screenshot below:



CHEF MANAGE			
	Nodes	Reports	Policy Administration
	Feedback   Organization: educ   Signed in as Saurabh   0 0		
> Cookbooks	Showing All Cookbooks		
Roles	Cookbook	Current Version	
Data Bags	learn_chef_apache2	0.3.0	
Environments	learn_chef_httpd	0.2.0	
Clients			

Now our final step is to add Chef Node. I have setup a Workstation, a Chef Server and now I need to add my Clients to the Chef Server for automation.

Step 9: Adding Chef Node to the Chef Server.

For demonstration purpose I will use one CentOS machine as Chef Node. There can be hundreds of Nodes connected to one 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:

```
1 ifconfig
```

```
[root@ChefNode ~]# ifconfig
eth1      Link encap:Ethernet  HWaddr 08:00:27:BA:23:1B
          inet addr:10.0.2.16  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::a00:27ff:feba:231b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:62 errors:0 dropped:0 overruns:0 frame:0
          TX packets:66 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7822 (7.6 KiB)  TX bytes:5885 (5.7 KiB)

eth2      Link encap:Ethernet  HWaddr 08:00:27:85:36:02
          inet addr:192.168.56.102  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe85:3602/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1909 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:254899 (248.9 KiB)  TX bytes:8260 (8.0 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
```

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:

```
1 knife bootstrap 192.168.56.102 --ssh-user root --ssh-password edureka --node-name chefNode
```

```
[root@Workstation cookbooks]# knife bootstrap 192.168.56.102 --ssh-user root --ssh-password edureka --node-name chefNode
Node chefNode exists, overwrite it? (Y/N) y
Client chefNode exists, overwrite it? (Y/N) y
Creating new client for chefNode
Creating new node for chefNode
Connecting to 192.168.56.102
192.168.56.102 ----> Installing Chef Omnibus (-v 12)
192.168.56.102 downloading https://omnitruck-direct.chef.io/chef/install.sh
192.168.56.102 to file /tmp/install.sh.5743/install.sh
192.168.56.102 trying wget...
192.168.56.102 el 6 x86_64
192.168.56.102 Getting information for chef stable 12 for el...
192.168.56.102 downloading https://omnitruck-direct.chef.io/stable/chef/metadata?v=12&p=el&pv=6&m=x86_64
192.168.56.102 to file /tmp/install.sh.5754/metadata.txt
192.168.56.102 trying wget...
192.168.56.102 sha1 abe23132483c1ff015148ad72becf092c0fac428
192.168.56.102 sha256 bca097d9107bd4b20df88045b676ald032b68a8b66bcf751dabc2e58715ae0ae
192.168.56.102 url https://packages.chef.io/files/stable/chef/12.16.42/el/6/chef-12.16.42-1.el6.x86_64.rpm
192.168.56.102 version 12.16.42
192.168.56.102 downloaded metadata file looks valid...
192.168.56.102 downloading https://packages.chef.io/files/stable/chef/12.16.42/el/6/chef-12.16.42-1.el6.x86_64.r
```



```

192.168.56.102 to file /tmp/install.sh.5754/chef-12.16.42-1.el6.x86_64.rpm
192.168.56.102 trying wget...
192.168.56.102 Comparing checksum with sha256sum...
192.168.56.102 Installing chef 12
192.168.56.102 installing with rpm...
192.168.56.102 warning: /tmp/install.sh.5754/chef-12.16.42-1.el6.x86_64.rpm: Header V4 DSA/SHA1 Signature, key ID
0 83ef826a: NOKEY
192.168.56.102 Preparing... ##### [100%]
192.168.56.102 1:chef ##### [100%]
192.168.56.102 Thank you for installing Chef!
192.168.56.102 Starting the first Chef Client run...
192.168.56.102 Starting Chef Client, version 12.16.42
192.168.56.102 resolving cookbooks for run list: []
192.168.56.102 Synchronizing Cookbooks:
192.168.56.102 Installing Cookbook Gems:
192.168.56.102 Compiling Cookbooks...
192.168.56.102 [2016-11-28T11:13:27+00:00] WARN: Node chefNode has an empty run list.
192.168.56.102 Converging 0 resources
192.168.56.102
192.168.56.102 Running handlers:
192.168.56.102 Running handlers complete
192.168.56.102 Chef Client finished, 0/0 resources updated in 49 seconds

```

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:

```
1 Knife node list
```

```

[root@Workstation cookbooks]# knife node list
chefNode
[root@Workstation cookbooks]#

```

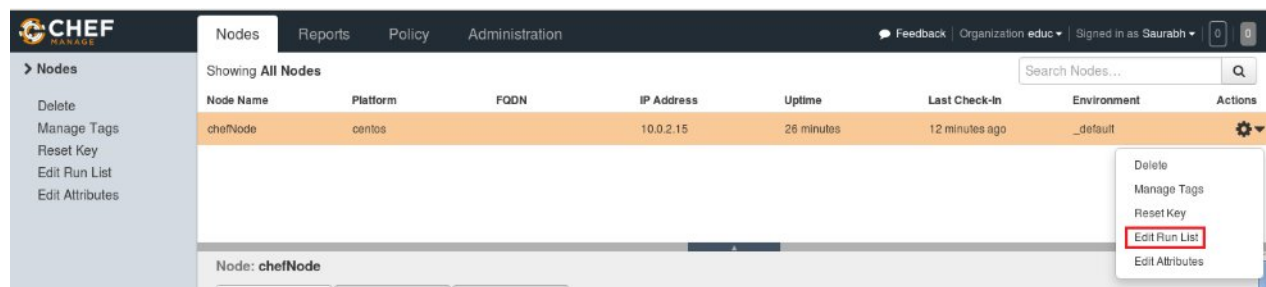
You can also verify from the Chef Server. Go to the nodes tab in your Server Management Console, here you will notice that the node that you have added is present. Refer the screenshot below.



Node Name	Platform	FQDN	IP Address	Uptime	Last Check-In	Environment	Actions
chefNode	centos		10.0.2.15	26 minutes	12 minutes ago	_default	

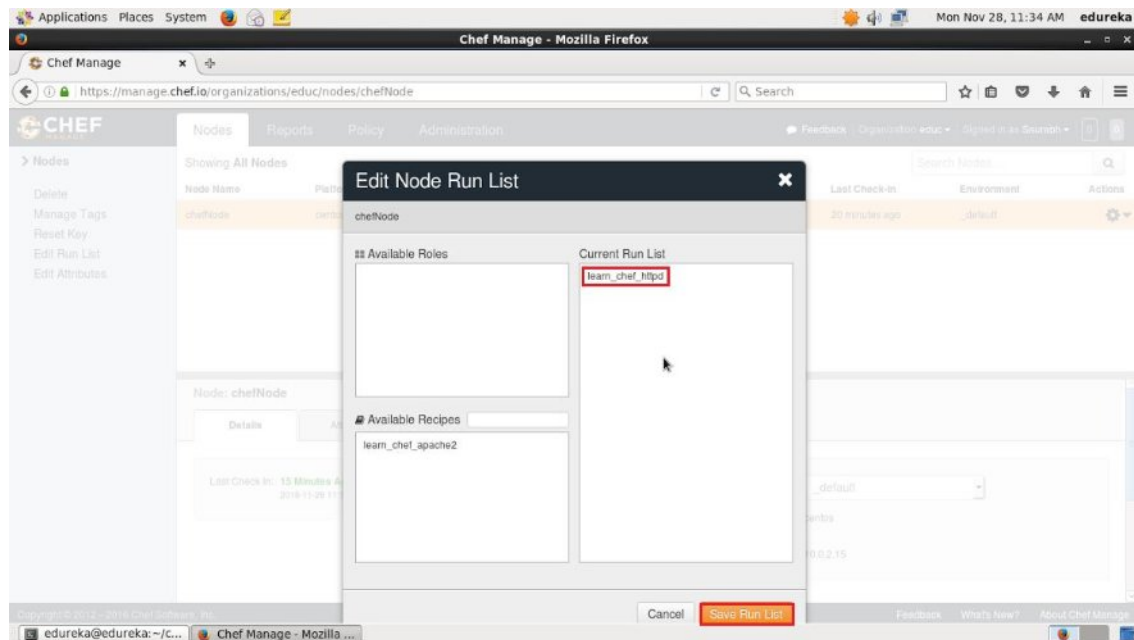
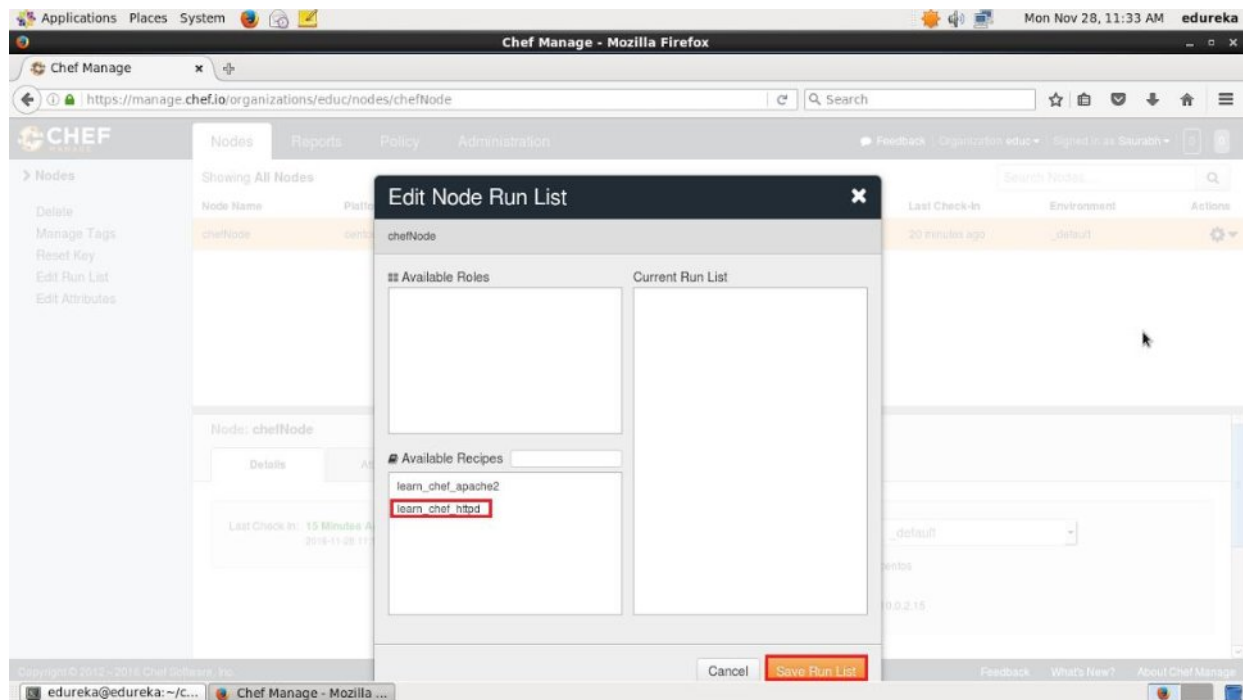
## Step 10: Manage Node Run List

Let's see how we can add a Cookbook to the Node and manage its Run list from the Chef Server. As you can see in the screenshot below, click the Actions tab and select the Edit Run list option to manage the Run list.



Node Name	Platform	FQDN	IP Address	Uptime	Last Check-In	Environment	Actions
chefNode	centos		10.0.2.15	26 minutes	12 minutes ago	_default	<ul style="list-style-type: none"> <li>Delete</li> <li>Manage Tags</li> <li>Reset Key</li> <li><b>Edit Run List</b></li> <li>Edit Attributes</li> </ul>

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.

Execute this:

```
1 chef-client
[root@ChefNode ~]# chef-client
Starting Chef Client, version 12.16.42
resolving cookbooks for run list: ["learn_chef_httpd"]
Synchronizing Cookbooks:
- learn_chef_httpd (0.2.0)
Installing Cookbook Gems:
Compiling Cookbooks...
Converging 4 resources
Recipe: learn_chef_httpd::default
* yum_package[httpd] action install (up to date)
* service[httpd] action enable
- enable service service[httpd]
* service[httpd] action start
- start service service[httpd]
* template[/var/www/html/index.html] action create
- create new file /var/www/html/index.html
- update content in file /var/www/html/index.html from none to ef4ffd
--- /var/www/html/index.html      2016-11-28 11:47:17.414304893 +0000
+++ /var/www/html/.chef-index20161128-6284-hzok74.html      2016-11-28 11:47:17.414304893 +0000
@@ -1 +1,6 @@
+<html>
```

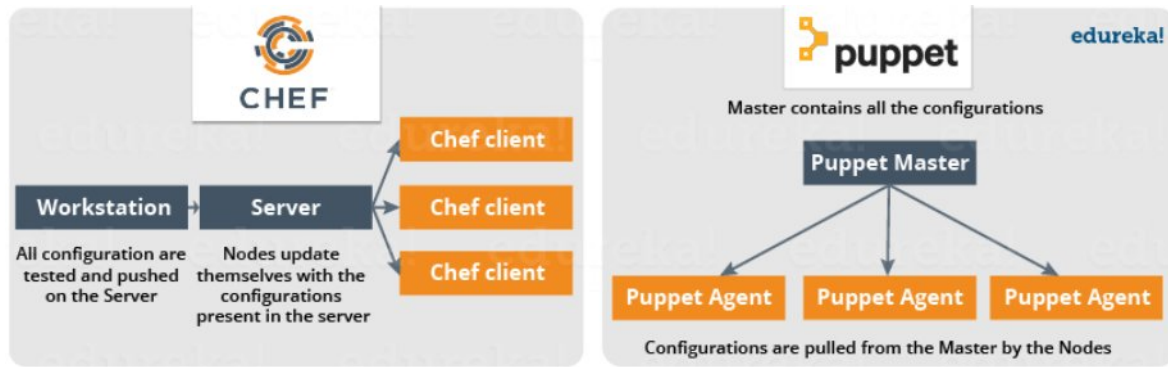
I hope you enjoyed this Chef Tutorial and learned how Chef can be used to configure hundreds of Nodes. Chef is playing a vital role in many organizations to achieve DevOps. With Chef organizations are releasing applications more frequently and reliably

## What Is Chef?

Chef is an automation tool that provides a way to define infrastructure as code. Infrastructure as code (IAC) simply means that managing infrastructure by writing code (Automating infrastructure) rather than using manual processes. It can also be termed as programmable infrastructure. Chef uses a pure-Ruby, domain-specific language (DSL) for writing system configurations. Below are the types of automation done by Chef, irrespective of the size of infrastructure:

- Infrastructure configuration
- Application deployment
- Configurations are managed across your network

Like [Puppet](#) which has a Master-Slave architecture even Chef has a Client-Server architecture. But Chef has an extra component called Workstation. I will talk about workstation in my next blog. Refer the diagram below:



In Chef, Nodes are dynamically updated with the configurations in the Server. This is called Pull Configuration which means that we don't need to execute even a single command on the Chef server to push the configuration on the nodes, nodes will automatically update themselves with the configurations present in the Server. My next blog on Chef Tutorial will explain the Chef architecture along with all the Chef components in detail.

Now, let us look at reasons behind the popularity of Chef.

## What Is Chef – Chef Key Metrics

- Chef supports multiple platforms like AIX, RHEL/CentOS, FreeBSD, OS X, Solaris, Microsoft Windows and Ubuntu. Additional client platforms include Arch Linux, Debian and Fedora.
- Chef can be integrated with cloud-based platforms such as Internap, Amazon EC2, Google Cloud Platform, OpenStack, SoftLayer, Microsoft Azure and Rackspace to automatically provision and configure new machines.
- Chef has an active, smart and fast growing community support.
- Because of Chef's maturity and flexibility, it is being used by giants like Mozilla, Expedia, Facebook, HP Public Cloud, Prezi, Xero, Ancestry.com, Rackspace, Get Satisfaction, IGN, Marshall University, Socrata, University of Minnesota, Wharton School of the University of Pennsylvania, Bonobos, Splunk, Citi, DueDil, Disney, and Cheezburger.

According to Phil Dibowitz, Production Engineer, Facebook

"There are three dimensions of scale we generally look at for infrastructure — the number of servers, the volume of different configurations across those systems, and the number of people required to maintain those configurations. Chef provided an automation solution flexible enough to bend to our scale dynamics without requiring us to change our workflow."

Without a doubt Chef is one of the most famous Configuration Management tools and is closely competing with [Puppet](#). But, before diving deep into "What is Chef", it's only fair that I first explain what is Configuration Management and why it is important.

## Configuration Management

Don't worry, there won't be any heavy definition of Configuration Management in this blog :)

Let us understand Configuration Management this way – suppose you have to deploy a software on top of hundreds of systems. This software can be an operating system or a code or it can be an update of an existing software. You can do this task manually, but what happens if you have to finish this task overnight because tomorrow might be a Big Billion Day sale in the company or some Mega Sale etc. in which heavy traffic is expected. Even if you were able to do this manually there is a high possibility of multiple errors on your big day. What if the software you updated on hundreds of systems is not working, then how will you revert back to the previous stable version, will you be able to do this task manually? AF-course not!

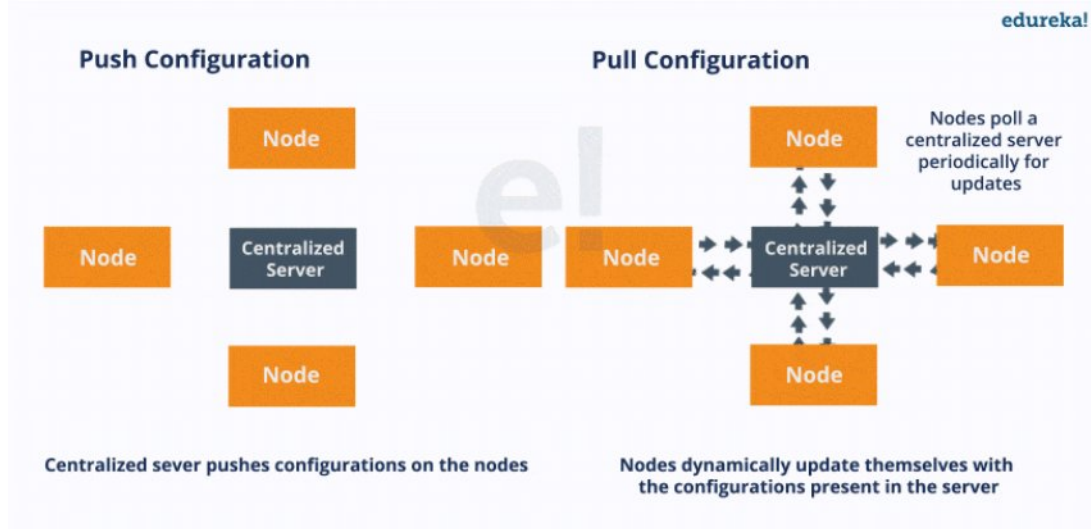
To solve this problem, Configuration Management was introduced. By using Configuration Management tools like Chef, Puppet, etc. you can automate this task. All you have to do is to specify the configurations in one centralized server and accordingly all the nodes will be configured. It allows access to an accurate historical record of system state for project management and audit purposes. So basically, we need to specify the configurations once on the central server and replicate that on thousands of nodes. Configuration Management helps in performing the below tasks in a very structured and easy way:

- Figuring out which components to change when requirements change.
- Redoing an implementation because the requirements have changed since the last implementation.
- Reverting to a previous version of the component if you have replaced with a new but flawed version.
- Replacing the wrong component because you couldn't accurately determine which component was supposed to be replaced.

There are broadly two ways to manage your configurations namely Push and Pull configurations.

- Pull Configuration: In this type of Configuration Management, the nodes poll a centralized server periodically for updates. These nodes are dynamically configured so basically they are pulling configurations from the centralized server. Pull configuration is used by tools like Chef, Puppet etc.
- Push Configuration: In this type of Configuration Management, the centralized Server pushes the configurations to the nodes. Unlike Pull Configuration, there are certain commands that have to be executed in the centralized server in order

to configure the nodes. Push Configuration is used by tools like Ansible.



[Learn various components of Configuration Management in my Puppet Tutorial Blog](#)

Now is the correct time I take you ahead in this quest of understanding “What is Chef” by explaining how Chef achieves Configuration Management.

### What Is Chef – Configuration Management With Chef

We have understood what is Chef, now I will explain you how Chef achieves Configuration Management with a use-case. Gannett is a publicly traded American media holding company. It is the largest U.S. newspaper publisher as measured by total daily circulation.

Gannett’s traditional deployment workflow was characterized by multiple handoffs and manual tests. Let us see what were the problems they faced with this process:

- Maintaining accurate, repeatable builds was difficult.
- There were many build failures and tests were often running in the wrong environments.
- Deployment and provisioning times could range from a few days to several weeks.
- Operations team didn’t have access to the cloud or development environments.
- Every group used its own tool-set, and there was no accountability to finance or security. No one knew how much an application actually cost. Security had no way to audit the software stacks.

Gannett was ready for the change. Developers wanted to deploy their applications quickly. Operations wanted a stable infrastructure where they could build and deploy in a repeatable way. Finance wanted to know the true cost of an application. Security wanted to view and audit all stacks and to be able to track changes.

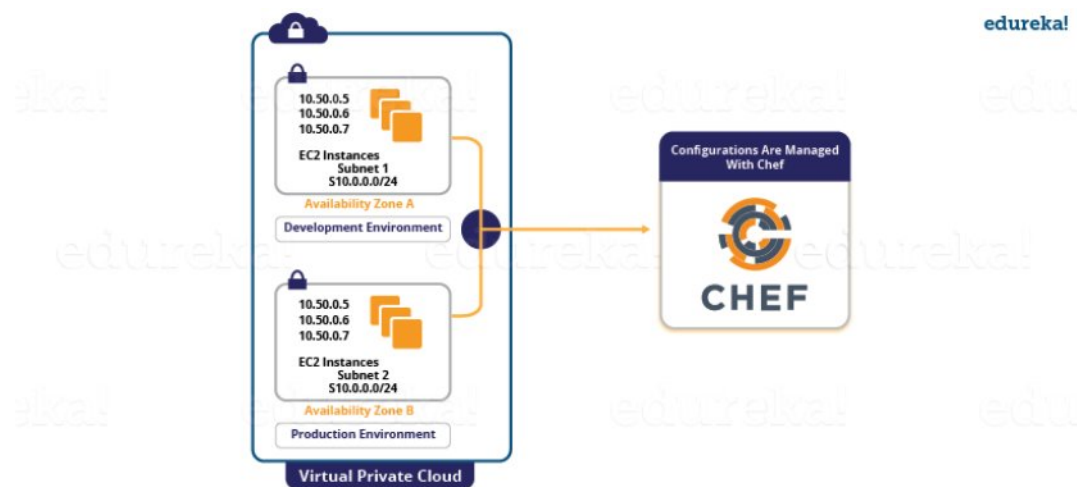
Gannett saw that cloud as a service offered many advantages. Developers had access to standardized resources. It was easier to handle peaky traffic because of cloud’s compute-on-demand model, and handoffs were minimized.



Chef allows you to dynamically provision and de-provision your infrastructure on demand to keep up with peaks in usage and traffic. It enables new services and features to be deployed and updated more frequently, with little risk of downtime. With Chef, you can take advantage of all the flexibility and cost savings that cloud offers.

Let us see what were the functions performed by Chef at Gannett:

- Gannett started building VPC (Virtual Private Cloud) for development environment that would mimic the production. None of the tools that they were already using were appropriate. But they found that Chef worked well with the cloud and both Linux and Windows environment. They used Chef to build a development environment that perfectly matched production environment.
- For an application to move into the VPC, it had to be provisioned and deployed with Chef.
- Security would be involved early on and would manage the mandatory controls for access to Chef and for maintaining system security standards.



Now is the time to understand what were the results of this process:

- Gannett's deployment became quicker and more reliable. Application provisioning and deployment, which once took weeks, after using Chef it took minutes.
- All new applications were deployed on the cloud with Chef. These applications were deployed to all environments in the same way that they were deployed to production. Also, testing occurred in each environment, so that the deployments were reliable.
- All infrastructure was treated as code, which greatly increases visibility into any changes that occurred. Development, Operations, Security and Finance all were benefited from this.

# INTERVIEW QUS

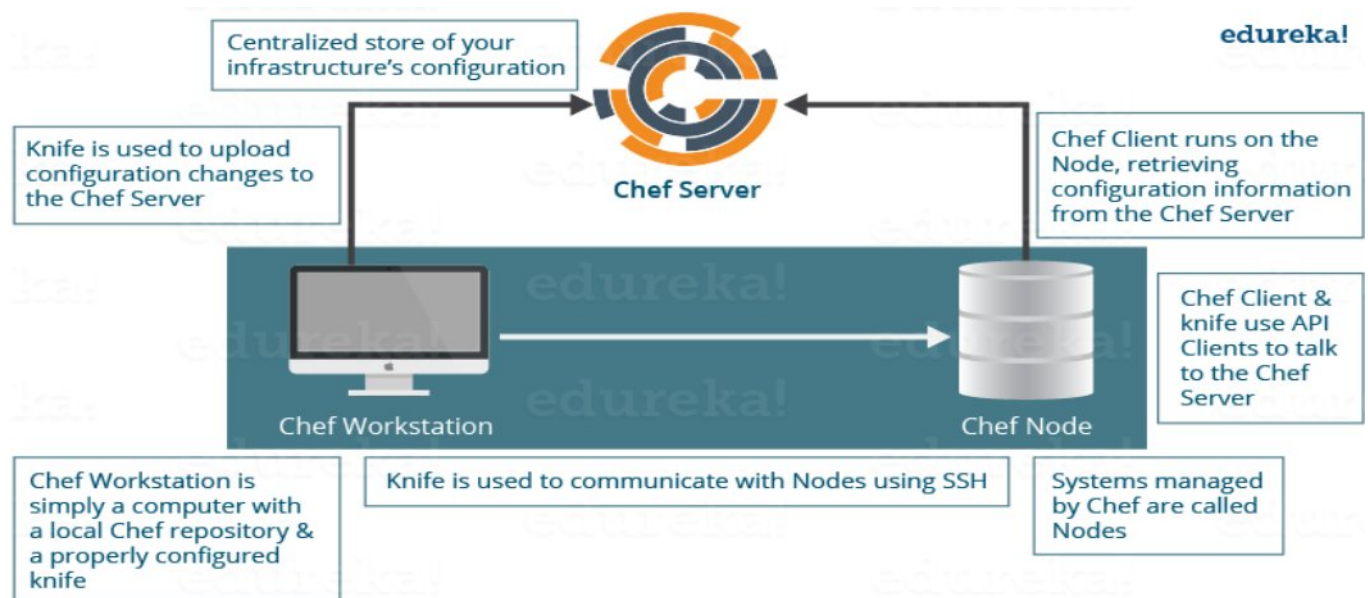
Q1. What is Chef?

Begin this answer by defining Chef.

It is a powerful automation platform that provides a way to transform infrastructure into code. Chef is a tool for which you write scripts that are used to automate processes. What processes? Pretty much anything related to IT.

Now you can explain the architecture of Chef, it consists of:

- **Chef Server:** The Chef Server is the central store of your infrastructure's configuration data. The Chef Server stores the data necessary to configure your nodes and provides search, a powerful tool that allows you to dynamically drive node configuration based on data.
- **Chef Node:** A Node is any host that is configured using Chef-client. Chef-client runs on your nodes, contacting the Chef Server for the information necessary to configure the node. Since a Node is a machine that runs the Chef-client software, nodes are sometimes referred to as "clients".
- **Chef Workstation:** A Chef Workstation is the host you use to modify your cookbooks and other configuration data. All the configurations are first tested in the Chef Workstation and then it is forwarded to the Chef Server.



Q1. What is a Resource in Chef?

My suggestion is to first define Resource.

A Resource represents a piece of infrastructure and its desired state, such as a package that should be installed, a service that should be running, or a file that should be generated. A block of Resource can be considered as a Recipe.

Now you should explain about the functions of Resource for that include the following points:

- 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.

Remember, you have mentioned the word Recipe in your previous answer, so the next question in this Chef interview questions blog has to be related to Recipe.

Q2. What is a Recipe in Chef?

Here also I will suggest you to use the above mentioned flow, first define Recipe.

A Recipe is a collection of Resources that describes a particular configuration or policy. A Recipe describes everything that is required to configure part of a system.

Now after the definition I will explain the functions of Recipes by including the following points:

- Install and configure software components.
- Manage files.
- Deploy applications.
- Execute other Recipes.

Q3. What is a Node in Chef?

This will be probably the easiest question you can encounter answer this by saying:

A Node represents a server and is typically a virtual machine, container instance, or physical server – basically any compute resource in your infrastructure that is managed by Chef.

Q4. How does a Cookbook differ from a Recipe in Chef?

The answer to this is pretty direct My suggestion is to simply tell:

A Recipe is a collection of Resources, and primarily configures a software package or some piece of infrastructure. A Cookbook groups together Recipes and other information in a way that is more manageable than having just Recipes alone.

Now the following set of Chef interview questions are to test your experience with Chef:

Q5. What happens when you don't specify a Resource's action in Chef?

My suggestion is to first give a direct answer.

When you don't specify a resource's action, Chef applies the default action.

Now explain this with an example, the below resource:

```
1 file 'C:\Users\Administrator\chef-repo\settings.ini' do
2   content 'greeting=hello world'
```

```
3     end
```

is same as the below resource:

```
1   file 'C:\Users\Administrator\chef-repo\settings.ini' do
2     action :create
3     content 'greeting=hello world'
4   end
```

because: create is the file Resource's default action.

Q6. Are these two Chef recipes the same?

```
1   package 'httpd'
2   service 'httpd' do
3     action [:enable, :start]
4   end
```

**&&**

```
1   service 'httpd' do
2     action [:enable, :start]
3   end
4   package 'httpd'
```

No, they are not. Remember that Chef applies resources in the order they appear. So the first Recipe ensures that the httpd package is installed and then configures the service. The second Recipe configures the service and then ensures the package is installed.

Q7. 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   service 'httpd' do
2     action [:stop, :disable]
3   end
```

Q8. How does Chef-apply differ from Chef-client?

I suggest you to follow the below mentioned flow to answer this question:

Chef-apply is an executable program that runs a single Recipe from the command line. It is a part of the Chef development kit and a great way to explore resources.

Syntax for Chef-apply is:

```
1 chef-apply name_of_recipe.rb
```

Chef-client applies a Cookbook. It is used for production purposes where you typically run Chef-client to apply one or more cookbooks.

Q9. What is run-list in Chef?

My advise is to first explain what is the use of run-list

run-list lets you specify which Recipes to run, and the order in which to run them. The run-list is important when you have multiple Cookbooks, and the order in which they run matters.

Depending on the discussion if you think more explanation is required just mention the below points

A run-list is:

- An ordered list of roles and/or recipes that are run in the exact order defined in the run-list; if a recipe appears more than once in the run-list, the chef-client will not run it twice.
- Always specific to the node on which it runs; nodes may have a run-list that is identical to the run-list used by other nodes.
- Stored as part of the node object on the Chef server.
- Maintained using knife, and then uploaded from the workstation to the Chef server, or is maintained using the Chef management console.

Q10. 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.

Q11. How do you apply an updated Cookbook to your node in Chef?

There are three ways to apply an updated Cookbook to a node you can mention all or any one, I will suggest you to mention all three:

- Run knife ssh from your workstation.
- SSH directly into your server and run chef-client.
- You can also run chef-client as a daemon, or service, to check in with the Chef server on a regular interval, say every 15 or 30 minutes.

Q12. What is the role of Starter Kit in Chef?

Begin this answer by mentioning the functions of Starter Kit.

Starter Kit will create the necessary configuration files like chef directory, knife.rb, the ORGANIZATION-validator.pem, and USER.pem files etc. with the correct information that is required to interact with the Chef server.

Now tell how to use Starter Kit, you can simply download the starter kit and then move it to the desired location on your workstation.

Q13. 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".

Q14. What would you set your cookbook's version to once it is ready to use in production?

According to Semantic Versioning, you should set your cookbook's version number to 1.0.0 once it is ready to use in production.

Q15. What is the value of local development using Test Kitchen in Chef?

I will mention the below points, this will give the interviewer a clear picture of your understanding of Test Kitchen.

- Test Kitchen enables you to use a variety of virtualization providers that create virtual machine or container instances locally on your workstation or in the cloud.
- It enables you to run your cookbooks on servers that resemble those that you use in production.
- It speeds up the development cycle by automatically provisioning and tearing down temporary instances, resolving cookbook dependencies, and applying your cookbooks to your instances.

Q16. 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>.