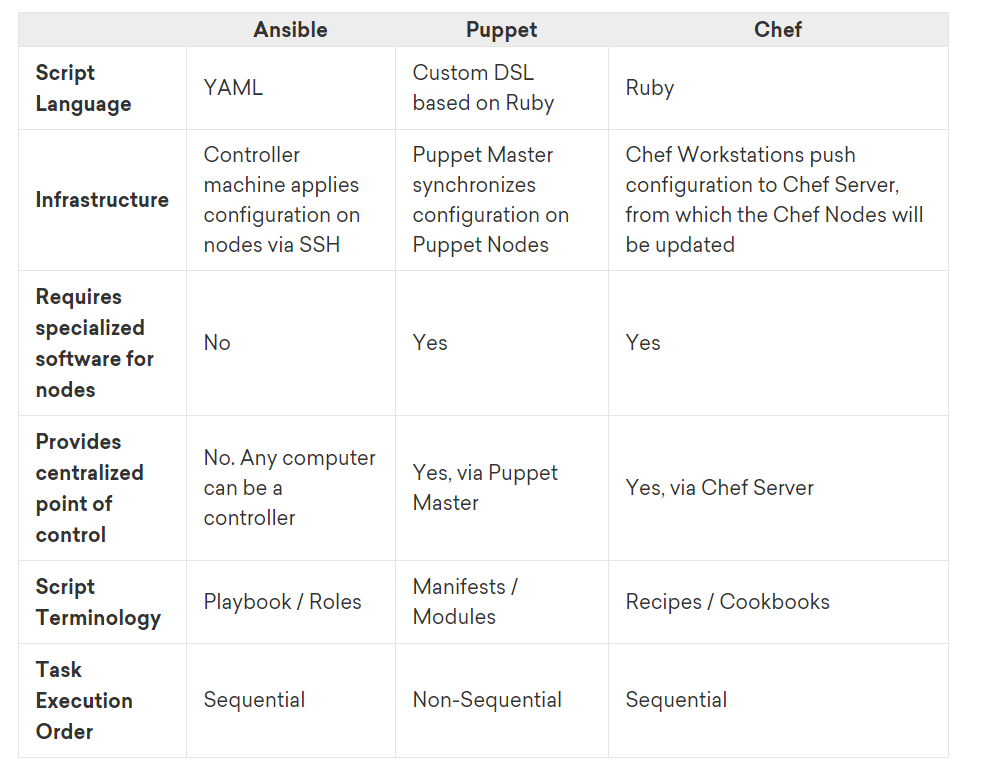
# What is Configuration management

This is all about trying to ensure that the files and software you are expecting to be on a machine are present, configured correctly, and working as intended.

When you have only a single machine this is fairly simple. When you have five or ten servers, it is still possible to do this manually, but it may take all day. However, when your infrastructure scales up into the thousands we need a better way of doing things.

2 Good URL for Interview –

1. <https://www.digitalocean.com/community/tutorials/an-introduction-to-configuration-management>
2. <https://www.upguard.com/blog/5-configuration-management-boss>



# What is Chef? A primer for DevOps newbies

Chef is an open source cloud configuration management and deployment application.

Chef enables programmers and system administrators to work together. Instead of developers writing applications and then waiting on **Ops staff to figure out how to deploy the software, Chef helps both communities.**

**Chef moves the process from a tedious release cycle to**[**a continuous delivery model**](https://insights.hpe.com/articles/the-basics-explaining-kubernetes-mesosphere-and-docker-swarm-1702.html)**by enabling an effective and automated workflow—a central goal of the DevOps movement.**

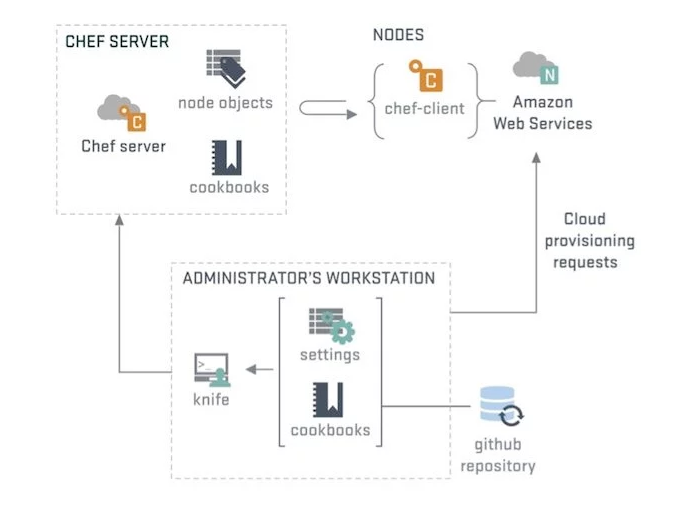
Net result? You get **more work from your existing IT crew**, and everything in IT moves faster with fewer errors. **When defects do crop up (because this is the real world), they get fixed sooner**.

Ultimately, Chef is a framework to[**automate and manage infrastructure and applications**](https://insights.hpe.com/articles/devops-lessons-in-software-infrastructure-and-business-success-1702.html). Specifically, Chef translates system administration tasks into **reusable definitions,** known as cookbooks (multiple recipes), recipes (multiple resources) and resources.

# Chef Architecture

A typical Chef deployment comprises three core components:

1. the Chef server,
2. workstations, and
3. Chef nodes or clients



# Chef basics

You can use Chef in any **on-premise** **data center** or **cloud infrastructure**. It runs on several platforms, including Windows; enterprise Linux distributions; AIX; FreeBSD; Solaris; Cisco IO; and Nexus.

Chef supports cloud platforms including **Amazon Web Services (AWS), Google Cloud Platform,** OpenStack, IBM Bluemix, HPE Cloud, Microsoft Azure, VMware vRealize Automation, and Rackspace.

## Recipes and cookbooks **-** These are the heart of the configuration management. They are written using the Ruby programming language, however, **the domain specific language (DSL) used by Chef** is designed to be able to be **understood by everyone**. As the configuration is just code it can be tested and it can be version controlled. This means that there is less downtime, more reliable services and less stressed people on both the dev and ops sides.

The recipes and cookbooks capture the process, and are tested using tools like **cookstyle**, [**Test Kitchen**](https://docs.chef.io/kitchen.html)**, [ChefSpec](https://docs.chef.io/chefspec.html" \t "_blank), and [Foodcritic](http://www.foodcritic.io/" \t "_blank).**

Once they’re working properly, the recipes and cookbooks are deployed to the Chef server using the [knife](https://docs.chef.io/knife.html) and [chef](https://docs.chef.io/ctl_chef.html) command-line tools.

# Chef Workstation Creation

Chef development kit allows you to write and manage your chef infrastructure from any machine and any operating system.

Install ChefDK



1. Create a EC2 Instance
2. Download the latest chefDK (Chef Development KIT) –

**wget https://packages.chef.io/files/stable/chefdk/4.7.73/el/8/chefdk-4.7.73-1.el7.x86\_64.rpm**

1. [root@ip-172-31-0-38 ~]# **rpm -Uvh chefdk-4.6.35-1.el7.x86\_64.rpm**

warning: chefdk-4.6.35-1.el7.x86\_64.rpm.1: Header V4 DSA/SHA1 Signature, key ID 83ef826a: NOKEY

Preparing... ################################# [100%]

package chefdk-4.6.35-1.el7.x86\_64 is already installed

1. Check the version

[root@ip-172-31-0-38 ~]# chef --version

ChefDK version: 4.6.35

Chef Infra Client version: 15.5.17

Chef InSpec version: 4.18.39

Test Kitchen version: 2.3.4

Foodcritic version: 16.2.0

Cookstyle version: 5.13.7

Congratulations !!

# Exercise: Your First Chef recipe

Create a recipe **hello.rb** and then run it

file 'hello.txt' do

content 'Hello, world!'

action :create # Default action

end

**Zero Mode or Solo mode** ->

chef-client **--local-mode** hello.rb or

chef-client **-z** hello.rb

# About Resources

<https://docs.chef.io/resources/>

A 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

## Resource Syntax

A resource is a Ruby block with four components: a type, a name, one (or more) properties (with values), and one (or more) actions. The syntax for a resource is like this:

type 'name' **do**

attribute 'value'

action :type\_of\_action

**end**

Every resource has its own set of actions and properties. Most properties have default values. Some properties are available to all resources, for example those used to send notifications to other resources and guards that help ensure that some resources are idempotent.

For example, a resource that is used to install a tar.gz package for version 1.16.1 may look something like this:

package 'tar' **do**

version '1.17.1'

action :remove

**end**

All actions have a default value. Only non-default behaviors of actions and properties need to be specified. For example, the **package** resource’s default action is :install and the name of the package defaults to the name of the resource. Therefore, it is possible to write a resource block that installs the latest tar.gz package like this:

package 'tar'

package ‘tree’

and a resource block that installs a tar.gz package for version 1.6.1 like this:

package 'tar' **do**

version '1.16.1'

**end**

In both cases, Chef Infra Client will use the default action (:install) to install the tar package.

### **Test and Repair** – This Example will show that How to use Chef to have desired state. We will change something **which is not as per desired state as** per chef recipe and then we will runit again and then chef will be used to repair it.



### Exercise 2-



# ~/setup.rb

package 'tree' do

action :install

end

package 'ntp'

file '/etc/motd' do

    content 'This server is the property of ...'

    action :create

    owner 'root'

    group 'root'

end

One more example

# ~/ setup.rb - Install package tree & ntp followed by file creation and ntp enable reboot and start.

package 'tree' do

    action :install

end

package 'ntp'

file '/etc/motd' do

    content 'This server is the property of ...'

end

service 'ntpd' do

    action [:enable, :start]

end

### Exercise 3 - Write a Apache recipe for below task –

ABC.com - Static Website

1. Install apache - package

2. Create a New Doc root - /var/www/html/abc.com - directory

3. copy the code from git dev branch to New Doc root - git

4. open httpd.conf and change the doc root to new one which is created 2nd step - remote\_file

5. restart and Enable the apache – service

#Install apache - package

package 'httpd' do

action :install

end

package 'git' do

action :install

end

#Create a New Doc root - /var/www/html/abc.com - directory

directory '/var/www/html/abc.com' do

owner 'root'

group 'root'

mode '0755'

action :create

end

#copy the code from git dev branch to New Doc root - git

git '/var/www/html/abc.com' do

repository 'https://github.com/mailrahulsre/abc.com.git'

revision 'dev'

action :sync

end

#open httpd.conf and change the doc root to new one which is created 2nd step - template

remote\_file '/etc/httpd/conf/httpd.conf' do

source 'file:///var/www/html/abc.com/httpd.conf'

mode '0755'

owner 'root'

group 'root'

end

#restart and Enable the apache - service

service 'httpd' do

action [ :enable, :start, :reload ]

end

## Create Chef Server Account –

<https://manage.chef.io/login> - go here and create

## Cookbook –

Chef -help

chef generate cookbook apache

chef-client -z --runlist 'recipe[test]'

chef generate cookbook first\_cookbook

where first\_cookbook is an arbitrary cookbook name.

1. Navigate to the first\_cookbook directory.
2. Update the cookbooks/first\_cookbook/recipes/default.rb recipe in the generated cookbook to contain:

file "*#{*ENV['HOME']*}*/test.txt" **do**

content 'This file was created by Chef Infra!'

**end**

1. Run Chef Infra Client using the default.rb recipe:

chef-client --local-mode --override-runlist first\_cookbook

This will create a file named test.txt at the home path on your machine. Open that file and it will say This file was created by Chef Infra!.

**Sample Recipe for Apache**

Chef config files to install Apache and add a hello world html page

