Chef Fundamentals

Presented to Unix Capability

1 June 2017

Unleash the Next

Chef Fundamentals

Objective

This training is intend to equip the participant with an understanding of Chef Fundamentals.

The session will cover following sections

- What is Chef
- Core Components of Chef
- Installing Chef Dev Kit
- Sample programs(recipes)

What is Chef

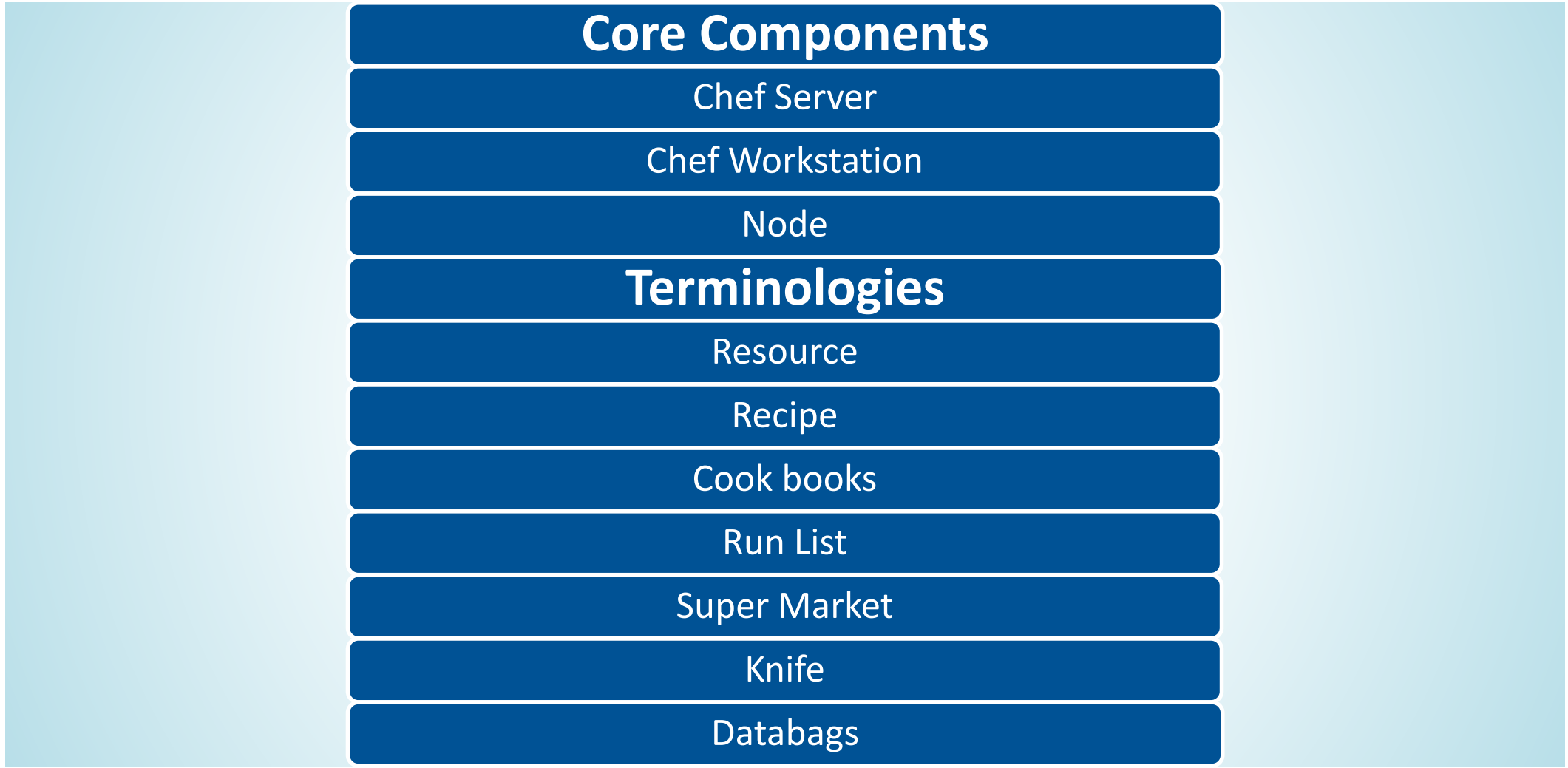
- Chef is an automation platform that transforms infrastructure into code. Whether you're operating in the cloud, on-premises, or in a hybrid environment, Chef automates how infrastructure is configured, deployed, and managed across your network, no matter its size.
- Chef has a complete automation solution for both infrastructure and applications that takes you all the way from development to production
- It speed up app deployment, turns infra into code, which is versionable, testable and repeatable.
- It can describe your system resources, customize to your needs reducing complexity of management.

What is Chef- cont'd

- This diagram shows how you develop, test, and deploy your Chef code.

- Describe system resources, interacting with state of machines through ruby and chef clients and distributing changes to infra through chef server.

Chef Components/Architecture



Chef Components/Architecture – Cont'd

❑ **Chef Server**

Chef server is the central element or brain of a Chef framework. Chef server works as a central repository or hub that contains configuration policies for nodes and cookbooks and has detailed information of registered components. Other components of a Chef framework use it for automated configuration and deployment.

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 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.

Chef Components/Architecture – Cont'd

❑Workstations

Workstations are static computers or virtual servers where all codes are created or changed.

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

Chef Components/Architecture – Cont'd

❑**Nodes**

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.

Nodes are kept up-to-date through the use of chef-client, which runs as a convergence between the node and chef server.

Chef Terminologies

❑Resource

- Represents a piece of the system and its desired state
- Eg ; a package that needs to be installed
- A service that should be running
- A file that should be generated
- A cron job that should be configured
- A user that should be managed
- They are the fundamental building block of Chef Configuration

Chef Terminologies-Cont'd

❑Recipe

- Resources are gathered in to recipes written in ruby language.
- Recipes ensures system is in desired state
- Recipes are the fundamental part of cookbooks. Recipe contain information in regards to everything that needs to be run, changed, or created on a node.
- Recipe can;
- Install and configure software components
- Manage files
- Deploy applications
- Execute other recipes

Chef Terminologies-Cont'd

❑Cook books

- Cookbooks are group of recipe for a node which is fundamental unit of configuration and policy distribution
- For eg, it will have all the steps required to install a software and all the components that are required to install that software.
- It will have all the recipes that specify the resources to use and the order in which those needs to be applied
- 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.
- 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.

Chef Terminologies-Cont'd

❑Run List

- It's a subset of policies that should be applied to a node. Chef-client will be looking for this run list to apply the policy on the node.
- A run list is a combined list of recipes to be applied to a given node in a certain order. A run list can comprise of zero or more roles or recipes , and order is important as the run list's items are executed in the order specified. Therefore, if one recipe is dependent upon the execution of another, you need to ensure that they run in the correct order

Chef Terminologies-Cont'd

❑Super Market

- 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 and is located at https://supermarket.chef.io.
- 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.

Chef Terminologies-Cont'd

❑Knife

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

- knife is a command-line tool that provides an interface between a local chef-repo and the Chef server. knife helps users to manage:

- Nodes
- Cookbooks and recipes
- Roles, Environments, and Data Bags
- Resources within various cloud environments
- The installation of the chef-client onto nodes
- Searching of indexed data on the Chef server

Chef Terminologies-Cont'd

❑Databags

In addition to storing configuration data with a node or a role, Chef has the notion of infrastructure-wide data storage. This data is accessible to any recipe through the data bags interface. Information about users, groups, firewall rules, internal IP address lists, software versions, and other data can be stored here. Data bags contain data about your infrastructure as a whole and are a good way to store any information your recipes need about system-wide configuration.

Installing Chef Dev Kit

➢ Download the rpm *chefdk-1.0.3-1.el6.x86_64.rpm* from the below link

https://downloads.chef.io/chefdk/1.0.3

➢ Install it on your work station

- rpm –ivh chefdk-1.0.3-1.el6.x86_64.rpm
- Run # chef –v to get the version details.
- root@localhost# chef -v
- Chef Development Kit Version: 1.0.3
- chef-client version: 12.16.42
- delivery version: master (83358fb62c0f711c70ad5a81030a6cae4017f103)
- berks version: 5.2.0
- kitchen version: 1.13.2

Sample programs(recipe)

- **Prepare your recipe**

```
[prasu@rhel ~]$ vim hello.rb
[prasu@rhel ~]$ cat hello.rb
file '/home/prasu/hello.txt' do
 content 'Hello World!'
end
```

- `[prasu@rhel ~]$ chef-client --local-mode hello.rb`

```
 [2017-05-26T15:29:48-07:00] WARN: No config file found or specified on command line, using command line options.
[2017-05-26T15:29:48-07:00] WARN: No cookbooks directory found at or above current directory.  Assuming /home/prasu.
Starting Chef Client, version 12.16.42
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks...
[2017-05-26T15:29:59-07:00] WARN: Node rhel has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/prasu/hello.rb
  * file[/home/prasu/hello.txt] action create
    - create new file /home/prasu/hello.txt
    - update content in file /home/prasu/hello.txt from none to 7f83b1
    --- /home/prasu/hello.txt   2017-05-26 15:29:59.263506456 -0700
    +++ /home/prasu/.chef-hello20170526-2512-sjf7s5.txt 2017-05-26 15:29:59.262506456 -0700
    @@ -1 +1,2 @@
    +Hello World!
    - restore selinux security context
 Running handlers:
Running handlers complete
Chef Client finished, 1/1 resources updated in 09 seconds
```

Understanding sample programs

- **[prasu@rhel ~]$ more hello.txt**

- **Hello World!**

```
[prasu@rhel ~]$ cat hello.rb
file '/home/prasu/hello.txt' do
 content 'Hello World!'
end
```

- It’s the type (File/Service/Package)

- It’s the desired state

Understanding sample programs

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

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

 file '/home/prasu/hello.txt' do
 action :delete
end
```

Understanding sample programs

- If no action is mentioned it will pick default action

For type file(Default is create), For pkg , default is install

For service, default is start

- We don’t need to specify action statements explicitly for default actions.

- If you change of content of hello file, when chef re-runs it corrects it, but if you change perm, it wont,

because a ownership change is not defined in its recipe

Understanding sample programs

```
[prasu@rhel ~]$ cat setup.rb
package 'ntp' do    >> pacakage ‘ntp’ is enough here as it will perform default action(Install)
 action :install
end

[prasu@rhel ~]$ chef-client --local-mode setup.rb
[2017-05-29T07:43:20-07:00] WARN: No config file found or specified on command line, using command line options.
[2017-05-29T07:43:20-07:00] WARN: No cookbooks directory found at or above current directory.  Assuming /home/prasu.
Starting Chef Client, version 12.16.42
resolving cookbooks for run list: []
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks...
[2017-05-29T07:43:24-07:00] WARN: Node rhel has an empty run list.
Converging 1 resources
Recipe: @recipe_files::/home/prasu/setup.rb
  * yum_package[ntp] action install (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 0/1 resources updated in 04 seconds
```

Understanding sample programs

- file ‘/etc/motd’ do
- content ‘This is Prasad’s computer, don’t disturb’
- action : create
- owner ‘root’
- group ‘root’
- end
- service ‘ntpd’ do
- action [:enable, :start]
- end

References

- https://learn.chef.io/modules/fundamentals-series/fundamentals-series-week-1#/
- https://docs.chef.io/
- https://www.linode.com/docs/applications/configuration-management/beginners-guide-chef

THANK YOU

Presentation By – Prasad K
Email ID : Prasad.K01@mphasis.com

About Mphasis

Mphasis an HP Company is a USD 1 billion global service provider, delivering technology-based solutions across industries, including Banking & Capital Markets, Insurance, Manufacturing, Communications, Media & Entertainment, Healthcare & Life Sciences, Transportation & Logistics, Retail & Consumer Goods, Energy & Utilities and Governments around the world. Mphasis' integrated service offerings in Applications, Infrastructure Services and Business Process Outsourcing help organizations adapt to changing market conditions and derive maximum value from IT investments. For more information about Mphasis, log on to www.mphasis.com