Chef

Module-1

Agenda

1. Introduction
2. Architecture
3. Hello World: Chef Server
4. Hello World: Chef Workstation
5. Hello World: Chef Node
6. Cookbooks
7. Recipes

Introduction

1. **Chef is a configuration management system**
   1.1. Configure web servers, database servers and etc.,
2. **Chef is a platform automation tool**
   2.1. On cloud, on premises, hybrid environments
3. **Chef transforms infrastructure into code**
   3.1. Cookbooks are the code
4. **Written by Adam Jacobs**
   4.1. Jessy Robbins of Amazon joins to setup Opscode in 2008
   4.2. Became Chef subsequently
5. **Became open source in 2019**
   5.1. With Apache 2.0 licence
6. **Current versions**
   6.1. Client: 14.10.9 and Server: 12.18.14

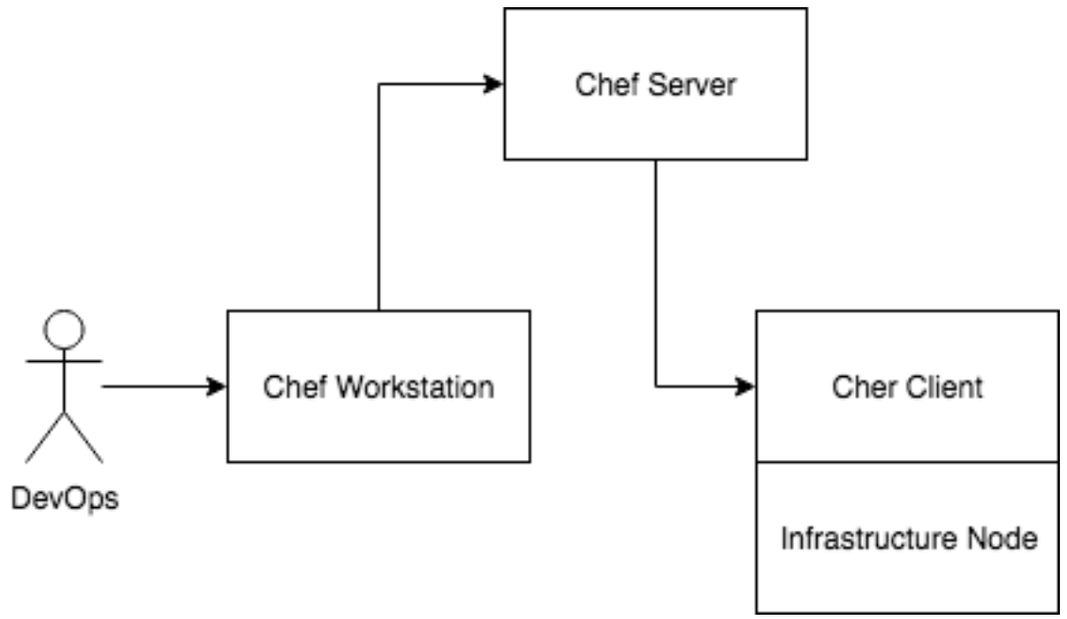
Module-1

Architecture

Architecture

1. **Chef is a client-server system**
2. **Architectural Components**
    2.1. Chef Workstation
        2.1.1. Uploads configuration to the server
        2.1.2. Written in Ruby
    2.2. Chef Server
        2.2.1. Maintains the centralised configuration
        2.2.2. Written in Erlang
    2.3. Chef Client
        2.3.1. Applies configuration on the infrastructure
        2.3.2. Written in Ruby

Architecture



Architecture - Chef Workstation

1. **Development machine to author cookbooks**
   1.1. Cookbook is a collection of recipes
   1.2. Recipe is a desired state of configuration
2. **Test machine to test cookbooks**
3. **Connects to Chef Server to upload cookbooks**
4. **Synchronises with version control system**
5. **Bootstraps nodes**
6. **Components**
   6.1. Chef Development Kit to author cookbooks
   6.2. Command Line Tools like chef, knife and etc.,
   6.3. Chef-Repo to give a structure to author cookbooks
   6.4. Test Kitchen to validate cookbooks
7. **Linux, MacOS and Windows**

Architecture - Chef Server

1. **A hub of infrastructure configuration**
    1.1. Maintains cookbooks
    1.2. Maintains policies
2. **Gets information from Chef Workstations**
3. **Indexes the node information**
4. **Chef Management Console**
    4.1. Web UI
    4.2. Manage several resources
        4.2.1. cookbooks, run-lists
        4.2.2. organisations, environments
        4.2.3. attributes, data-bags, roles
5. **Linux box**

Architecture - Chef Client

1. **Runs on infrastructure nodes**
    1.1. Servers, Network Gear and etc.,
    1.2. Physical or virtual
    1.3. On-premises or on-cloud
    1.4. Windows, Linux, MacOS
2. **Gets installed from Chef Workstation**
3. **Maintains run-lists**
4. **Interacts with Chef Server**
    4.1. To fetch configuration data
5. **Applies configuration on the node, if needed**

Module-1

Hello World

Chef Server

1. **Hosted Chef Server**
    1.1. Sign up at https://manage.chef.io/signup
    1.2. Sign in at https://manage.chef.io/login
2. **Create organization**
    2.1. An isolated part of infrastructure
    2.2. Nothing is shared across organisations
3. **Download the Starter Kit**
4. **Extract it into a workspace folder**
    4.1. chef-repo
        4.1.1. cookbooks
        4.1.2. roles
        4.1.3. and .chef

Chef Workstation

1. **Download Chef Workstation 0.8.7**
   1.1. https://downloads.chef.io/chef-workstation/
2. **Install on a development machine**
3. **Verify the** `<home>/.chef-workstation` **folder**
4. **Run the commands anywhere**
   4.1. **`which chef`**
   4.2. **`chef verify`**
   4.3. **`which chef-client`**
5. **Run the first cookbook**
   5.1. **`cd`** `workspace/chef-repo`
   5.2. **`chef-client --local-mode --override-runlist`** `starter`
6. **Verify that the log messages are printed appropriately**

Chef Client

1. **Choose the node**
   1.1. Real physical machine
   1.2. Virtual machine
   1.3. Amazon EC2 instance
2. **Ubuntu Amazon EC2 Machine as a node**
   2.1. Download the Key Pair (e.g. chef.pem)
   2.2. `chmod 6000 chef.pem`
3. **Bootstrap chef-client on the node from the workstation**
   3.1. `cd /path/to/workspace/chef-repo`
   3.2. `knife bootstrap node-ip-address -U root -i /path/to/chef.pem -N node-name —sudo`
   3.3. `knife client list`
   3.4. Update /etc/hosts with node-ip-address and node-name

Connect the dots

1. **Upload the cookbooks to server**
   1.1. `cd /path/to/chef-repo`
   1.2. `knife cookbook upload starter`
2. **Setup the run-list on the node**
   2.1. `knife node run_list add node-name 'recipe[starter::default]'`
3. **Run the chef-client on the node**
   3.1. `knife ssh -i /path/to/chef.pem 'name:node-name' 'sudo chef-client' -x root`
   3.2. Verify that the log message is printed

Module-1

Cookbooks

Cookbook: `cron-delvalidate`

1. **Move to the repository**
   1.1. `cd /path/to/chef-repo`
2. **Download and install `cron-delvalidate`**
   2.1. `knife supermarket install cron-delvalidate`
3. **Upload the cookbook to the server**
   3.1. `knife cookbook upload cron-delvalidate`
4. **Verify the list of cookbooks**
   4.1. `ls cookbooks`
5. **Add the cookbook to the run-list on the node**
   5.1. `knife node run_list add node-name 'recipe[cron-delvalidate::default]'`
6. **Run the run-list on the node**
   6.1. `knife ssh -i /path/to/chef.pem 'name:node-name' 'sudo chef-client' -x root`

Recipe: `cron-delvalidate`

```
1. cron "clientrun" do
2.   minute ‘0'
3.   hour ‘*/1'
4.   command “/usr/bin/chef-client"
5.   action :create
6. end

8. file "/etc/chef/validation.pem" do
9.   action :delete
10.end
```

Cookbook Fundamentals

1. **Chef uses cookbooks to bring a node into a specific state**
    1.1. They are the fundamental unit of configuration and policy details
2. **Cookbooks are organised in a directory structure**
    2.1. Recipes collections of resources
    2.2. Attributes: key-value settings
    2.3. Files: static files to be placed on the node
    2.4. Templates: code to generate files dynamically
    2.5. Libraries: code to extend Chef
    2.6. Metadata.rb: dependency details of cookbooks

Cookbook: Recipe

1. **/recipes/default.rb under cookbook**
2. **Collection of ordered resources**
   - 2.1. Type: around 100 predefined types
     - 2.1.1. *package, template, service, file, log, route and et.,*
   - 2.2. Name: unique within the recipe
   - 2.3. Parameters: pre-defined based on resource type
   - 2.4. Action: pre-defined based on resource type
   - 2.5. Notifications: pre-defined
3. **Resources are declarative**

```
type 'name' do
 parameter 'value'
 parameter 'value'
 action :type | [type, type, ...]
 notifies :type, type, ...
end
```

Cookbook: helloworld

1. **Generate the cookbook named helloworld**
   1.1. `cd /path/to/chef-repo/cookbooks`
   1.2. `chef generate cookbook helloworld`
2. **Author the cookbook**
   2.1. `cookbooks/helloworld/recipes/default.rb`
   2.2. `cookbooks/helloworld/files/index.html`
3. **Upload the cookbook, update the run-list and run the cookbook**
   3.1. `cd /path/to/chef-repo`
   3.2. `knife cookbook upload helloworld`
   3.3. `knife node run_list add node-name 'recipe[helloworld::default]'`
   3.4. `knife ssh -i chef.pem ‘name:node-name' 'sudo chef-client' -x root`
4. **Verify the cookbook**
   4.1. http://54.88.196.164:80/glarimy/index.html

Cookbook: recipes/default.rb

```
1. package 'apache2' do
2.   action :install
3. end

5. service 'apache2' do
6.   action [ :enable, :start ]
7. end

9. cookbook_file "/var/www/html/glarimy/
   index.html" do
10.  source "index.html"
11.  mode "0644"
12.end
```

Cookbook: files/index.html

```
<html>
  <head>
    <title>Glarimy Portal</title>
  </head>
  <body>
    <h1>Glarimy Portal</h1><hr/>
    Welcome to Chef Infra Cookbooks
  </body>
</html>
```

Thank You