CHEF™

Chef 101

Installation and Configuration Guide

With Sample Use Cases

1. Overview **4**
1.1. Cookbook Structure 4
1.2. Building Blocks of Chef 4

2. Platforms **6**
2.1. Install VirtualBox 6
Ubuntu 6
Mac OS X 6
2.2. Install Vagrant 7
Ubuntu 7
Mac OS X 7
2.3. Install Chef Development Kit 7
Ubuntu 7
Mac OS X 8
2.4. Install Git 8
Ubuntu 8
Mac OS X 9
2.5. Pre-Download Vagrant Boxes 9

3. Runtimes and Applications **10**
3.1. Redis - Kitchen - Vagrant 10
3.2 NGINX - Chef Client on Cloud 19
3.3. NGINX - Kitchen - Vagrant 25
3.4. MongoDB - Debian - Kitchen - Vagrant 27
3.5. MongoDB - APT - Kitchen - Vagrant 29
3.6. Cassandra - Debian - Kitchen - Vagrant 32
3.7. Cassandra - APT - Kitchen - Vagrant 34

References **37**

Troubleshooting **38**

1. Overview

A cookbook is the fundamental unit of configuration and policy distribution. A cookbook defines a scenario and contains everything that is required to support that scenario:

- Recipes that specify the resources to use and the order in which they are to be applied
- Attribute values
- File distributions
- Templates
- Extensions to Chef, such as custom resources and libraries

1.1. Cookbook Structure

```
|__ attributes
    |__ default.rb
|__ Berksfile
|__ chefignore
|__ files
|__ LICENSE
|__ metadata.rb
|__ README.md
|__ recipes
    |__ default.rb
|__ templates
    |__ default.conf.erb
|__ test
    |__ fixtures
        |__ default
            |__ data_bags
                |__ passwords
    |__ integration
        |__ default
            |__ default_test.rb
```

1.2. Building Blocks of Chef

`attributes`	They are basically settings. They can be thought of as a key value pair of anything which one wants to use in the cookbook
`Berksfile`	Contains the location on where to look for dependencies. By default it will get dependencies from https://supermarket.chef.io
`files`	It's a subdirectory within the cookbook that contains any static file which will be placed on the nodes that uses cookbooks.

`metadata.rb`	It is used to manage the metadata about the package. This includes details like the name and details of the package.
`recipes`	It can be defined as a collection of attributes which are used to manage the infrastructure. These attributes which are present in the recipe are used to change the existing state or setting of a particular infrastructure node.
`templates`	They are similar to files, but they are not static. Template files end with the `.ebr` extension, which means they contain embedded Ruby.
`test`	Contains tests that is used to verify the state of the instance

Berkshelf

- *is a tool that helps you resolve cookbook dependencies. Berkshelf uses a file named Berksfile to hold configuration settings. This file is generated for you when you run chef generate cookbook to create a cookbook.*

2. Platforms

2.1. Install VirtualBox

Ubuntu

a. Download the installer at the following URL

```
https://www.virtualbox.org/wiki/Linux_Downloads
```

b. Download the debian version. An example URL

```
$ curl
https://download.virtualbox.org/virtualbox/5.2.8/virtualbox-5.2_5.2.8-121009~Ubuntu~xenial_amd64.deb
--output virtualbox-5.2.deb
```

! *File size : ~70.0 Mb*

c. Execute the following command to install VirtualBox

```
$ sudo dpkg -i virtualbox-5.2.deb
```

d. Test the installation by running the following on your terminal. If you see a version, then you have installed VirtualBox successfully.

```
$ vboxmanage --version
```

```
>
5.2.12r122591
```

Mac OS X

Download the installer at https://www.virtualbox.org/wiki/Downloads

Specifically you can download the following
https://download.virtualbox.org/virtualbox/5.2.12/VirtualBox-5.2.12-122591-OSX.dmg

Double-click the .dmg file and follow the installation

2.2. Install Vagrant

Ubuntu

a. Download the installer using the following command

```
$ curl https://releases.hashicorp.com/vagrant/2.1.1/vagrant_2.1.1_x86_64.deb --output
vagrant_2.1.1_x86_64.deb
```

! *File size : ~41.5 Mb*

b. Install Vagrant using the following

```
$ sudo dpkg -i vagrant_2.1.1_x86_64.deb
```

c. Test the installation by running the following on your terminal. If you see a version, then you have installed Vagrant successfully.

```
$ vagrant --version
```

```
>
Vagrant 2.1.1
```

Mac OS X

Download DMG file from https://www.vagrantup.com/downloads.html or https://releases.hashicorp.com/vagrant/2.1.1/vagrant_2.1.1_x86_64.dmg.
Double-click or open the DMG image file and follow next steps.

2.3. Install Chef Development Kit

Ubuntu

Download and Install the Chef Development Kit

```
$ curl https://omnitruck.chef.io/install.sh | sudo bash -s -- -P chefdk -c stable -v 3.0.36
```

! *File size : ~108.5 MB*
Takes about 5-10 minutes to complete

Test the installation by running the following on your terminal. If you see a version, then you have installed ChefDK successfully.

```
$ chef --version
```

```
>
Chef Development Kit Version: 3.0.36
chef-client version: 14.1.12
delivery version: master (7206afaf4cf29a17d2144bb39c55b7212cfafcc7)
berks version: 7.0.2
kitchen version: 1.21.2
inspec version: 2.1.72
```

! *In addition to showing the Chef version, it will also show the following*
-- chef-client version
-- delivery version
-- berks version
-- kitchen version
-- inspec version

Mac OS X

Download DMG file from https://downloads.chef.io/chefdk or https://packages.chef.io/files/stable/chefdk/3.0.36/mac_os_x/10.13/chefdk-3.0.36-1.dmg. Double-click or open the DMG image file and follow next steps.

2.4. Install Git

Ubuntu

Install git from the ubuntu repository

```
$ apt-get update
$ apt-get install git
```

Test the installation by running the following on your terminal. If you see a version, then you have installed ChefDK successfully.

```
$ git --version
```

```
>
git version 2.7.4
```

Mac OS X

Pre-installed already

2.5. Pre-Download Vagrant Boxes

Ubuntu 16.04 LTS / Mac OSX

```
$ vagrant box add bento/ubuntu-16.04 --provider=virtualbox
```

! *Takes about ~5-10 minutes to complete*

3. Runtimes and Applications

! *Vagrant, VirtualBox and Git must be installed before proceeding*

3.1. Redis - Kitchen - Vagrant

3.1.1. Pull the repository using the following command

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-redis-kitchen-vagrant
or
$ git clone {{ some-other-repo-url }}
or
Local file system
```

! *Repository Size : ~237.6 KB*

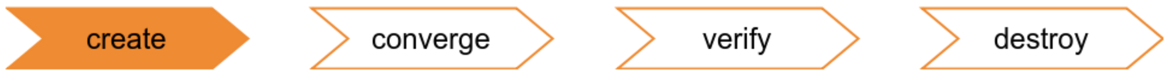
3.1.2. Enter your git account **username** and **password**

3.1.3. Navigate to the cookbook directory

```
$ cd chef-redis-kitchen-vagrant
```

3.1.4. Provision the server using the kitchen command

```
$ kitchen create
```



! *Depending if this is the first time this server is provisioned. It might take some time as it downloads the specific VirtualBox for Vagrant.*

! ***`kitchen create`***
-- creates a new instance, if the instance is not yet provisioned

-- 1. Imports the base VagrantBox bento/ubuntu-16.04
-- 2. Configures network. Allows access to port 22 for SSH

After executing kitchen create it will provision the specified platform in the `chef-redis-kitchen-vagrant/.kitchen.yml`. This will provision the instance.

</> **`chef-redis-kitchen-vagrant/.kitchen.yml`**

```
---
driver:
  name: vagrant
  ...

provisioner:
  ...

verifier:
  ...

platforms:
  - name: ubuntu-16.04
```

A variation of `.kitchen.yml` that provisions an **AWS EC2 Instance**.

</>

```
---
driver:
  name: ec2
  aws_ssh_key_id: id_rsa-aws
  security_group_ids: ["sg-1a2b3c4d"]
  region: us-west-2
  availability_zone: b
  require_chef_omnibus: true
  subnet_id: subnet-6e5d4c3b
  iam_profile_name: chef-client
  instance_type: m3.medium
  associate_public_ip: true
  interface: dns

transport:
  ssh_key: /path/to/id_rsa-aws
  connection_timeout: 10
  connection_retries: 5
  username: ubuntu

platforms:
  - name: ubuntu-16.04
  - name: centos-6.9
  - name: centos-7
  driver:
      image_id: ami-c7d092f7
      block_device_mappings:
        - device_name: /dev/sdb
          ebs:
            volume_type: gp2
            virtual_name: test
            volume_size: 8
            delete_on_termination: true
    transport:
      username: centos
  - name: windows-2012r2
  - name: windows-2016

suites:
# ...
```

A variation of `.kitchen.yml` that provisions an **Azure Instance**.

```
</>
---
driver:
  name: azurerm
  subscription_id: '4801fa9d-YOUR-GUID-HERE-b265ff49ce21'
  location: 'West Europe'
  machine_size: 'Standard_D1'

transport:
  ssh_key: ~/.ssh/id_kitchen-azurerm

provisioner:
  name: chef_zero

platforms:
  - name: ubuntu-14.04
    driver:
      image_urn: Canonical:UbuntuServer:14.04.4-LTS:latest
      vm_name: trusty-vm
      vm_tags:
        ostype: linux
        distro: ubuntu

suites:
  - name: default
    run_list:
      - recipe[kitchentesting::default]
    attributes:
```

Full reference can be found at :
https://docs.chef.io/config_yml_kitchen.html#driver-settings

The following is a partial output of the **`kitchen create`** command

```
>
-----> Starting Kitchen (v1.21.2)
```

```
-----> Creating <default-ubuntu-1604>...
       Bringing machine 'default' up with 'virtualbox' provider...
   ==> default: Importing base box 'bento/ubuntu-16.04'...
==> default: Matching MAC address for NAT networking...
       ==> default: Checking if box 'bento/ubuntu-16.04' is up to date...
       ==> default: Setting the name of the VM:
default-ubuntu-1604_default_1527215383080_36040
...
       Vagrant instance <default-ubuntu-1604> created.
       Finished creating <default-ubuntu-1604> (0m41.41s).
```

If executed for the first time andthe Vagrant Boxes have not been previously installed/downloaded, this step might take some time.

3.1.5. Check the status of the instance

```
$ kitchen list
```

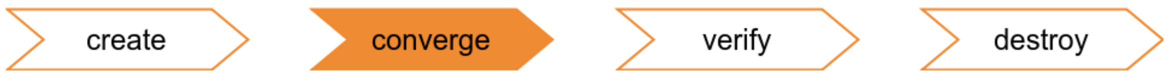
! ***kitchen list***
-- *List the instance the status*

Below is the sample output of the **kitchen list**

```
>
Instance            Driver  Provisioner Verifier Transport Last Action Last Error
default-ubuntu-1604 Vagrant ChefZero    Inspec   Ssh       Created     <None>
```

3.1.6. Execute the setup recipe using the following kitchen command

```
$ kitchen converge
```



! *Takes about ~3-5 minutes to complete*

! ***kitchen converge***
-- *Executes the recipe, installing redis and starting up the service*
-- *Note : converge will create an instance if the instance is not yet provisioned*

The **metadata.rb** contains the name of the cookbook. This is used by the .kitchen.yml to identify the namespace of the recipe.

</> chef-redis-kitchen-vagrant/metadata.rb

```
name 'custom-redis'
maintainer 'The Authors'
maintainer_email 'you@example.com'
license 'All Rights Reserved'
description 'Installs/Configures custom-redis'
long_description 'Installs/Configures custom-redis'
version '0.1.0'
chef_version '>= 12.14' if respond_to?(:chef_version)
```

After executing **kitchen converge** it will execute the recipe in the `.kitchen.yml`

</> chef-redis-kitchen-vagrant/.kitchen.yml

```
---
...

suites:
  - name: default
    run_list:
      - recipe[custom-redis::default]
```

Below is the recipe for redis. It will add update the APT repository that will be used for installing redis. Note that **node["redis"]["apt_repository"]** and other attributes will be retrieved from the **chef-redis-kitchen-vagrant/attributes/default.rb**

{ } chef-redis-kitchen-vagrant/recipes/default.rb

```
...

# Adds the repository redis
apt_repository node["redis"]["apt_repository"] do
  uri node["redis"]["apt_uri"]
  distribution node["redis"]["apt_distribution"]
  components node["redis"]["apt_components"]
  keyserver node["redis"]["apt_keyserver"]
  key node["redis"]["apt_key"]
End

...
# Installs the redis-server package
package "redis-server" do
  action node["redis"]["auto_upgrade"] ? :upgrade : :install
end

...

# Registers the service redis-server
service "redis-server" do
  supports restart: true
  action [:enable, :start]
end

# Setup the redis configuration from a template and restarts the redis-server
template "/etc/redis/redis.conf" do
```

```
  source "redis.conf.erb"
  owner  "root"
  group  "root"
  mode   "0644"
  notifies :restart, "service[redis-server]"
end
```

Note that for demo purpose we have combined the setup and installation procedure into one recipe. But it is advisable to split your cookbook into different recipes. Each with its own function. Example is to split the repository setup, installation and configuration.

Below is the attributes file that contains the configuration used in the cookbook

{ } chef-redis-kitchen-vagrant/attributes/default.rb (partial content)

```
...

default["redis"]["apt_distribution"]   = node["lsb"]["codename"]
default["redis"]["apt_repository"]     = "dotdeb"
default["redis"]["apt_uri"]            = "http://packages.dotdeb.org"
default["redis"]["apt_components"]     = ["all"]
default["redis"]["apt_key"]            = "http://www.dotdeb.org/dotdeb.gpg"

...

default["redis"]["pidfile"]            = "/var/run/redis/redis-server.pid"
default["redis"]["daemonize"]          = "yes"
default["redis"]["port"]               = 6379
default["redis"]["bind"]               = "127.0.0.1"
default["redis"]["unixsocket"]         = "/var/run/redis/redis.sock"
default["redis"]["unixsocketperm"]     = 755
default["redis"]["timeout"]            = 300
default["redis"]["loglevel"]           = "notice"
default["redis"]["logfile"]            = "/var/log/redis/redis-server.log"
default["redis"]["syslog_enabled"]     = "no"
default["redis"]["syslog_ident"]       = "redis"
default["redis"]["syslog_facility"]    = "local0"
default["redis"]["databases"]          = 16

...
```

Below is template configuration for **redis.conf.**

</> chef-redis-kitchen-vagrant/templates/redis.conf.erb (partial content)

```
daemonize <%= node["redis"]["daemonize"] %>

pidfile <%= node["redis"]["pidfile"] %>

port <%= node["redis"]["port"] %>

<% unless node["redis"]["bind"].empty? %>
bind <%= node["redis"]["bind"] %>
<% end %>

<% unless node["redis"]["unixsocket"].empty? %>
```

```
unixsocket <%= node["redis"]["unixsocket"] %>
unixsocketperm <%= node["redis"]["unixsocketperm"] %>
<% end %>

loglevel <%= node["redis"]["loglevel"] %>

logfile <%= node["redis"]["logfile"] %>

dir <%= node["redis"]["dir"] %>

<% if node["redis"]["slaveof"] && !node["redis"]["slaveof"].empty? %>
slaveof <%= node["redis"]["slaveof"] %>
<% end %>

<% if node["redis"]["masterauth"] && !node["redis"]["masterauth"].empty? %>
masterauth <%= node["redis"]["masterauth"] %>
<% end %>

...

maxmemory <%= node["redis"]["maxmemory"] %>

...
```

Below is the sample output of the **kitchen converge**

```
>
-----> Starting Kitchen (v1.21.2)
-----> Converging <default-ubuntu-1604>...
...
Installing Cookbook Gems:
       Compiling Cookbooks...
       Converging 7 resources
       Recipe: custom-redis::default
        * apt_update[daily] action periodic
...
        * apt_repository[dotdeb] action add
...
        * apt_package[redis-server] action install
...
        * service[redis-server] action enable (up to date)
        * service[redis-server] action start (up to date)
        * template[/etc/redis/redis.conf] action create
...
        * service[redis-server] action restart
           - restart service service[redis-server]
...
      Finished converging <default-ubuntu-1604> (1m34.83s).
```

! ***Tip***

-- Use converge if you want to make iterative update to the recipe.
-- Example if you want to install additional package after previously running converge.

3.1.7. Use the following command to execute tests

```
$ kitchen verify
```

create
converge
verify
destroy

After executing `kitchen verify` it will execute the tests in the `.kitchen.yml`

chef-redis-kitchen-vagrant/.kitchen.yml

```
---
...

suites:
  - name: default
    run_list:
      - recipe[custom-redis::default]
    verifier:
      inspec_tests:
        - test/integration/default
```

Below are the tests that will be executed

chef-redis-kitchen-vagrant/test/integration/default/default_test.rb

```
describe package('redis-server') do
  it { should be_installed }
end

describe port(6379) do
  it { should be_listening }
end
```

Below is the sample output of the **`kitchen verify`**

```
-----> Starting Kitchen (v1.21.2)
...
-----> Verifying <default-ubuntu-1604>...
Profile: tests from {:path=> …
...
 System Package redis-server
      ✔  should be installed
  Port 6379
      ✔  should be listening

Test Summary: 2 successful, 0 failures, 0 skipped
       Finished verifying <default-ubuntu-1604> (0m0.73s).
-----> Kitchen is finished. (0m1.82s)
```

! ***`kitchen verify`***
-- runs the tests specified in the test directory
*-- checks if the **redis-server** package is installed*
*-- checks if it is listening on port **6379***

! ***Tip - kitchen verify***
-- You can add more tests and re-run kitchen verify, it will not re-create the instance. Use this to your advantage to verify the state of your instance. Some examples of things to test are

a. if files have been created,
b. services are running,
c. and ports are accessible.

Additional Command : **`kitchen test`**

! ***Tip - kitchen test***
*-- Use **kitchen test** if you want a clean start. This will remove the instance and perform tests.*
*-- Use **kitchen test** to run InSpec tests.*
-- Since this destroys the instance, use test with caution if provisioning with Cloud Providers such as AWS and Microsoft Azure.

3.1.8. Access the provisioned server using the kitchen command

```
$ kitchen login
```

Below is the sample output of the **`kitchen login`**. This will perform an SSH client connection to the instance.

```
>
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-87-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

108 packages can be updated.
47 updates are security updates.


Last login: Fri May 25 11:47:58 2018 from 10.0.2.2
```

! ***kitchen login***
-- performs an SSH client connection to the instance

3.1.9. Test if Redis is installed and configured correctly by executing the command

```
$ redis-cli PING
```

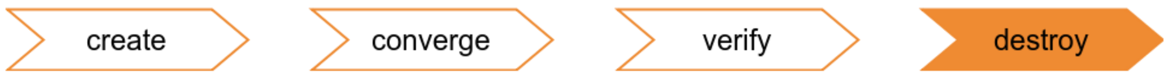
The output should be a **`PONG`**

Create a variable and increment it. Execute multiple times to see the value increasing.

```
$ redis-cli incr chef_counter
```

3.1.10. Exit from the SSH session and destroy the provisioned server. Releasing any resources taken up by the instance

```
$ exit
$ kitchen destroy
```



Below is the sample output of the **`kitchen destroy`**.

```
>
-----> Starting Kitchen (v1.21.2)
-----> Destroying <default-ubuntu-1604>...
       ==> default: Forcing shutdown of VM...
       ==> default: Destroying VM and associated drives...
       Vagrant instance <default-ubuntu-1604> destroyed.
       Finished destroying <default-ubuntu-1604> (0m3.52s).
-----> Kitchen is finished. (0m4.62s)
```

! ***`kitchen destroy`***
-- *Destroys the instance. Removing any VirtualBox created by this Cookbook*

3.2 NGINX - Chef Client on Cloud

3.2.1. Prepare the SSH keys on the following directory

! *~/ws/keys/devops-lightsail-key-01.pem*

Update the permission of the key by executing the following:

```
$ chmod 400  ~/ws/keys/devops-lightsail-key-01.pem
```

3.2.2. Access the server using the following commands

```
$ ssh-add ~/ws/keys/devops-lightsail-key-01.pem
$ ssh ubuntu@52.74.176.197
```

```
>
The authenticity of host '52.74.176.197 (52.74.176.197)' can't be established.
ECDSA key fingerprint is SHA256:5b4oV67yN2Z09ggOy9qf4SR9/yIomlIQbtyvCw7XJnY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '52.74.176.197' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1052-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

52 packages can be updated.
0 updates are security updates.


*** System restart required ***
Last login: Mon May 28 08:24:11 2018 from 120.28.111.167
ubuntu@ip-172-26-47-191:~$
```

3.2.3. Pull the repository using the following command

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-nginx-chef-client-cloud

# or

$ git clone {{ some-other-repo-url }}

# or

Local file system
```

! *Repository Size : ~75.9 KB*

3.2.4. Enter your git account **username** and **password**

3.2.5. Navigate to the cookbook directory

```
$ cd chef-nginx-chef-client-cloud
```

3.2.6. Execute the following to setup NGINX on the local machine

```
$ sudo chef-client --local-mode setup_nginx.rb
```

chef-client setup_nginx.rb chef-client remove_nginx.rb

</> **chef-nginx-local/setup_nginx.rb**

```
# Update apt packages
# Equivalent to : sudo apt-get update
apt_update 'Update the apt cache daily' do
  frequency 86_400
  action :update
end

# Install nginx
# Equivalent to : sudo apt-get install nginx
package 'nginx'

# Start nginx service and enable on boot
# Equivalent to : sudo service nginx start
service 'nginx' do
  supports status: true
  action [:enable, :start]
end

# Create the directory for websites
# Equivalent to : mkdir /var/www/chef
directory '/var/www/chef' do
  owner 'www-data'
  group 'www-data'
  mode '0755'
  action :create
end

# Create the html file
# Equivalent to : nano /var/www/chef/index.html
file '/var/www/chef/index.html' do
```

```
  content '<html>
  <body>
    <h1>Hello NGINX from Chef</h1>
  </body>
</html>
'
end

# Create nginx config
# Equivalent to : nano /var/nginx/sites-available/chef.conf
file '/etc/nginx/sites-available/chef.conf' do
  content 'server
{
  listen      80;
  server_name _;
  root        /var/www/chef;
  index       index.html index.htm;
}
'
end

# Delete default nginx config
# Equivalent to : rm -r /etc/nginx/sites-enabled/default
file '/etc/nginx/sites-enabled/default' do
  action :delete
end

# Enable nginx config
# Equivalent to : sudo ln -s /etc/nginx/sites-available/chef.conf
/etc/nginx/sites-enabled/chef.conf
link '/etc/nginx/sites-enabled/chef.conf' do
  to '/etc/nginx/sites-available/chef.conf'
  link_type :symbolic
end

# Equivalent to : sudo service nginx restart
# Restart nginx service
service 'nginx' do
  action [:restart]
end
```

Output of the chef-client command. It shows the step-by-step execution of the recipe.

```
>
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks ...
...
Recipe: @recipe_files ...
  * apt_update[Update the apt cache daily] action update
...
 * apt_package[nginx] action install
...
  * service[nginx] action enable (up to date)
  * service[nginx] action start (up to date)
  * directory[/var/www/chef] action create
    - create new directory /var/www/chef
    - change mode from '' to '0755'
    - change owner from '' to 'www-data'
    - change group from '' to 'www-data'
  * file[/var/www/chef/index.html] action create
```

```
    - create new file /var/www/chef/index.html
...
Running handlers:
Running handlers complete
Chef Client finished, 9/14 resources updated in 51 seconds
```

! ***sudo chef-client --local-mode {ruby_file}.rb***
-- *Executes the ruby code/config*

3.2.7 Test if NGINX is installed and configured correctly by executing the command

```
$ curl localhost
```

Below is the output of the curl command. It shows that NGINX has been successfully deployed and configured.

```
>
<html>
  <body>
    <h1>Hello NGINX from Chef</h1>
  </body>
</html>
```

Access the following on the browser and it should also show the **Hello NGINX from Chef** message.

```
http://52.74.176.197
```

3.2.8. Perform cleanup using a different recipe

```
$ sudo chef-client --local-mode remove_nginx.rb
```

chef-client setup_nginx.rb
chef-client remove_nginx.rb

</> **chef-nginx-local/remove_nginx.rb**

```
# Remove nginx
# Equivalent to : sudo apt-get purge nginx
package 'nginx' do
```

```
  action :purge
end

# Equivalent to : sudo apt-get purge nginx-common
package 'nginx-common' do
  action :purge
end

# Equivalent to : sudo apt-get purge nginx-core
package 'nginx-core' do
  action :purge
end
```

Output of the cleanup recipe

```
>
Synchronizing Cookbooks:
Installing Cookbook Gems:
Compiling Cookbooks...
...
Recipe: @recipe_files::/home/vagrant/chef-nginx-local/remove_nginx.rb
  * apt_package[nginx] action purge
    - purge  package nginx
  * apt_package[nginx-common] action purge
    - purge  package nginx-common
  * apt_package[nginx-core] action purge (up to date)

Running handlers:
Running handlers complete
Chef Client finished, 2/3 resources updated in 14 seconds
```

3.2.9. Verify that NGINX has been removed.

```
$ nginx -v
```

Output of the NGINX version should show that there is no package installed

```
>
The program 'nginx' can be found in the following packages:
 * nginx-core
 * nginx-extras
 * nginx-full
 * nginx-light
Try: sudo apt install <selected package>
```

Verify also that the HTML page is not accessible anymore

```
$ curl localhost
```

```
>
curl: (7) Failed to connect to localhost port 80: Connection refused
```

3.3. NGINX - Kitchen - Vagrant

3.3.1. Pull the repository using the following command

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-nginx-kitchen-vagrant
# or
$ git clone {{ some-other-repo-url }}
# or
Local file system
```

! *Repository Size : ~201.4 KB*

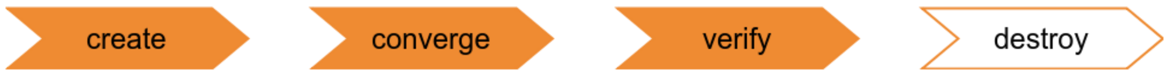
3.3.2. Enter your git account **username** and **password**

3.3.3. Navigate to the cookbook directory

```
$ cd chef-nginx-kitchen-vagrant
```

3.3.4. Provision the server using the kitchen command

```
$ kitchen verify
```



! *Depending if this is the first time this server is provisioned. It might take some time as it downloads the specific VirtualBox for Vagrant. It will also run the recipe, installing **nginx**.*

! ***`kitchen verify`***
-- creates a new instance, if the instance is not yet provisioned
-- runs the tests specified in the test directory
*-- checks if the **nginx** package is installed*
*-- checks if it is listening on **port 80***

3.3.5. Access the provisioned server using the kitchen command

```
$ kitchen login
```

! ***kitchen login***
-- performs an SSH client connection to the instance

3.3.6. Test if NGINX is installed and configured correctly by executing the command

```
$ curl localhost
```

It should show the following HTML output

```
>
<html>
  <title>Chef - NGINX</title>
  <body>
       Hello from Chef - NGINX
  </body>
</html>
```

This can also be accessed in the Private Network by setting the following URL in the browser. This IP should be reserved in the network beforehand. If this has been taken, it is advised to use a different IP.

```
http://192.168.99.99
```

3.3.7. Exit from the SSH session and destroy the provisioned server. Releasing any resources taken up by the instance

```
$ exit
$ kitchen destroy
```

create
converge
verify
destroy

3.4. MongoDB - Debian - Kitchen - Vagrant

3.4.1. Pull the repository using the following command

Note that this version of MongoDB downloads the debian packages included in the repository. It might take some time to pull the repository.

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-mongo-debian-kitchen-vagrant

# or

$ git clone {{ some-other-repo-url }}

# or

Local file system
```

! *Repository Size : ~85 MB*

! *Takes about ~3-5 minutes to complete*

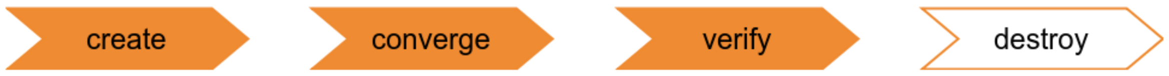
3.4.2. Enter your git account **username** and **password**

3.4.3. Navigate to the cookbook directory

```
$ cd chef-mongo-debian-kitchen-vagrant
```

3.4.4. Provision the server using the kitchen command

```
$ kitchen verify
```



! *Takes about ~3-5 minutes to complete*

! *Depending if this is the first time this server is provisioned. It might take some time as it downloads the specific VirtualBox for Vagrant. It will also run the recipe, installing **MongoDB**.*

! ***kitchen verify***
-- if run first, it will function as if create and converge are executed
-- creates a new instance, if the instance is not yet provisioned
-- runs the tests specified in the test directory
*-- checks if the **mongodb** package is installed*
*-- checks if it is listening on port **27017***

3.4.5. Access the provisioned server using the kitchen command

```
$ kitchen login
```

! ***`kitchen login`***
-- performs an SSH client connection to the instance

3.4.6. Test if MongoDB is installed and configured correctly by executing the command

```
$ mongo
```

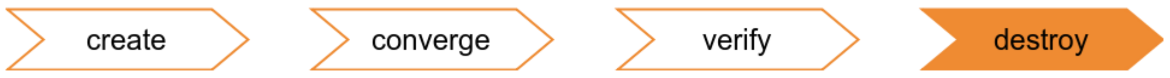
3.4.7. Execute some mongodb commands

Note: Do not copy paste the creation of collection and document directly from this document. As the apostrophe might become a special character.

```
> show databases;

# show all collections
> show collections;

# create a new collection
> db.createCollection('chef_mongo');

# show all collections
> show collections;

# add a new document
> db.chef_mongo.insert({'message' : 'Hi Chef!'});

# query the collection, and view the added document
> db.chef_mongo.find();
```

3.4.8. Exit from the SSH session and destroy the provisioned server. Releasing any resources taken up by the instance

```
# exit from the mongo shell
$ exit

# exit from the server
$ exit

# destroy the server
$ kitchen destroy
```



! ***kitchen destroy***
-- *Destroys the instance. Removing any VirtualBox created by this Cookbook*

3.5. MongoDB - APT - Kitchen - Vagrant

3.5.1. Pull the repository using the following command

Note that this version of MongoDB downloads the debian packages included in the repository. It might take some time to pull the repository.

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-mongo-kitchen-vagrant
# or
$ git clone {{ some-other-repo-url }}
# or
Local file system
```

! *Repository Size : ~300 kb*

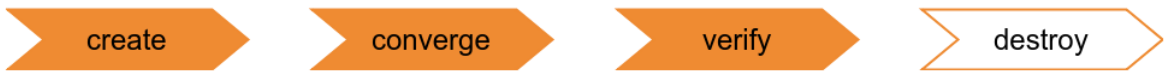
3.5.2. Enter your git account **username** and **password**

3.5.3. Navigate to the cookbook directory

```
$ cd chef-mongo-kitchen-vagrant
```

3.5.4. Provision the server using the kitchen command

```
$ kitchen verify
```



! *Takes about ~3-5 minutes to complete*

! *Depending if this is the first time this server is provisioned. It might take some time as it downloads the specific VirtualBox for Vagrant. It will also run the recipe, installing **MongoDB**.*

! ***kitchen verify***
-- if run first, it will function as if create and converge are executed

-- creates a new instance, if the instance is not yet provisioned
-- runs the tests specified in the test directory
*-- checks if the **mongodb** package is installed*
*-- checks if it is listening on port **27017***

3.5.5. Access the provisioned server using the kitchen command

```
$ kitchen login
```

! ***kitchen login***
-- performs an SSH client connection to the instance

3.5.6. Test if MongoDB is installed and configured correctly by executing the command

```
$ mongo
```

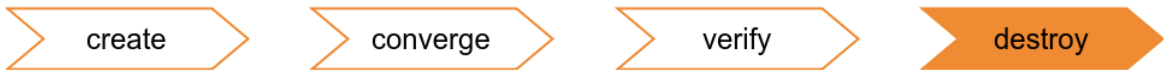
3.5.7. Execute some mongodb commands

Note: Do not copy paste the creation of collection and document directly from this document. As the apostrophe might become a special character.

```
> show databases;

# show all collections
> show collections;

# create a new collection
> db.createCollection('chef_mongo');

# show all collections
> show collections;

# add a new document
> db.chef_mongo.insert({'message' : 'Hi Chef!'});

# query the collection, and view the added document
> db.chef_mongo.find();
```

3.5.8. Exit from the SSH session and destroy the provisioned server. Releasing any resources taken up by the instance

```
# exit from the mongo shell
$ exit

# exit from the server
$ exit

# destroy the server
$ kitchen destroy
```



! ***`kitchen destroy`***
-- *Destroys the instance. Removing any VirtualBox created by this Cookbook*

3.6. Cassandra - Debian - Kitchen - Vagrant

3.6.1. Pull the repository using the following command

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-cassandra-debian-kitchen-vagrant

# or

$ git clone {{ some-other-repo-url }}

# or

Local file system
```

! *Repository Size : ~60 MB*

! *Takes about ~3-5 minutes to complete*

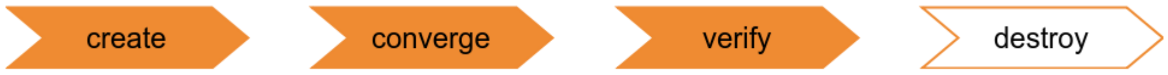
3.6.2. Enter your git account **username** and **password**

3.6.3. Navigate to the cookbook directory

```
$ cd chef-cassandra-debian-kitchen-vagrant
```

3.6.4. Provision the server using the kitchen command

```
$ kitchen verify
```



! *Takes about ~3-5 minutes to complete*

! *Depending if this is the first time this server is provisioned. It might take some time as it downloads the specific VirtualBox for Vagrant. It will also run the recipe, installing **MongoDB**.*

! ***`kitchen verify`***
-- if run first, it will function as if create and converge are executed
-- creates a new instance, if the instance is not yet provisioned
-- runs the tests specified in the test directory
*-- checks if the **cassandra** package is installed*

3.6.5. Access the provisioned server using the kitchen command

```
$ kitchen login
```

! ***`kitchen login`***
-- performs an SSH client connection to the instance

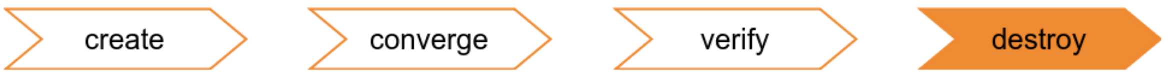
3.6.6. Test if Cassandra is installed and configured correctly by executing the command

```
$ cqlsh
```

```
>
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

3.6.7. Exit from the Cassandra Shell and SSH session and destroy the provisioned server. Releasing any resources taken up by the instance

```
# exit from the Cassandra shell
$ exit

# exit from the server
$ exit

# destroy the server
$ kitchen destroy
```



3.7. Cassandra - APT - Kitchen - Vagrant

3.7.1. Pull the repository using the following command

```
$ cd ~/ws/chef
```

```
$ git clone https://{{ repo-root-path }}/chef-cassandra-kitchen-vagrant

# or

$ git clone {{ some-other-repo-url }}

# or

Local file system
```

! *Repository Size : ~ 195.2 KB*

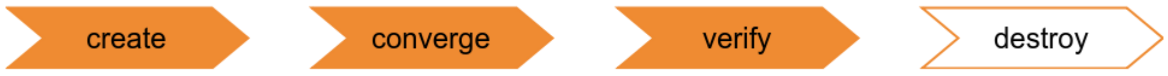
3.7.2. Enter your git account **username** and **password**

3.7.3. Navigate to the cookbook directory

```
$ cd chef-cassandra-kitchen-vagrant
```

3.7.4. Provision the server using the kitchen command

```
$ kitchen verify
```



! *Takes about ~4-6 minutes to complete*

! *Depending if this is the first time this server is provisioned. It might take some time as it downloads the specific VirtualBox for Vagrant. It will also run the recipe, installing **MongoDB**.*

! ***`kitchen verify`***
-- if run first, it will function as if create and converge are executed
-- creates a new instance, if the instance is not yet provisioned
-- runs the tests specified in the test directory
*-- checks if the **cassandra** package is installed*

3.7.5. Access the provisioned server using the kitchen command

```
$ kitchen login
```

! ***`kitchen login`***
-- performs an SSH client connection to the instance

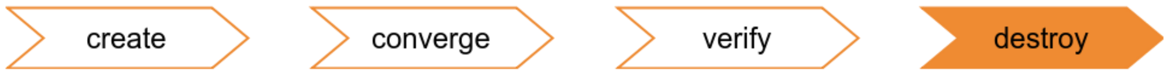
3.7.6. Test if Cassandra is installed and configured correctly by executing the command

```
$ cqlsh
```

```
>
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh>
```

3.7.7. Exit from the Cassandra Shell and SSH session and destroy the provisioned server. Releasing any resources taken up by the instance

```
# exit from the Cassandra shell
$ exit

# exit from the server
$ exit

# destroy the server
$ kitchen destroy
```



References

- Chef Automation
  - https://docs.chef.io/chef_automate.html
- Full list of supported drivers and cloud platforms
  - https://docs.chef.io/config_yml_kitchen.html#driver-settings
- Kitchen CI - Infrastructure Testing
  - https://kitchen.ci/

Troubleshooting

- ShellCommandFailed for apt_repository - This error occurs when the node is unable to connect to the APT key server. A retry of the kitchen command will try to resolve it.
    - For example, if Cassandra is unable to download the APT key from the pool.sks-keyservers.net key server.
- A good internet connection is required for APT repository install. An alternative to use APT repository is to create a recipe that includes the debian packages or any other installers.