Create custom resources in Chef and use them in your recipes

Table of Contents

- Description
- Problem Statement
- Prerequisites
- Implementation Steps
  - Step-1: Define the Custom Resource
  - Step-2: Use the Custom Resource in a Recipe
- References

Description

Custom resources in Chef allow you to package reusable code for frequently performed tasks. By defining custom resources, you can encapsulate logic and actions, making Chef recipes more modular and maintainable. This guide demonstrates how to create a custom resource that installs and configures a web server, and how to use it in your Chef recipes.

Problem Statement

When managing multiple configurations or systems, it becomes difficult to maintain and replicate complex configurations. Using Chef's custom resources, you can:

- Modularize configuration logic.
- Reuse resources across multiple recipes.
- Standardize configurations.

Prerequisites

Software Required

- **Chef Workstation**: To create and test the custom resource and recipe.
- **Chef Infra Client**: To apply the configuration.

Hardware Requirement

- A **Chef Workstation** with internet access for downloading dependencies.

Implementation Steps

Step-1: Define the Custom Resource

1. **Create a Directory for the Resource**: Inside your Chef cookbook, create a directory named `resources`.

```
mkdir -p cookbooks/webserver/resources
```

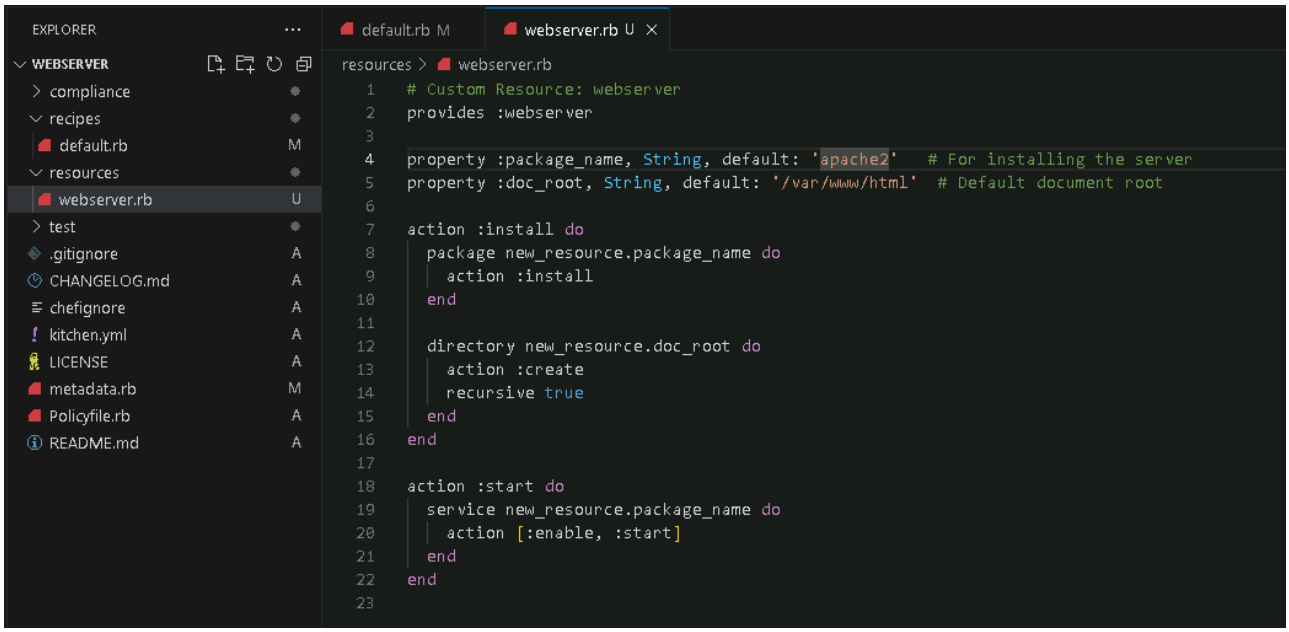
2. **Define the Custom Resource**: Create a Ruby file in the `resources` directory. This example shows a resource that installs a web server, configures a document root, and starts the server.

   File: `cookbooks/webserver/resources/webserver.rb`

```
# Custom Resource: webserver
provides :webservers

property :package_name, String, default: 'apache2'   # For installing the
server
property :doc_root, String, default: '/var/www/html'  # Default document
root

action :install do
  package new_resource.package_name do
    action :install
  end

  directory new_resource.doc_root do
    action :create
    recursive true
  end
end

action :start do
  service new_resource.package_name do
    action [:enable, :start]
  end
end
```

   In this example:

   - `property :package_name` and `property :doc_root` define configurable parameters.
   - `:install` and `:start` are actions that will be performed by the custom resource.



3. **Define a Default Action**: By setting a default action, you control which action runs if none is specified.

```
default_action :install
```

Step-2: Use the Custom Resource in a Recipe

Once the custom resource is defined, you can use it in any recipe in the same cookbook or from other cookbooks that include this one.

1. **Create a Recipe File**: Inside your `recipes` directory, create a recipe that uses the `webserver` custom resource.

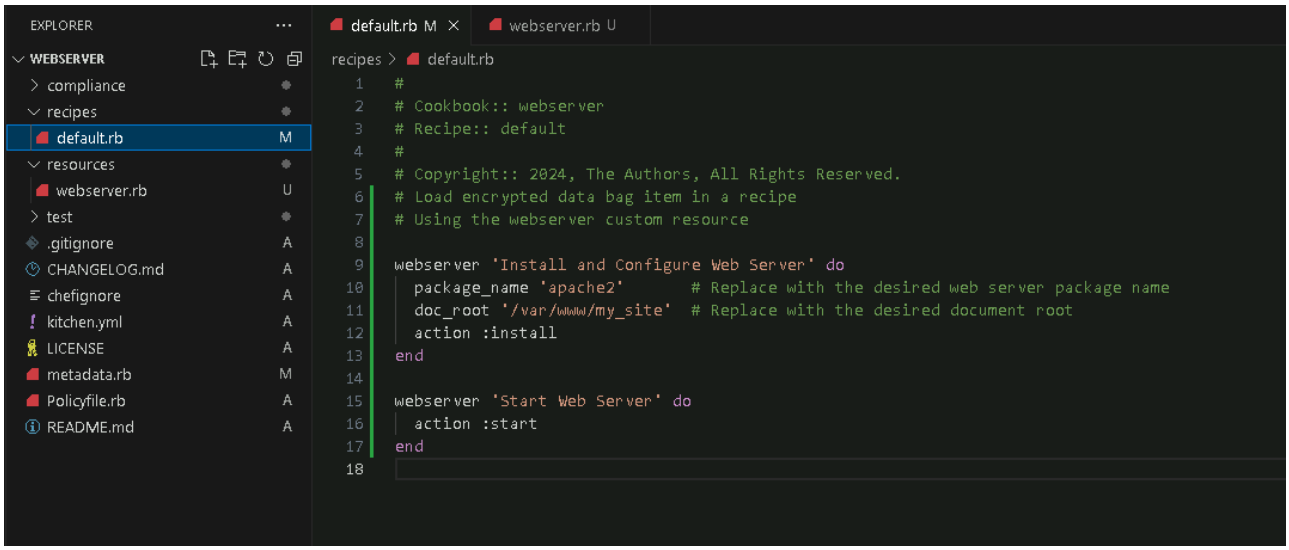
   File: `cookbooks/webserver/recipes/default.rb`

```
# Using the webserver custom resource

webserver 'Install and Configure Web Server' do
  package_name 'apache2'       # Replace with the desired web server package
name
  doc_root '/var/www/my_site'  # Replace with the desired document root
  action :install
end

webserver 'Start Web Server' do
  action :start
end
```

   In this example:

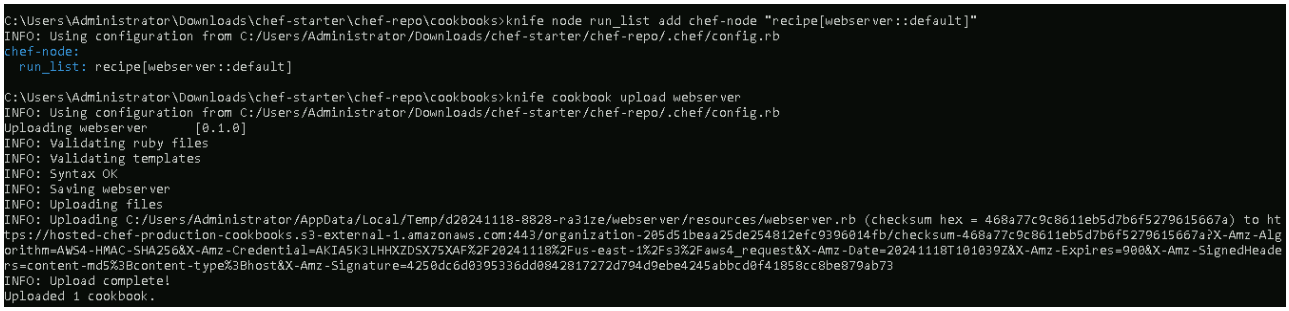
   - **First Block**: Installs and configures the web server by setting the package name and document root.

- **Second Block**: Starts the web server.

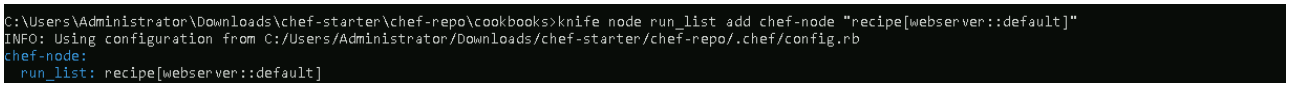


2. **Upload cookbook**

```
knife cookbook upload webserver
```




```
knife node run_list add chef-node "recipe[webserver::default]"
```



3. **Run the Recipe**: Apply the recipe to a node to verify the custom resource works as expected:

```
sudo chef-client
```

```
vagrant@default-ubuntu-2004:~$ sudo chef-client
Chef Infra Client, version 18.5.0
Patents: https://www.chef.io/patents
Infra Phase starting
Resolving cookbooks for run list: ["webserver::default"]
Synchronizing cookbooks:
  - webserver (0.1.0)
Installing cookbook gem dependencies:
Compiling cookbooks...
Loading Chef InSpec profile files:
Loading Chef InSpec input files:
Loading Chef InSpec waiver files:
Converging 2 resources
Recipe: webserver::default
  * webserver[Install and Configure Web Server] action install
    * apt_package[apache2] action install (up to date)
    * directory[/var/www/my_site] action create (up to date)
     (up to date)
  * webserver[Start Web Server] action start
    * service[apache2] action enable (up to date)
    * service[apache2] action start (up to date)
     (up to date)

Running handlers:
Running handlers complete
Infra Phase complete, 0/6 resources updated in 13 seconds
vagrant@default-ubuntu-2004:~$ _
```

4. **Verify the Web Server**: Confirm that the web server is installed, running, and accessible. Check that the document root is created at the specified path.

```
sudo systemctl status apache2
```

```
vagrant@default-ubuntu-2004:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-11-18 09:55:19 UTC; 23min ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 2781 (apache2)
      Tasks: 55 (limit: 2256)
     Memory: 5.6M
     CGroup: /system.slice/apache2.service
             ├─2781 /usr/sbin/apache2 -k start
             ├─2783 /usr/sbin/apache2 -k start
             └─2784 /usr/sbin/apache2 -k start
```

```
vagrant@default-ubuntu-2004:~$ ls -ld /var/www/my_site
drwxr-xr-x 2 root root 4096 Nov 18 09:55 /var/www/my_site
vagrant@default-ubuntu-2004:~$
```

References

- Chef Custom Resources Documentation
- Chef Properties and Attributes