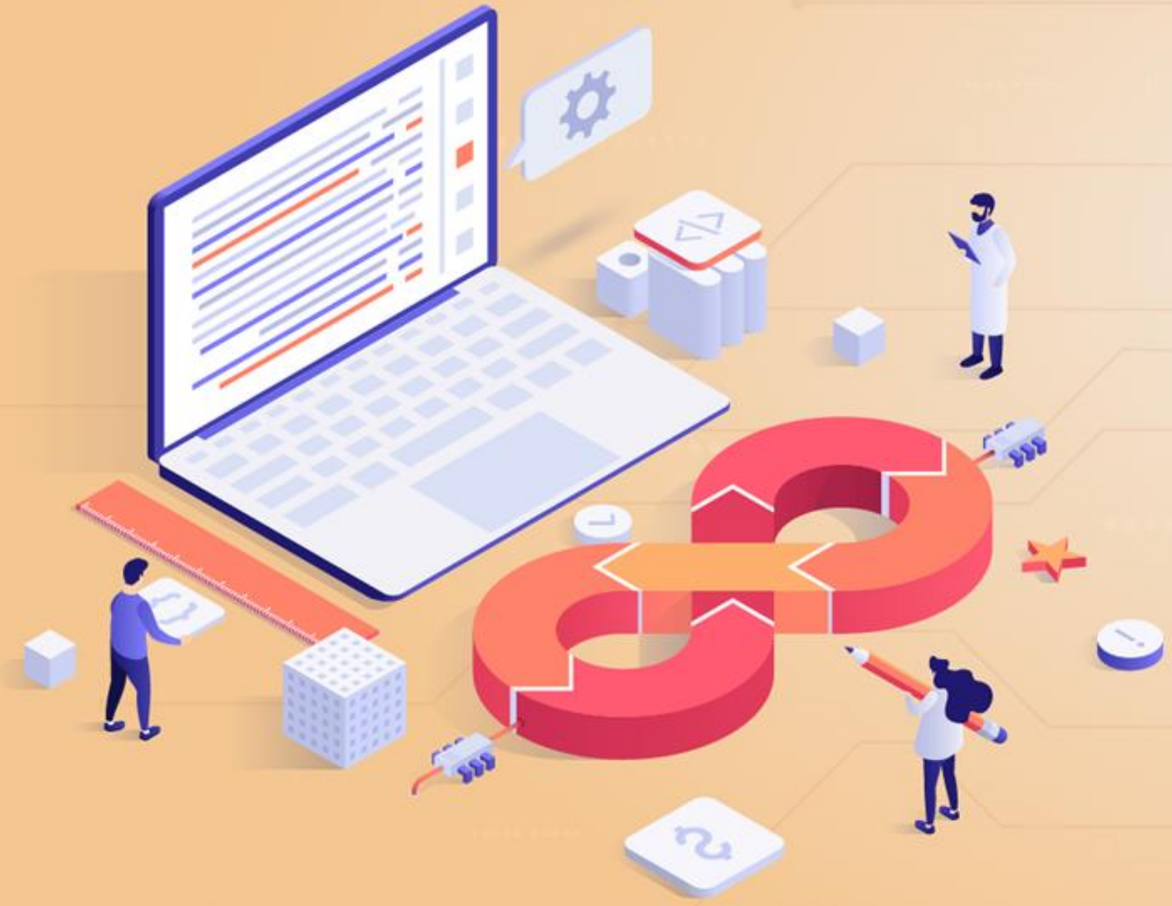
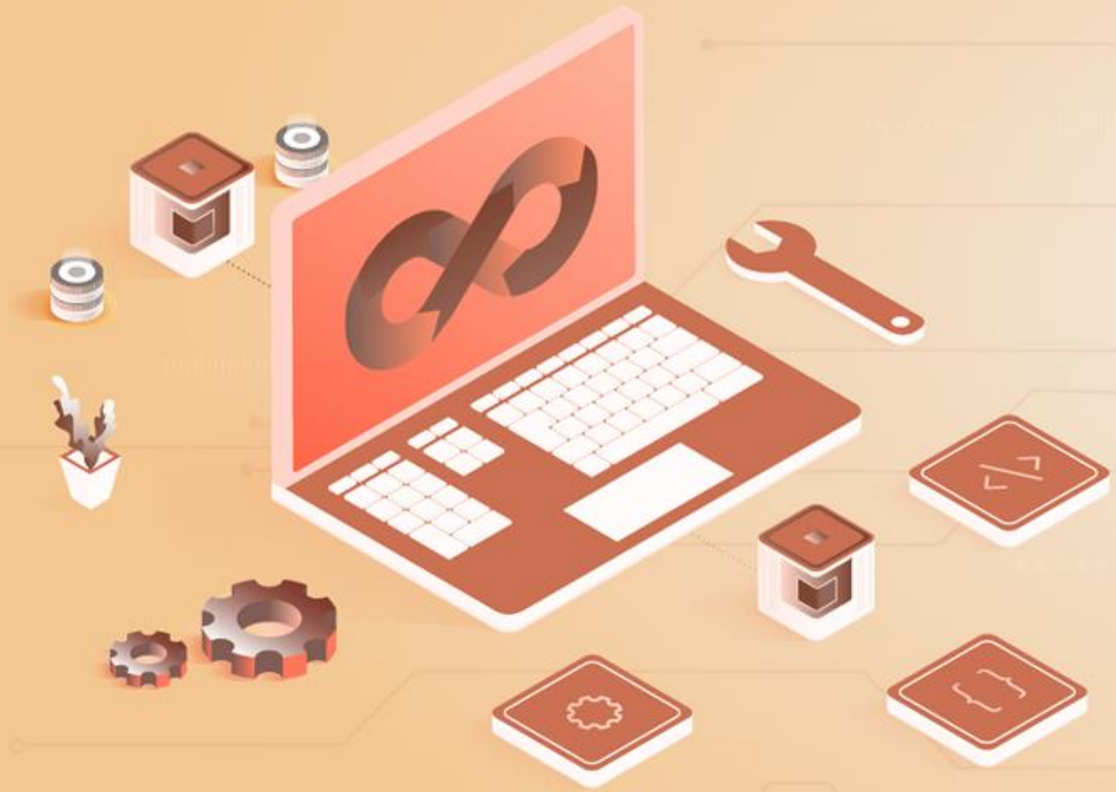
DevOps



Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

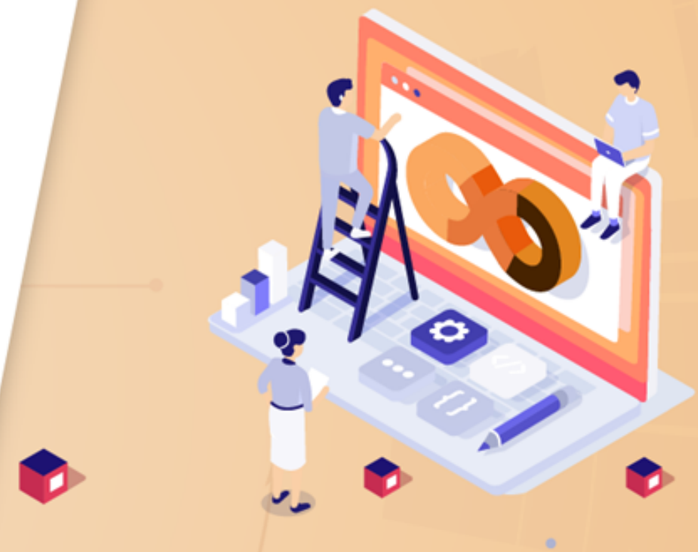


Chef Cookbooks and Recipes

Learning Objectives

By the end of this lesson, you will be able to:

- Define and explain what is Cookbook
- Outline the building blocks of Cookbook
- Create a Cookbook using Chef command
- Explain Chef roles and environment
- Outline the fundamentals of Chef Recipes
- Implement Recipe configuration in Cookbook



Introduction to Cookbook

What Is Cookbook?

Cookbook is a fundamental unit of configuration and policy distribution.

It comprises of recipes and other components such as files and directories.

It was created using Ruby language and domain specific language for specific resources.

cookbook

Key Components of Cookbook

Key Components

1. Recipes
2. Attributes
3. File Distribution
4. Libraries
5. Custom Resources
6. Templates
7. Metadata.rb

Recipes

recipe

- Recipe is a fundamental configuration element within the organization.
- It must define everything required to configure a part of the system and must reside in the cookbook.

Attributes

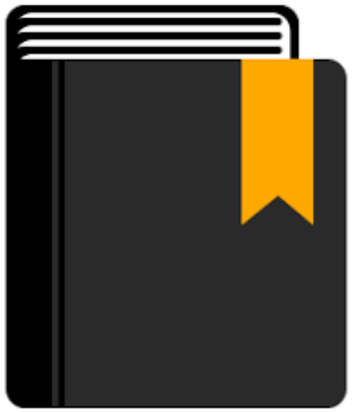
- Attributes are used to override the default settings on a node and are defined in a cookbook (or a recipe).

File Distribution

files

- A specific type of resource that tells a cookbook how to distribute files by node, by platform, or by file version

Libraries



libraries

- These allow the arbitrary Ruby code in a cookbook to either extend the Chef Infra Client language or to implement a new class.

Custom Resources

- An abstract approach for defining a set of actions and (for each action) a set of properties and validation parameters

Template

templates

- A file written in markup language that uses Ruby statements to solve complex configuration scenarios

Metadata

metadata

- A file that contains cookbook information such as cookbook name, description, and version

Cookbook Configuration

Assisted Practice

Create a Cookbook Using Chef Command

Problem Statement:

You are given a project to create a cookbook and update it for further process in Chef.

Assisted Practice: Guidelines

Steps to perform:

1. Create a cookbook in the directory
2. Confirm the cookbook creation
3. Update the cookbook in the Chef environment

Cookbook Dependencies

Metadata.rb File

- Located at the top in the cookbook directory structure
- Compiled when the cookbook is uploaded to Chef server using knife command
- Compiled with knife cookbook metadata subcommand
- Created automatically when the knife cookbook create command is run

Chef Roles

Chef Roles: Working

Roles in Chef are a logical way of grouping the nodes.

Working Procedure:

- Define a role in a Ruby file inside the roles folder of Chef repository.
- A role consists of a name and a description attribute.
- A role consists of role-specific run list and role-specific attribute settings.
- Every node that has a role in its run list will have the role's run list exacted into its own.

Chef Roles: Working

Working Procedure:

- All the recipes in the role's run list will be executed on the node.
- The role will be uploaded to Chef server using the knife role from file command.
- The role will be added to the node run list.
- Running Chef client on a node having the role in its run list will execute all the recipes listed in the role.

Assisted Practice

Create, Upload, and Assign a Chef Role to the Client

Problem Statement:

You are assigned to create and upload Chef role and assign it to the Chef-client in your project.

Assisted Practice: Guidelines

Steps to perform:

1. Create a Chef role
2. Upload a role to the server
3. Assign a role to the node
4. Execute the Chef client

Chef Environment

Assisted Practice

Create and Test the Chef Environment

Problem Statement:

You are given a project to create and test the Chef environment to check Chef tool functions.

Assisted Practice: Guidelines

Steps to perform:

1. Create a Chef environment
2. Test the Chef environment

Chef-Shell

Chef-Shell

It’s a recipe debugging tool that allows the use of breakpoints within recipes.
It runs as an interactive Ruby session and has three modes.

1. Standalone
2. Solo
3. Client

Chef-Shell: Syntax

Chef-shell command has the following syntax:

```
chef-shell OPTION VALUE OPTION VALUE ...
```

Assisted Practice

Execute Chef-Shell in Standalone Mode

Problem Statement:

You are given a project to execute Chef-shell in the standalone mode.

Assisted Practice: Guidelines

Steps to perform:

1. Execute Chef shell
2. Switch to attribute mode in Chef shell
3. Set attribute value
4. Switch to recipe mode
5. Create a file resource
6. Commence Chef run

Testing Cookbooks

Testing Cookbooks: Working Method

- The knife cookbook test executes a Ruby syntax check on all the Ruby files within the cookbook.
- It loops through Ruby files and runs Ruby –c against each file.
- Ruby –c checks the syntax of the script and quits without executing it.
- The knife cookbook test checks all ERB templates and pipes.
- The redundant version of the cookbook test is either created using –x or Ruby –c.

Testing Cookbooks: Limitations

- Syntax check is restricted to Ruby files and ERB templates.
- The test does not support test-driven frameworks.

Chef-Repo

Chef-Repo

Chef-repo is a directory on the workstation that stores everything needed to define the infrastructure with Chef Infra.

It includes:

1. Cookbooks
2. Data bags
3. Policyfiles

Chef-Repo: Directories

The various sub-directories in the chef-repo are:

1. .chef/
2. .cookbooks/
3. .data_bags/
4. .policyfiles/

Chef Recipe

Introduction to Chef Recipe

- A chef recipe is authored using the Ruby programming language.
- It is the fundamental configuration element within organization.
- It is a collection of resources defined using patterns.
- It should be added to a run-list before it is used by a client.

Recipe Attribute Types

1. default
2. force_default
3. normal
4. override
5. automatic

File Methods

Use the below methods within the attribute file for a cookbook or within a recipe. These methods correspond to the attribute type of the same name.

1. override
2. default
3. normal
4. _unless
5. attribute?

Environment Variables

Environment is a set of key-value pairs made available to a process. If the process is started using the **execute** or **script** resources, use the `environment` attribute to alter it.

```
bash 'env_test' do
  code &lt;&lt;-EOF
  echo $FOO
EOF
  environment ({ 'FOO' => 'bar' })
end
```

Working with Recipes

Data Bags

- Data bags store global variables as JSON data.
- They are indexed for searching and can be loaded by a cookbook or accessed during a search.
- The contents of a data bag can be loaded into a recipe.

Data Bags: Example

The example given below displays the name and repository of the data bag.

```
{
  "id": "test",
  "repository":
"git://github.com/company/test.git"
}
```

Accessing Data Bag

Data bag can be accessed in a recipe as shown below:

```
my_bag = data_bag_item('apps','test')
```

Secret Keys

Secret Keys

Secret key is used to encrypt a data bag and is created in multiple ways.

Example:

```
openssl rand -base64 512 | tr -d
'\r\n' > encrypted_data_bag_secret
```

Store Keys on Nodes

- An encryption key is stored in an alternate file on the nodes.
- EncryptedDataBagItem.load anticipates the actual secret key as the third argument, rather than a path to the secret file.

Assign Dependencies

Assigning Dependencies

If a cookbook has a dependency on a recipe that is located in another cookbook, then it must be declared in the metadata.rb file using the **depends** keyword.

Example:

```
include_recipe 'apache2::mod_ssl'

Metadata file:

depends 'apache2'
```

Include Recipes

Include Recipes

A recipe can include one or more recipes from cookbooks by using the `include_recipe` method.

```
include_recipe 'recipe'
```

Include Recipes

Multiple Recipes can be included using the syntax below:

```
include_recipe 'cookbook::setup'
include_recipe 'cookbook::install'
include_recipe 'cookbook::configure'
```

Reload Attributes

Reload Attributes

Attributes depend on actions taken from recipes. Hence, it's necessary to reload a given attribute from a recipe.

```
ruby_block 'some_code' do
  block do

node.from_file(run_context.resolve_att
ribute('COOKBOOK_NAME', 'ATTR_FILE'))
  end
  action :nothing
end
```

Use Tags

Use Tags

A tag is a custom description that is applied to a node. Once applied, a tag helps in managing nodes using knife.

Using tags in a recipe:

```
tag('mytag')
```

Tagging a machine:

```
tagged?('mytag')
```

Remove a tag:

```
untag('mytag')
```

End Chef Client Infra Run

End Chef Client Infra Run

Steps to end a Chef Client Infra Run are given below:

- return keyword is used to stop processing a recipe
- raise keyword is used to stop Chef Infra Client run by unhandled exception
- rescue block is used to stop Ruby instance
- Chef::Application.fatal! is used to log a fatal message to stop client run

return Keyword

The return keyword is used to stop the processing of a recipe based on a condition, but continues the processing of a Chef Infra Client run.

```
file '/tmp/name_of_file' do
  action :create
end
return if
platform?('windows')
package 'name_of_package'
do
  action :install
end
```

raise and fail Keyword

The raise and fail keywords are used to stop a Chef Infra Client run in both the compile and execute phases.

```
file '/tmp/name_of_file' do
  action :create
end
raise "message" if
platform?('windows')
package 'name_of_package'
do
  action :install
end
```

rescue Blocks

Since recipes are written in Ruby, they can be used to attempt the handle error conditions with the help of rescue blocks.

```
begin
  dater = data_bag_item(:basket,
'flowers')
rescue Net::HTTPClientException
  # maybe some retry code here?
  raise 'message_to_be_raised'
end
```

Fatal Messages

Fatal Messages notify the user about a failure in the system. A Chef Infra Client run is stopped after a fatal message is sent to the logger and STDERR.

```
Chef::Application.fatal!("log_message",
error_code) if condition
```

Dynamic Recipe Configuration

Dynamic Recipe Configuration

Steps to dynamically configure a recipe are given below:

- Create a default file for cookbook attributes and add attributes
- Define the attribute inside the recipe
- Upload the modified cookbook
- Execute Chef Client of the defined node

Override Attribute at Node and Environment Level

Steps to override attribute at node and environment level are given blow:

- Create a role
- Upload the role to the Chef server
- Assign the role to the node
- Execute the Chef client

ChefSpec

Assisted Practice
Create and Execute ChefSpec

Problem Statement:

You are assigned to create and execute ChefSpec in the given project.

Assisted Practice: Guidelines

Steps to perform:

1. Create a gem file containing ChefSpec gem
2. Install gem
3. Create a spec directory
4. Create a spec
5. Validate the spec
6. Edit cookbook default recipe
7. Create a template file
8. Run rspec again

Key Takeaways

- Cookbook is a fundamental unit of configuration and policy distribution.
- Recipe is a fundamental configuration element within the organization and is a collection of resources defined using patterns.
- Chef Infra Client run can be stopped using return, raise keyword, and fatal messages.