DevOps

Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

DevOps

Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

Advanced Chef

Learning Objectives

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

- Explain the working of Chef templates
- Test the Chef libraries
- Create a cookbook using data bags
- Outline cross platform cookbook
- Implement lightweight resource provider and Chef blueprints

Chef Templates

What Is Template?

- It is an Embedded Ruby (ERB) template used to dynamically generate static text files.
- It contains Ruby expressions and statements that are a great way to manage configuration files.
- Template resource is used to add cookbook templates to recipes and place the corresponding Embedded Ruby (ERB) template file in a cookbook's or template's directory.

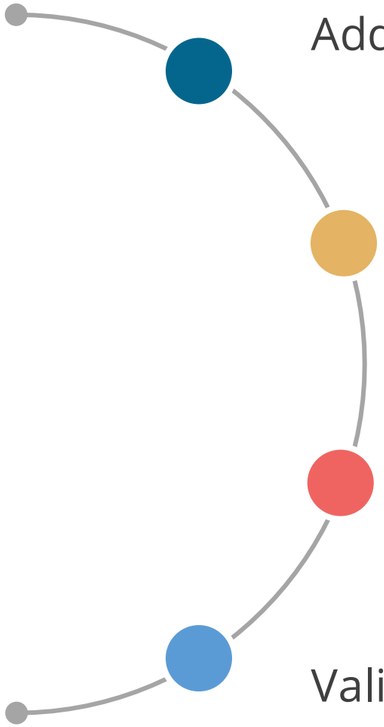
Template Resource Block

Template resource block declares the location in which a file is created, the source template used to create the file, and permissions needed for that file.

Syntax:

```
template '/etc/motd' do
  source 'motd.erb'
  owner 'root'
  group 'root'
  mode '0755'
end
```

Template Usage Approach



Add the template resource to the recipe

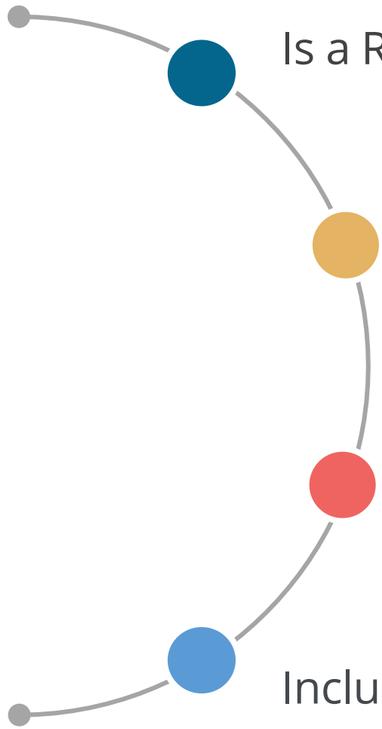
Append ERB template to the cookbook

Upload modified cookbook to the Chef server

Validate the content of the uploaded file

Recipe DSL

Recipe DSL



- Is a Ruby DSL primarily used to declare resources from a recipe
- Ensures that recipes interact with nodes in the desired manner
- Finds a specific parameter to tell Chef Infra Client what actions to take based on node
- Includes custom resources and validates the dependencies

Recipe DSL Methods

Recipe DSL Methods

The Recipe DSL provides support for using attributes, data bags, and search results in a recipe. It also supports four helper methods that can be used to check node's platform.

The four helper methods are:

1. platform?
2. platform_family?
3. value_for_platform?
4. value_for_platform_family?

attribute?

The **attribute?** method ensures that certain actions only execute in the presence of a particular node attribute. It will return true if one of the listed node attributes matches a node attribute detected by Ohai during Client Infra run.

```
attribute?('name_of_attribute')

Example:


if node.attribute?('ipaddress')
  # the node has an ipaddress
end
```

declare_resource

The declare_resource method is used to instantiate a resource and then add it to the resource collection.

Syntax:

```
declare_resource(:resource_type, 'resource_name',
resource_attrs_block)
```

- **resource_type** determines the type of the resource such as file, template, etc.
- **resource_name** refers to the the default name of the resource, typically the string that appears in the resource.
- **resource_attrs_block** is a block in which properties of the instantiated resources are declared.

find_resource

The find_resource method can be used to:

- Find a resource in the resource collection.
- Define a resource block. If a resource block with the same name exists in the resource collection, it will be returned.

```
find_resource(:template,
'/etc/seapower') do
  source 'seapower.erb'
  cookbook 'seapower'
  variables({:seapower => {} })
  notifies :run,
'execute[newseapower]'
end
```

platform?

The platform? method ensures that certain actions run for specific platforms. The platform? method will return true if one of the listed parameters matches the node['platform'] attribute that is detected by Ohai during every Chef Infra Client run.

Syntax:

```
platform?('parameter', 'parameter')
```

- **parameter** is a comma-separated list, each specifying a platform such as Red Hat, CentOS, or Fedora.
- **platform?** method is used with an if, elsif, or case statement that contains Ruby code that is specific for the platform if detected.

Parameters

Parameter	Platforms
aix	AIX. All platform variants of AIX return aix.
amazon	Amazon Linux
arch	Arch Linux
debian	Debian
fedora	Fedora
freebsd	FreeBSD. All platform variants of FreeBSD return freebsd
gentoo	Gentoo

Parameters

Parameter	Platforms
mac_os_x	macOS
netbsd	NetBSD. All platform variants of NetBSD return netbsd
openbsd	OpenBSD. All platform variants of OpenBSD return openbsd
slackware	Slackware
suse	SUSE Enterprise Linux Server
ubuntu	Ubuntu Linux
windows	Microsoft Windows

Basic Conditions of Recipe DSL

Basic Conditions

- To sort the names, Recipe DSL is necessary
- Printing operating system should loop through the nodes
- To construct gem names, install multiple Ruby gems with arrays and strings

Chef Libraries

Chef Library: Working Method

Libraries in Chef provide a place to encapsulate compiled logic so that the cookbook recipes remain neat and clean.

Working Method:

Chef library code opens the Chef Recipe class and adds new methods. A simple way of performing this task is given below:

```
class Chef::Recipe
def netmask(ipaddress)
...
end
end
```

Assisted Practice

Create and Test the Chef Library

Problem Statement:

You are given a project to create and test the Chef library in the Chef environment.

Assisted Practice: Guidelines

Steps to perform:

1. Create a Chef library
2. Use the helper method
3. Upload the modified cookbook to the server
4. Test the library

Best Practices

Once you open the **chef::recipe class,** there are changes that override new session. As a best practice, it is always good to introduce a new subclass inside the library and define a method as class method. This avoids pulling the chef::recipe namespace.

```
/chef-repo $ subl
cookbooks/my_cookbook/libraries/ipaddress.rb
class Chef::Recipe::IPAddress
def self.netmask(ipaddress)
IPAddress(ipaddress).netmask
end
end
```

Data Bags

Data Bags

Chef data bags can be defined as an arbitrary collection of data which one can use with cookbooks. Using data bags is very helpful when one does not wish to hardcode attributes in recipes nor to store attributes in cookbooks.

data bag

Assisted Practice

Create a Data Bag Using Knife

Problem Statement:

You are assigned to create a data bag item using Chef knife method.

Assisted Practice: Guidelines

Steps to perform:

1. Create a Chef library
2. Use the helper method
3. Upload the modified cookbook to the server
4. Test the library

Assisted Practice

Create a Data Bag with Manual Approach

Problem Statement:

You are assigned to create a data bag item using manual method.

Assisted Practice: Guidelines

Steps to perform:

1. Create a data bag item in the Chef-repo
2. Specify the file name to create a data bag item
3. Validate the item

Storing Data in a Data Bag

Storing Data in a Data Bag

When the chef-repo is cloned from GitHub:

- A directory named data_bags is created.
- A sub-directory with the same name as the data bag is created.
- A JSON file is created and placed in the appropriate sub-directory.

Storing Data in a Data Bag

The data_bags directory can be placed under version source control. When deploying from a private repository using a data bag, use the deploy_key option to ensure that the private key is present.

```
{
  'id': 'my_app',
  ... (truncated) ...
  'deploy_key':
'ssh_private_key'
}
```

Data Bag Items

Data Bag Items

It is a container of related data bag items, where each individual data bag item is a JSON file. The only structural requirement of a data bag item is that it must have an id.

```
{
  /* This is a supported comment style */
  // This style is also supported
  "id": "ITEM_NAME",
  "key": "value"
}
```

where:

- key and value are the key:value pair for each additional attribute within the data bag item.
- /* ... */ and // ... show two ways to add comments to the data bag item.

Encrypt a Data Bag Item

- A data bag item is encrypted using shared secret encryption that allows each data bag item to store confidential information.

- Each data bag item can be encrypted individually. If a data bag contains multiple encrypted data bag items, it's not compulsory for these data bag items to share the same encryption keys.

Encryption Versions

The type of data bag item encryption depends on the Chef Infra Client version used. In the snapshot given below, R, W, D stand for read, write, and disable respectively.

src: chef.io

Encryption Versions: Version 0

The sequence of steps occurring in version 0 is given below:

- An encrypted data bag item is written using YAML as the serialization format.
- Base64 encoding is used to preserve special characters in encrypted contents.
- Data is encrypted using AES-256-CBC (as defined by the OpenSSL package in the Ruby Standard Library).
- Chef Infra Client uses shared secret encryption. An encrypted file can only be decrypted by a node or a user with the same shared secret.
- A recipe can load encrypted data as long as the shared secret is present in a file on the node or is accessible from a URI path.
- Only the values of a data bag item are decrypted, whereas keys are still searchable. The values associated with the id key of a data bag item are not encrypted.

Encryption Versions: Version 1

The sequence of steps occurring in version 1 is given below:

- An encrypted data bag item is written using JSON as the serialization format.
- Base64 encoding is used to preserve special characters in encrypted contents.
- Data is encrypted using AES-256-CBC (as defined by the OpenSSL package in the Ruby Standard Library).
- A data bag item is encrypted using a random initialization vector each time a value is encrypted. This protects from some forms of cryptanalysis.
- A recipe can load encrypted data as long as the shared secret is present in a file on the node or is accessible from a URI path.
- Only the values of a data bag item are decrypted, whereas keys are still searchable. The values associated with the id key of a data bag item are not encrypted.

Encryption Versions: Version 2

The sequence of steps occurring in version 2 is given below:

- All the functionalities in version 2 are similar to those of version 1.
- It can be used to disable version 0 and version 1 data bag item encryption formats.
- This also adds Encrypt-then-MAC(EtM) protection.

Editing a Data Bag Item

Editing a Data Bag Item

A Data bag item can be edited in two ways: using knife or by using the Chef management console.

Steps to edit a data bag using Knife:

- Use the edit argument to edit the data in a data bag.
- If encryption is being used, the data bag will be decrypted and the data will be made available in the $EDITOR.
- The data will be encrypted again before saving it to the Chef Infra Server.

Editing a Data Bag Item

Steps to edit a data bag using Chef management console:

1. Open the Chef management console
2. Click **Policy**
3. Click **Data Bags**
4. Select a data bag
5. Select the **Items** tab
6. Select an item
7. Click **Edit**
8. Make the changes
9. Click **Save Item**

Cross-Platform Cookbook

Cross-Platform Cookbooks

- Cross-platform cookbooks are cookbooks that that adopt to the environment on which they are being run.
- Chef helps to write cross-platform cookbooks capable of running on any OS on which it is going to get deployed.
- This helps a developer to write a completely operational cookbook.

Chef Resources

Available Resources

Resource	Description
apt_package	Manages the package for Debian and Ubuntu
Bash	Executes the scripts using Bash interpreter
Batch	Executes batch script using cmd.exe interpreter
bff_package	Manages the AIX platform using installp utility
chef_gem	Installs gem only instance for Ruby
File	Manages the files directly on the node
Group	Manages the local group

Chef Automate

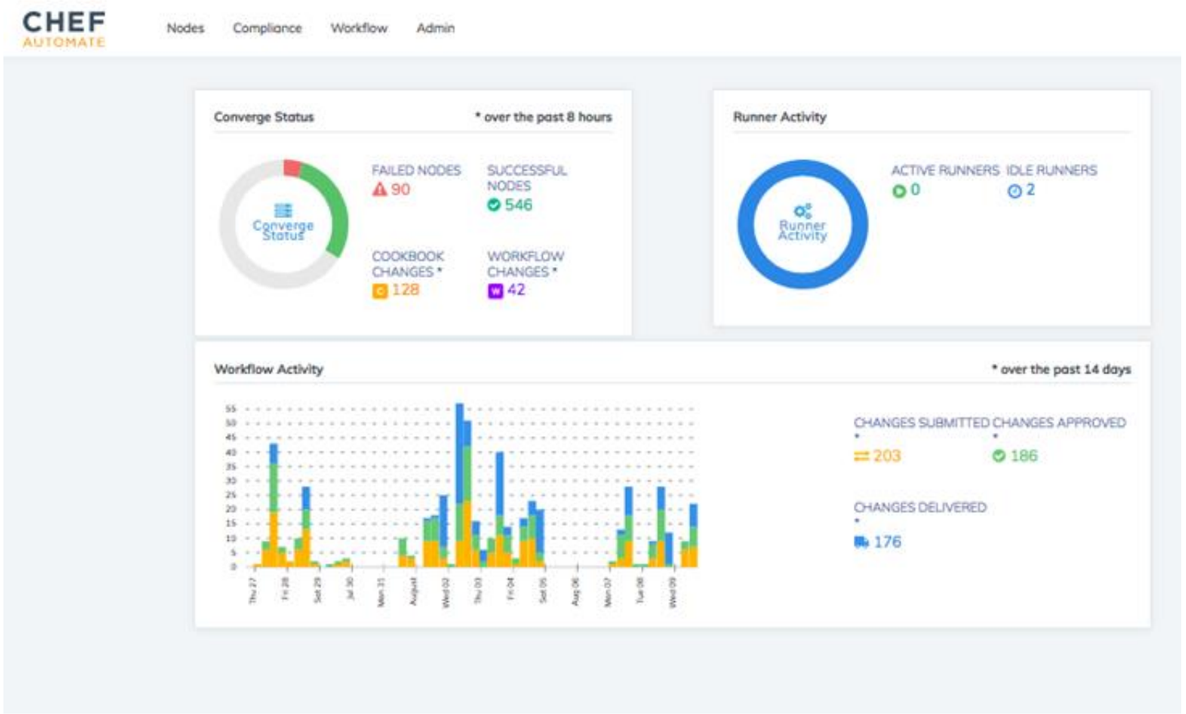
Chef Automate

- Chef Automate provides a full suite of enterprise capabilities for node visibility and compliance.
- It integrates with the open-source products Chef Infra Client, Chef InSpec, and Chef Habitat.
- It also comes with comprehensive 24x7 support services for the entire platform, including open source components.
- It provides a full-stack continuous compliance and security, and a visibility into your applications and infrastructure.

Chef Automate

Chef Automate provides a data warehouse that accepts input from Chef Server, Chef Habitat, and Chef Automate workflow and compliance. It provides views into operational and workflow events. There is a query language available through the UI and customizable dashboards.

Chef Automate dashboard screenshot is given below:



src: chef.io

Key Takeaways

- Template contains Ruby expressions and statements that are a great way to manage configuration files.
- A data bag item is encrypted using shared secret encryption that allows each data bag item to store confidential information.
- Chef Automate provides a full suite of enterprise capabilities for node visibility and compliance.