EXTENDING CHEF BADGE TOPICS

The Extending Chef badge is awarded when someone proves that they understand how to extend and customize Chef. Candidates must show:

- An understanding of how to extend OHAI.
- An understanding of custom resources.
- An understanding of Chef handlers.
- An understanding of definitions and libraries.
- An understanding of knife plugins.
- An understanding of the Chef API.
- An understanding of Ruby gems.

Here is a detailed breakdown of each area.

EXTENDING OHAI

BASIC OHAI PLUGIN AUTHORING

Candidates should understand:

Platform distinctions
Attribute structure
Ohai DSL syntax
How to collect attributes on Linux vs Windows
How to collect a kernel setting as an attribute
How to format STDOUT into a nested hash?
Dedicated namespaces

OHAI COOKBOOK

Candidates should understand:

How to deploy Ohai plugins
How to use the Ohai community cookbook (v4.2.0 and above)
How to use the 'ohai_plugin' resource
How to load Ohai at the start of a chef-client run
How to reload Ohai during a chef-client run
How to configure the plugin path

TROUBLESHOOTING PLUGINS

Candidates should understand:

What happens when an Ohai plugin fails
Monitoring log messages
How to use IRB or chef-shell
What Ohai hints are
How to write Ohai hints

ENABLING & DISABLING OHAI PLUGINS

Candidates should understand:

How to enable a plugin
How to disable a plugin
The client.rb settings

Whitelisting & blacklisting attributes

Candidates should understand:
How to reduce Ohai content
The 'automatic_attribute_whitelist' precedence levels
The ramifications of whitelisting

CUSTOM RESOURCES

Why/when to use custom resources

Candidates should understand:
What a custom resource is
Why/when to use a custom resource
When custom resources are appropriate and when libraries are more appropriate
How to use custom resources to restrict tunable data
How to use custom resources to reduce recipe size and to abstract out code

Custom resource DSL

Candidates should understand:
How to write custom resources
DSL components
Methods, actions, properties, and resource names.
How to create custom resource properties and property validation parameters
How to have multiple actions
How to assign default actions

Sharing resources

Candidates should understand:
How to share a custom resource
How to manage dependencies

Nested resource collections

Candidates should understand:
How to nest custom resources
What a resource collection is
The 'use inline resources' directive
If notifications get passed up to the parent resource collection
What happens if a resource in the 'child' resource collection calls for an immediate restart

CHEF HANDLERS

Handler types

Candidates should understand:

What the three types of handlers are
When these handlers are run
What the 'run_status' is and how to use it
What handler runs when 'run_status.success?' is true
What handler runs when 'run_status.failed?' is true

CHEF_HANDLER COOKBOOK

Candidates should understand:
What the purpose of the chef_handler_cookbook is
What the 'chef_handler' resource is
What arguments the 'chef_handler' resource takes

HANDLER DSL

Candidates should understand:
What the Chef handler DSL is
What event types are
What the 'Chef::Handler' class is
Chef handler DSL code

DISTRIBUTING HANDLERS

Candidates should understand:
How handler code gets delivered to a node
How to configure handlers in the 'client.rb' file

DEFINITIONS & LIBRARIES

USING LIBRARIES

Candidates should understand:
What a library is
Where libraries are stored
When to use a library
How to use a library
How to write helper files
How to share libraries

CREATING RESOURCES

Candidates should understand:
How to use libraries to write resources
What 'Chef::Resource::Base' is
What 'Chef::Provider::Base' is

MODIFYING CORE CLASSES

Candidates should understand:
How to use libraries to extend core components of Chef client

Using Definitions

Candidates should understand:

What a definition is

What the pitfalls of using definitions are

Custom Resources vs Libraries

Candidates should understand:

What the difference is in writing a resource with the custom resource DSL versus using raw Ruby

When to write a resource in pure Ruby

KNIFE PLUGINS

Knife Source Code

Candidates should understand:

How to inherit knife plugins?

What the 'Chef::Knife' superclass is

When to use the 'Chef::Knife' superclass

Why you would use the 'Chef::Knife' superclass

Knife plugin use cases

Candidates should understand:

Why/when to use a knife plugin for a private cloud

Common knife plugins

CHEF API

When to use API

Candidates should understand:

How to communicate with Chef server

What languages you can use with the Chef API

How to authenticate with Chef server using the API

When to make organization-specific API queries vs. Chef server global queries

How to timestamp API requests

Common API endpoints

JUST ENOUGH RUBY TO CUSTOMIZE CHEF

Creating Ruby Gems

Candidates should understand:

How to create a Ruby gem consisting of custom plugin code

What the ‘chef gem ...’ command does

Managing Ruby Gems

Candidates should understand:

How to manage Ruby gems