BASIC CHEF FLUENCY BADGE TOPICS

The Basic Chef Fluency badge is awarded when someone proves that they understand the core elements that underpin Chef. Candidates must show:

- An understanding of basic Chef terminology.
- An understanding of Chef product offerings.
- An understanding of Chef's design philosophy.
- An understanding of Chef's approach to workflow and compliance.
- An understanding of basic Chef coding skills.

Here is a detailed breakdown of each area.

CHEF BASIC TERMINOLOGY

RESOURCES

Candidates should understand:

Idempotency/convergence - test & repair model
Common resources and their actions
Default actions **(QuestionNum:1)(In Docs?:Yes)(In Training?:Yes)**
The ':nothing' action
The 'supports' directive
The 'not_if' and 'only_if' directives **(QuestionNum:2)(In Docs?:Yes)(In Training?:No)**
Resource extensibility **(QuestionNum:3)(In Docs?:Yes)(In Training?:Yes)**

RECIPES

Candidates should understand:

What a recipe is
Importance of the resource order **(QuestionNum:6)(In Docs?:Yes)(In Training?:Yes)**
How to use 'include_recipe' **(QuestionNum:8)(In Docs?:Yes)(In Training?:Yes)**
What happens if a recipe is included multiple times in a run_list **(QuestionNum:9)(In Docs?:Yes)(In Training?:No)**
The 'notifies' and 'subscribes' directives **(QuestionNum:7)(In Docs?:Yes)(In Training?:No)**

COOKBOOKS

Candidates should understand:

Cookbook contents **(QuestionNum:11)(In Docs?:Yes)(In Training?:Yes)**
Naming conventions
Cookbook dependencies **(QuestionNum:12)(In Docs?:Yes)(In Training?:Yes)**
The default recipe **(QuestionNum:13)(In Docs?:Yes)(In Training?:Yes)**

CHEF SERVER

Candidates should understand:

How the Chef server acts as an artifact repository
How the Chef server acts as an index of node data
Chef solo vs Chef server **(QuestionNum:17)(In Docs?:Yes)(In Training?:No)**
Chef server's distributed architecture **(QuestionNum:16)(In Docs?:Yes)(In Training?:Yes)**

Scalability **(QuestionNum:18)(In Docs?:Yes)(In Training?:No)**

Search

Candidates should understand:

What search is**(QuestionNum:21)(In Training?:Yes)**
How to search for node information
What and how many search indexes Chef server maintains **(QuestionNum:23)(In Docs?:Yes)(In Training?:No)**
What a databag is
How to use search for dynamic orchestration
How to invoke a search from the command line **(QuestionNum:22)(In Docs?:Yes)(In Training?:Yes)**

Chef client

Candidates should understand:

What the Chef client is **(QuestionNum:28)(In Docs?:Yes)(In Training?:No)**
The function of Chef client vs the function of Chef server
What 'why-run' is
How to use '--local-mode'
How the Chef client and the Chef server communicate **(QuestionNum:26)(In Training?:No)**
The Chef client configuration **(QuestionNum:27)(In Docs?:Yes)(In Training?:Yes)**

Nodes

Candidates should understand:

What a node is
What a node object is
How a node object is stored on Chef server **(QuestionNum:33)**
How to manage nodes **(QuestionNum:31)(In Docs?:Yes)(In Training?:No)**
How to query nodes
How to name nodes **(QuestionNum:32)(In Docs?:Yes)(In Training?:No)**

Run List

Candidates should understand:

What a run_list is **(QuestionNum:38)(In Docs?:Yes)(In Training?:Yes)**
What nested run_lists are**(QuestionNum:36)(In Docs?:Yes)(In Training?:No)**
Where a run_list is stored
What does a run_list consist of
How to look up run_lists
How to set and change run_lists **(QuestionNum:37)(In Docs?:Yes)(In Training?:No)**

Roles

Candidates should understand:

What roles are
How a role ensures code consistency across nodes
Where roles can be stored **(QuestionNum:42)(In Docs?:Yes)(In Training?:Yes)**
How roles are defined
What the components of a role are **(QuestionNum:41)(In Docs?:Yes)(In Training?:Yes)**
Roles vs recipes vs run_lists
How to name roles

How to apply roles to nodes
How to edit roles
(QuestionNum:43)(In Docs?:Yes)(In Training?:No)

Environments

Candidates should understand:
The purpose of environments **(QuestionNum:47)(In Docs?:Yes)(In Training?:Yes)**
How to use environments to manage cookbook release cycles **(QuestionNum:46)(In Docs?:Yes)(In Training?:Yes)**
How to use environments to constrain cookbooks **(QuestionNum:48)(In Docs?:Yes)(In Training?:Yes)**
How to put nodes into an environment

Infrastructure as code

Candidates should understand:
What the advantages are of defining infrastructure as code **(QuestionNum:51)(In Docs?:Yes)(In Training?:Yes)**
The reasons for defining infrastructure as code **(QuestionNum:52)(In Docs?:Yes)(In Training?:No)**
The difference between rolling back and rolling forward **(QuestionNum:53)(In Docs?:Yes)**

Desired State Configuration

Candidates should understand:
The imperative vs the declarative approach to configuration management **(QuestionNum:57)(In Docs?:No)(In Training?:No)**
The push vs the pull approach **(QuestionNum:56)(In Docs?:Yes)(In Training?:Yes)**
What Windows DSC is
What happens if you remove a resource from a recipe **(QuestionNum:58)(In Docs?:Yes)(In Training?:No)**

Supermarket

Candidates should understand:
The Supermarket value proposition
What you can store in Supermarket**? (QuestionNum:62)(In Docs?:Yes)(In Training?:No)**
What a private Supermarket is
When to use a private Supermarket **(QuestionNum:61)(In Docs?:Yes)(In Training?:No)**
If Supermarket is a free or a premium feature **(QuestionNum:63)(In Docs?:Yes)(In Training?:No)**
If the items in Supermarket are free or cost money

Chef DK

Candidates should understand:
The Chef DK value proposition
Specific features of test-driven development (TDD)
Tools packaged in Chef DK

Test Kitchen

Candidates should understand:
The Test Kitchen value proposition

What TDD is
The platforms supported by Test Kitchen
How to include Test Kitchen in a pipeline
Basic `kitchen` commands **(QuestionNum:72)(In Docs?:Yes)(In Training?:Yes)**
Basic `kitchen` configuration **(QuestionNum:71)(In Docs?:Yes)(In Training?:Yes)**

DESCRIBING WHAT CHEF IS

CHEF COMMUNITY

Candidates should understand:
The open-source nature of Chef
The Chef community
What the open source tools are

PRODUCTS AND FEATURES

Candidates should understand:
The Chef Automate value proposition
The Chef Automate features **(QuestionNum:83)(In Docs?:Yes)(In Training?:Yes)**
What the workflow feature is and how it affects productivity
What the compliance feature is and how it affects workflow
What the visibility feature is and how it affects workflow
How a private Supermarket fits into a workflow **(QuestionNum:84)**
The Chef Automate open source components **(QuestionNum:77)**
What Visibility is **(QuestionNum:78)**
What Habitat is
What InSpec is **(QuestionNum:76)**
What Chef Compliance is **(QuestionNum:79)**

END-TO-END WORKFLOW

Candidates should understand:
How all Chef products, features and technologies fit together
The workflow scope **(QuestionNum:86)**
The compliance scope **(QuestionNum:88)(In Docs?:Yes)(In Training?:No)**
The Chef Automate scope **(QuestionNum:87)(In Docs?:Yes)(In Training?:Yes)**
How Chef Automate enhances DevOps behaviors
The aspects of Chef that are relevant to security and compliance teams
The aspects of Chef that are relevant to development teams
The aspects of Chef that are relevant to operations teams
The aspects of Chef that are relevant to change advisory boards
How Chef enforces consistency across infrastructure

DESIGN PHILOSOPHY

CHEF IS WRITTEN IN RUBY

Candidates should understand:
How Chef uses a Ruby-based DSL
How to use raw Ruby to extend Chef **(QuestionNum:91, 92)(In Docs?:Yes)(In Training?:No)**
What a library is

Explicit actions

Candidates should understand:

How Chef uses explicit actions and only does what you tell it to
Actions for common resources such as **(QuestionNum:98)(In Docs?:Yes)(In Training?:No)**
the :nothing action **(QuestionNum:97)(In Docs?:Yes)(In Training?:No)**
What it means to roll back infrastructure **(QuestionNum:96)(In Docs?:Yes)(In Training?:Yes)**
What happens if you reverse the order of resources in a recipe
If Chef can automagically detect what patches should be applied to a system

Push vs. Pull

Candidates should understand:

The difference between push and pull models
The benefits of a pull model **(QuestionNum:103)(In Docs?:Yes)(In Training?:Yes)**
When a push model is appropriate
What firewall rules need to be enabled for Chef client **(QuestionNum:101)(In Docs?:Yes)(In Training?:No)**
The Chef client converge intervals and how to invoke immediate updates **(QuestionNum:102)(In Docs?:Yes)(In Training?:No)**

Recommended workflows

Candidates should understand:

What wrapper cookbooks are **(QuestionNum:111)(In Docs?:Yes)(In Training?:Yes)**
How to use source control, e.g. GitHub **(QuestionNum:112)(In Docs?:??)(In Training?:??)**
How to use the TDD approach **(QuestionNum:113)(In Docs?:Yes)(In Training?:No)**

CHEF WORKFLOW BASICS

Continuous Delivery

Candidates should understand:

What continuous delivery (CD) is **(QuestionNum:117, 118)(In Docs?:No)(In Training?:No)**
What role Chef plays in CD
When to run tests **(QuestionNum:116)(In Docs?:Yes)(In Training?:Yes)**
Why automated configuration management is critical to CD
Why CD is *more* secure than manual processes

Using Compliance to scan

Candidates should understand:

The benefits of the agentless nature of Chef compliance
How to check for compliance on nodes that don't have the Chef client installed**(QuestionNum:123)**
Basic use cases for compliance **(QuestionNum:122)(In Docs?:Yes)(In Training?:No)**
What language is used to express compliance requirements **(QuestionNum:121)**

Using Chef DK to test your changes

Candidates should understand:

The Test Kitchen value proposition **(QuestionNum:126, 128)**
Basic use cases for Chef DK

Publishing Artifacts to Chef Server and Supermarket

Candidates should understand:

How to publish artifacts to Chef server **(QuestionNum:136, 138)**

What Berkshelf is

If the Chef Automate workflow feature can push artifacts to things other than a Chef server or Supermarket

How to manage cookbook dependencies **(QuestionNum:137)**

UNDERSTANDING BASIC CHEF CODE

Approachable custom code

Candidates should understand:

How to recognizing custom code **(QuestionNum:141)**

How to use libraries **(QuestionNum:142)**

How to customize Chef**(QuestionNum:143)**

Approachable Chef code

Candidates should understand:

How to read a recipe that includes the 'package', 'file', and 'service' resources and describe its intent. **(QuestionNum:146, 147)**