**About Cookbooks**

A cookbook is the fundamental unit of configuration and policy distribution. A cookbook defines a scenario and contains everything that is required to support that scenario:

* Recipes that specify the resources to use and the order in which they are to be applied
* Attribute values
* File distributions
* Templates
* Extensions to Chef, such as custom resources and libraries

The chef-client uses Ruby as its reference language for creating cookbooks and defining recipes, with an extended DSL for specific resources. The chef-client provides a reasonable set of resources, enough to support many of the most common infrastructure automation scenarios; however, this DSL can also be extended when additional resources and capabilities are required.

**Components**

Some important components of cookbooks include:

| **Feature** | **Description** |
| --- | --- |
| [Attributes](https://docs.chef.io/attributes.html) | An attribute can be defined in a cookbook (or a recipe) and then used to override the default settings on a node. When a cookbook is loaded during a chef-client run, these attributes are compared to the attributes that are already present on the node. Attributes that are defined in attribute files are first loaded according to cookbook order. For each cookbook, attributes in the default.rb file are loaded first, and then additional attribute files (if present) are loaded in lexical sort order. When the cookbook attributes take precedence over the default attributes, the chef-client will apply those new settings and values during the chef-client run on the node. |
| [Recipes](https://docs.chef.io/recipes.html) | A recipe is the most fundamental configuration element within the organization. A recipe:   * Is authored using Ruby, which is a programming language designed to read and behave in a predictable manner * Is mostly a collection of resources, defined using patterns (resource names, attribute-value pairs, and actions); helper code is added around this using Ruby, when needed * Must define everything that is required to configure part of a system * Must be stored in a cookbook * May be included in a recipe * May use the results of a search query and read the contents of a data bag (including an encrypted data bag) * May have a dependency on one (or more) recipes * May tag a node to facilitate the creation of arbitrary groupings * Must be added to a run-list before it can be used by the chef-client * Is always executed in the same order as listed in a run-list |

The chef-client will run a recipe only when asked. When the chef-client runs the same recipe more than once, the results will be the same system state each time. When a recipe is run against a system, but nothing has changed on either the system or in the recipe, the chef-client won’t change anything.

In addition to attributes and recipes, the following items are also part of cookbooks:

| **Components** | **Description** |
| --- | --- |
| [Definitions](https://docs.chef.io/definitions.html) | A definition is used to create new resources by stringing together one (or more) existing resources. |
| [Files](https://docs.chef.io/files.html) | A file distribution is a specific type of resource that tells a cookbook how to distribute files, including by node, by platform, or by file version. |
| [Libraries](https://docs.chef.io/libraries.html) | A library allows the use of arbitrary Ruby code in a cookbook, either as a way to extend the chef-client language or to implement a new class. |
| [Custom Resources](https://docs.chef.io/custom_resources.html) | A custom resource is an abstract approach for defining a set of actions and (for each action) a set of properties and validation parameters. |
| [Metadata](https://docs.chef.io/cookbook_repo.html) | A metadata file is used to ensure that each cookbook is correctly deployed to each node. |
| [Resources](https://docs.chef.io/resource.html) | A resource instructs the chef-client to complete various tasks like installing packages, running Ruby code, or accessing directories and file systems. The chef-client includes built-in resources that cover many common scenarios. For the full list of resources that are built-in to the chef-client, see /resources.html. |
| [Templates](https://docs.chef.io/templates.html) | A template is a file written in markup language that uses Ruby statements to solve complex configuration scenarios. |
| [Cookbook Versions](https://docs.chef.io/cookbook_versions.html) | A cookbook version represents a set of functionality that is different from the cookbook on which it is based. A version may exist for many reasons, such as ensuring the correct use of a third-party component, updating a bug fix, or adding an improvement. A cookbook version is defined using syntax and operators, may be associated with environments, cookbook metadata, and/or run-lists, and may be frozen (to prevent unwanted updates from being made).  A cookbook version is maintained just like a cookbook, with regard to source control, uploading it to the Chef server, and how the chef-client applies that cookbook when configuring nodes. |

**Community Cookbooks**

Chef maintains a large collection of cookbooks. In addition, there are thousands of cookbooks created and maintained by the community:

| **Components** | **Description** |
| --- | --- |
| [Cookbooks Maintained by Chef](https://github.com/chef-cookbooks) | Chef maintains a collection of cookbooks that are widely used by the community. |
| [Cookbooks Maintained by the Community](https://supermarket.chef.io/cookbooks) | The community has authored thousands of cookbooks, ranging from niche cookbooks that are used by only a few organizations to cookbooks that are some of the most popular and are used by nearly everyone. |