**Chef Components**

The Chef process consists of three core components that interact with one another: *Chef server*, actual servers called *nodes*, and *Chef workstation*.

**Chef Server**

The Chef server is as a hub for configuration data. It stores cookbooks, the policies that are applied to nodes, and metadata that describes each node managed by Chef.

**Nodes**

Nodes use a tool called *Chef client* to ask the Chef server for configuration details and then applies them to itself. This process of applying changes on nodes is called a *Chef run*.

Cookbooks are made of one or more recipes that will perform automated steps, called actions, such as installing and configuring applications or creating files. All data needed in this process is downloaded from the Chef server as the first step in a Chef run.

Note that the Chef client is installed within the bootstrap process, when a node is created and registered on a Chef server.

Each node can have one or more roles. A role defines a set of attributes (node specific settings) and a list of recipes for a node. It can be reused by multiple nodes, so you can have a cluster of nodes with the same role in the system. The computed list of recipes associated with a node via role or as dependency of other recipes is called a **run list**, and it is executed in the same order it’s defined.

**Chef Workstation**

The Chef workstation or *Chef repository*, shortly chef-repo, is the project structure of a Chef-managed project and it is used on a developer workstation. All Chef components are defined in it: cookbooks, environments, roles and a test suite. It is a good and common practise to keep the chef-repo in a version control system and manage the same way you would manage the source code of your application.

The exact directory structure of a chef-repo varies. Some people prefer to keep all their cookbooks in a single chef-repo. Others prefer to use a separate repository for each cookbook. In both situations your main chef-repo is the place from where you manage your infrastructure, and it should contain information about all of its parts.

**An Overview of Chef Components**

Here is a short summary of main parts of Chef architecture:

* **Chef Server**: Centralized server that holds all of your nodes’ configuration. It can be [self-hosted](https://downloads.chef.io/chef-server/) or [hosted by Chef](https://www.chef.io/chef/choose-your-version/) (the company).
* **Node**: Hosts to which recipes and roles are applied during Chef client run. The primary features of a node are its attributes and run list.
* **Cookbooks**: Contain all resources and instructions that you need to configure your nodes. These can be reused across different run lists. Cookbooks typically consist of many recipes.
* **Recipe**: The fundamental part of Chef, it is a collection of resources that are executed in the order to configure a node.
* **Resource**: A cross platform abstraction of configurable parts of a node. For example these could be users, packages, files or directories.
* **Attributes** – Represent node settings, for example hostname, versions of programming languages to install, database server etc.
* **Data bags**: Contain globally available data used by nodes and roles.
* **Chef Client**: Does all work on behalf of a node, where it executes recipes to configure and install software.
* **Chef Repository**: The place where cookbooks, roles, configuration files, and other artifacts live.
* **Chef Solo**: A command line tool that allows you to run Chef cookbooks without an actual Chef server. It is an open source version of the Chef client.
* **Knife**: A tool used by engineers to upload configuration changes to the Chef server.
* **Ohai**: A tool for collecting data about your operating system, used to provide system attributes used by Chef client during Chef run.
* **Role**: A way to group similar features of nodes, for easier management.