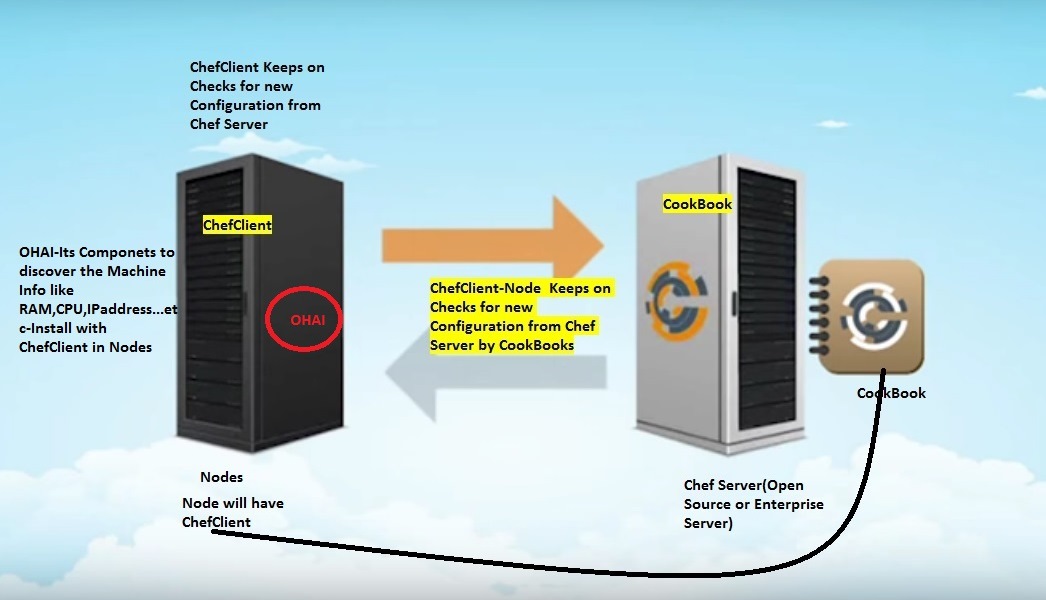
**Chef Server**

The Chef server acts as **a hub for configuration data**. The Chef server stores **cookbooks, the policies that are applied to nodes, and metadata that describes each registered node that is being managed by the chef-client**. Nodes use the chef-client to ask the Chef server for configuration details, such as recipes, templates, and file distributions. The chef-client then does as much of the configuration work as possible on the nodes themselves (and not on the Chef server). This scalable approach distributes the configuration effort throughout the organization.



| **Feature** | **Description** |
| --- | --- |
|  | Search indexes allow queries to be made for any type of data that is indexed by the Chef server, including data bags (and data bag items), environments, nodes, and roles. A defined query syntax is used to support search patterns like exact, wildcard, range, and fuzzy. A search is a full-text query that can be done from several locations, including from within a recipe, by using the search subcommand in knife, the search method in the Recipe DSL, the search box in the Chef management console, and by using the /search or /search/INDEX endpoints in the Chef server API. The search engine is based on Apache Solr and is run from the Chef server. |
|  | Chef management console is a web-based interface for the Chef server that provides users a way to manage the following objects:   * Nodes * Cookbooks and recipes * Roles * Stores of JSON data (data bags), including encrypted data * Environments * Searching of indexed data * User accounts and user data for the individuals who have permission to log on to and access the Chef server |
|  | A data bag is a global variable that is stored as JSON data and is accessible from a Chef server. A data bag is indexed for searching and can be loaded by a recipe or accessed during a search. |
|  | Policy defines how business and operational requirements, processes, and production workflows map to objects that are stored on the Chef server. Policy objects on the Chef server include roles, environments, and cookbook versions. |

**Policy**

Policy maps business and operational requirements, process, and workflow to settings and objects stored on the Chef server:

* Roles define server types, such as “web server” or “database server”
* Environments define process, such as “dev”, “staging”, or “production”
* Certain types of data—passwords, user account data, and other sensitive items—can be placed in data bags, which are located in a secure sub-area on the Chef server that can only be accessed by nodes that authenticate to the Chef server with the correct SSL certificates
* The cookbooks (and cookbook versions) in which organization-specific configuration policies are maintained

Some important aspects of policy include:

| **Feature** | **Description** |
| --- | --- |
|  | A role is a way to define certain patterns and processes that exist across nodes in an organization as belonging to a single job function. Each role consists of zero (or more) attributes and a run-list. Each node can have zero (or more) roles assigned to it. When a role is run against a node, the configuration details of that node are compared against the attributes of the role, and then the contents of that role’s run-list are applied to the node’s configuration details. When a chef-client runs, it merges its own attributes and run-lists with those contained within each assigned role. |
|  | An environment is a way to map an organization’s real-life workflow to what can be configured and managed when using Chef server. Every organization begins with a single environment called the \_default environment, which cannot be modified (or deleted). Additional environments can be created to reflect each organization’s patterns and workflow. For example, creating production, staging, testing, and development environments. Generally, an environment is also associated with one (or more) cookbook versions. |
|  | 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. |
|  | A run-list defines all of the information necessary for Chef to configure a node into the desired state. A run-list is:   * An ordered list of roles and/or recipes that are run in the exact order defined in the run-list; if a recipe appears more than once in the run-list, the chef-client will not run it twice * Always specific to the node on which it runs; nodes may have a run-list that is identical to the run-list used by other nodes * Stored as part of the node object on the Chef server * Maintained using knife, and then uploaded from the workstation to the Chef server, or is maintained using the Chef management console |