**Organizations and Groups**

**Organization is nothing but maintenable infrastructure unit and logical classification of infra(projects).**

The Chef server uses role-based access control (RBAC) to restrict access to objects—nodes, environments, roles, data bags, cookbooks, and so on. This ensures that only authorized user and/or chef-client requests to the Chef server are allowed. Access to objects on the Chef server is fine-grained, allowing access to be defined by object type, object, group, user, and organization. The Chef server uses permissions to define how a user may interact with an object, after they have been authorized to do so.

The Chef server uses organizations, groups, and users to define role-based access control:

| **Feature** | **Description** |
| --- | --- |
| Organisation | An organization is the **top-level entity for role-based access control in the Chef server**. Each organization contains the default groups (admins, clients, and users, plus billing\_admins for the hosted Chef server), at least one user and at least one node (on which the chef-client is installed). The Chef server supports multiple organizations. The Chef server includes a single default organization that is defined during setup. Additional organizations can be created after the initial setup and configuration of the Chef server. |
| [Group](https://docs.chef.io/_images/icon_server_groups.svg) | A group is used to define access to object types and objects in the Chef server and also to assign permissions that determine what types of tasks are available to members of that group who are authorized to perform them. Groups are configured per-organization.  Individual users who are members of a group will inherit the permissions assigned to the group. The Chef server includes the following default groups: admins, clients, and users. For users of the hosted Chef server, an additional default group is provided: billing\_admins. |
| [user](https://docs.chef.io/_images/icon_server_users.svg) | A user is any non-administrator human being who will manage data that is uploaded to the Chef server from a workstation or who will log on to the Chef management console web user interface. The Chef server includes a single default user that is defined during setup and is automatically assigned to the admins group. |
| [Chef client](https://docs.chef.io/_images/icon_chef_client.svg) | A client is an actor that has permission to access the Chef server. A client is most often a node (on which the chef-client runs), but is also a workstation (on which knife runs), or some other machine that is configured to use the Chef server API. Each request to the Chef server that is made by a client uses a private key for authentication that must be authorized by the public key on the Chef server. |

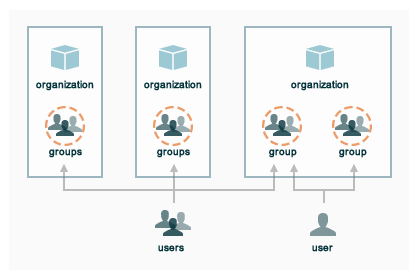
When a user makes a request to the Chef server using the Chef server API, permission to perform that action is determined by the following process:

1. Check if the user has permission to the object type
2. If no, recursively check if the user is a member of a security group that has permission to that object
3. If yes, allow the user to perform the action

Permissions are managed using the Chef management console add-on in the Chef server web user interface.

## Multiple Organizations

A single instance of the Chef server can support many organizations. Each organization has a unique set of groups and users. Each organization manages a unique set of nodes, on which a chef-client is installed and configured so that it may interact with a single organization on the Chef server.



A user may belong to multiple organizations under the following conditions:

* Role-based access control is configured per-organization
* For a single user to interact with the Chef server using knife from the same chef-repo, that user may need to edit their knife.rb file prior to that interaction

Using multiple organizations within the Chef server ensures that the same toolset, coding patterns and practices, physical hardware, and product support effort is being applied across the entire company, even when:

* Multiple product groups must be supported—each product group can have its own security requirements, schedule, and goals
* Updates occur on different schedules—the nodes in one organization are managed completely independently from the nodes in another
* Individual teams have competing needs for object and object types—data bags, environments, roles, and cookbooks are unique to each organization, even if they share the same name

## Permissions

Permissions are used in the Chef server to define how users and groups can interact with objects on the server. Permissions are configured per-organization.

### Object Permissions

The Chef server includes the following object permissions:

| **Permission** | **Description** |
| --- | --- |
| **Delete** | Use the **Delete** permission to define which users and groups may delete an object. This permission is required for any user who uses the knife [object] delete [object\_name] argument to interact with objects on the Chef server. |
| **Grant** | Use the **Grant** permission to define which users and groups may configure permissions on an object. This permission is required for any user who configures permissions using the **Administration** tab in the Chef management console. |
| **Read** | Use the **Read** permission to define which users and groups may view the details of an object. This permission is required for any user who uses the knife [object] show [object\_name] argument to interact with objects on the Chef server. |
| **Update** | Use the **Update** permission to define which users and groups may edit the details of an object. This permission is required for any user who uses the knife [object] edit [object\_name] argument to interact with objects on the Chef server and for any chef-client to save node data to the Chef server at the conclusion of a chef-client run. |

### Global Permissions

The Chef server includes the following global permissions:

| **Permission** | **Description** |
| --- | --- |
| **Create** | Use the **Create** global permission to define which users and groups may create the following server object types: cookbooks, data bags, environments, nodes, roles, and tags. This permission is required for any user who uses the knife [object] create argument to interact with objects on the Chef server. |
| **List** | Use the **List** global permission to define which users and groups may view the following server object types: cookbooks, data bags, environments, nodes, roles, and tags. This permission is required for any user who uses the knife [object] list argument to interact with objects on the Chef server. |

These permissions set the default permissions for the following Chef server object types: clients, cookbooks, data bags, environments, groups, nodes, roles, and sandboxes.

### Default Groups

The following sections show the default permissions assigned by the Chef server to the admins, billing\_admins, clients, and users groups.

Note

The creator of an object on the Chef server is assigned create, delete, grant, read, and update permission to that object.

#### admins

The admins group is assigned the following:

| **Group** | **Create** | **Delete** | **Grant** | **Read** | **Update** |
| --- | --- | --- | --- | --- | --- |
| admins | yes | yes | yes | yes | yes |
| clients | yes | yes | yes | yes | yes |
| users | yes | yes | yes | yes | yes |

#### billing\_admins

The billing\_admins group is assigned the following:

| **Group** | **Create** | **Delete** | **Read** | **Update** |
| --- | --- | --- | --- | --- |
| billing\_admins | no | no | yes | yes |

#### clients

The clients group is assigned the following:

| **Object** | **Create** | **Delete** | **Read** | **Update** |
| --- | --- | --- | --- | --- |
| clients | no | no | no | no |
| cookbooks | no | no | yes | no |
| cookbook\_artifacts | no | no | yes | no |
| data | no | no | yes | no |
| environments | no | no | yes | no |
| nodes | yes | no | yes | no |
| organization | no | no | yes | no |
| policies | no | no | yes | no |
| policy\_groups | no | no | yes | no |
| roles | no | no | yes | no |
| sandboxes | no | no | no | no |