**1. Get the chef-client cookbook**

In this part, you'll get the chef-client cookbook from Chef Supermarket and upload the chef-client cookbook and its dependent cookbooks to your Chef server.

There are multiple ways to set up chef-client to run on a regular interval. On Linux nodes, you might use a daemon, cron job, or service. On Windows, you might use a scheduled task.

You can of course manually set chef-client to run on a regular interval. Another way is to use the [chef-client](https://supermarket.chef.io/cookbooks/chef-client/) cookbook from [Chef Supermarket](https://supermarket.chef.io/). Earlier in this module, you obtained starter code from GitHub. Chef Supermarket is also a place for the community to share cookbooks. You'll work more with community cookbooks in future modules.On Linux, the chef-client cookbook sets up chef-client to run as a service..

There are several ways to obtain cookbooks from Chef Supermarket. One way is to use the [knife supermarket](https://docs.chef.io/plugin_knife_supermarket.html) command. However, the chef-client cookbook has *dependencies* on other cookbooks and knife supermarket does not resolve these dependencies for you.

[Berkshelf](http://berkshelf.com) is a tool that helps you resolve cookbook dependencies. Berkshelf can retrieve the cookbooks that your cookbook depends on and can upload your cookbooks to your Chef server. Berkshelf comes with the Chef DK.

To get started, first ensure you're in the ~/learn-chef directory.

|  |
| --- |
| cd ~/learn-chef |

Next, you need to create a configuration file that tells Berkshelf which cookbooks you want and where they're located. From your ~/learn-chef directory, create a file named Berksfile and add these contents.

|  |
| --- |
| source 'https://supermarket.chef.io'  cookbook 'chef-client' |

Chef provides a public Chef Supermarket site at [https://supermarket.chef.io](https://supermarket.chef.io/). You can also manage your own [private Chef Supermarket](https://docs.chef.io/supermarket.html#private-supermarket) server. Your Berksfile specifies that you want the chef-client cookbook and to pull cookbooks from the public Chef Supermarket server.

The next step is to run berks install to download the chef-client cookbook and its dependencies.

|  |
| --- |
| berks install  Resolving cookbook dependencies...Fetching cookbook index from https://supermarket.chef.io...Installing chef-client (10.0.4)Installing cron (5.1.0)Installing logrotate (2.2.0)Installing windows (4.2.2) |

Berkshelf downloads the chef-client cookbook and its dependent cookbooks to the ~/.berkshelf/cookbooks directory.

|  |
| --- |
| ls ~/.berkshelf/ |

Next, you need to upload the chef-client cookbook and its dependencies to your Chef server.

Previously, you ran knife cookbook upload to upload your learn\_chef\_httpd cookbook to the Chef server. Remember that the chef-client cookbook has dependencies on other cookbooks, so you need a way to upload everything.

You could run knife cookbook upload to manually upload each cookbook. An easier way is to run berks upload. Like berks install, berks upload handles dependencies for you.

Run berks upload to upload the chef-client cookbook and its dependencies to Chef server.

|  |
| --- |
| berks upload  Uploaded chef-client (10.0.4) to: 'https://api.chef.io:443/organizations/learn-chef-2'Uploaded cron (5.1.0) to: 'https://api.chef.io:443/organizations/learn-chef-2'Uploaded logrotate (2.2.0) to: 'https://api.chef.io:443/organizations/learn-chef-2'Uploaded windows (4.2.2) to: 'https://api.chef.io:443/organizations/learn-chef-2' |

**2. Create a role**

Now that the chef-client cookbook is on your Chef server, you need to update your node's run-list to use it. You also need to specify how often to run chef-client. In this part, you'll use a *role* to define both.

How often chef-client is run is controlled by two node attributes ([source code](https://github.com/chef-cookbooks/chef-client/blob/master/attributes/default.rb)):

* node['chef\_client']['interval'] – *interval* specifies the number of seconds between chef-client runs. The default value is 1,800 (30 minutes).
* node['chef\_client']['splay'] – *splay* specifies a maximum random number of seconds that is added to the interval. Splay helps balance the load on the Chef server by ensuring that many chef-client runs are not occurring at the same interval. The default value is 300 (5 minutes).

By default, chef-client will run every 30—35 minutes on your node. In practice, the values you choose depend on your requirements. For learning purposes, you'll specify an interval of 5 minutes (300 seconds) and a splay of 1 minute (60 seconds), causing your node to check in every 5—6 minutes.

To update your node's run-list, you could use the [knife node run\_list set](https://docs.chef.io/knife_node.html#run-list-set) command. However, that does not set the appropriate node attributes.

To accomplish both tasks, you'll use a [role](https://docs.chef.io/roles.html). Roles enable you to focus on the *function* your node performs collectively rather than each of its individual components (its run-list, node attributes, and so on). For example, you might have a *web server* role, a *database* role, or a *load balancer* role. Here, you'll create a role named *web* to define your node's function as a web server.

Roles are stored as objects on the Chef server. To create a role, you can use the [knife role create](https://docs.chef.io/knife_role.html#create) command. Another common way is to create a file (in JSON format) that describes your role and then run the [knife role from file](https://docs.chef.io/knife_role.html#from-file) command to upload that file to the Chef server. The advantage of creating a file is that you can store that file in a version control system such as Git.

First, ensure you have a directory named ~/learn-chef/roles.

|  |
| --- |
| mkdir ~/learn-chef/roles |

Now add the following to a file named ~/learn-chef/roles/web.json.

|  |
| --- |
| {  "name": "web",  "description": "Web server role.",  "json\_class": "Chef::Role",  "default\_attributes": {  "chef\_client": {  "interval": 300,  "splay": 60  }  },  "override\_attributes": {  },  "chef\_type": "role",  "run\_list": ["recipe[chef-client::default]",  "recipe[chef-client::delete\_validation]",  "recipe[learn\_chef\_httpd::default]"  ],  "env\_run\_lists": {  }  } |

This file defines the *web* role. It sets the required interval and splay attributes and sets the run-list to contain the chef-client cookbook as well as the learn\_chef\_httpd cookbook.

As a recommended practice, the run-list also contains the chef-client::delete\_validation recipe ([source code](https://github.com/chef-cookbooks/chef-client/blob/master/recipes/delete_validation.rb)). This recipe deletes the [validation certificate](https://docs.chef.io/chef_private_keys.html#chef-validator) (for example, /etc/chef/validation.pem) from your node. This certificate is used during the bootstrap process to authorize the node to connect to the Chef server, and is no longer needed.

Next, run the following knife role from file command to upload your role to the Chef server.

|  |
| --- |
| knife role from file roles/web.json  Updated Role web |

As a verification step, you can run knife role list to view the roles on your Chef server.

|  |
| --- |
| knife role list  web |

You can also run knife role show web to view the role's details.

|  |
| --- |
| knife role show webchef\_type: roledefault\_attributes: chef\_client: interval: 300 splay: 60description: Web server role.env\_run\_lists:json\_class: Chef::Rolename: weboverride\_attributes:run\_list: recipe[chef-client::default] recipe[chef-client::delete\_validation] recipe[learn\_chef\_httpd::default] |

The final step is to set your node's run-list. Run the following knife node run\_list set command to do that.

|  |
| --- |
| knife node run\_list set node1-amazonlinux "role[web]"  node1-amazonlinux:  run\_list: role[web] |

As a verification step, you can run the knife node show command to view your node's run-list.

|  |
| --- |
| knife node show node1-amazonlinux --run-list  node1-amazonlinux:  run\_list: role[web] |

You're now ready to run chef-client on your node.

**3. Run chef-client**

As before, run knife ssh to trigger chef-client to run on your node. This time, replace the search query 'name:node1-centos' with 'role:web'. If you had multiple nodes with the *web* role, chef-client would run on each of them.

This example shows key-based authentication. [Choose the method you used earlier](https://learn.chef.io/modules/manage-a-node-chef-server/rhel/hosted/update-your-nodes-configuration#step4).

|  |
| --- |
| knife ssh 'role:web' 'sudo chef-client' --ssh-user ec2-user --ssh-identity-file ~/.ssh/myec2.pem --attribute ipaddress  172.31.47.66 Recipe: chef-client::systemd\_service  172.31.47.66 \* service[chef-client] action restart  172.31.47.66 - restart service service[chef-client]  172.31.47.66  172.31.47.66 Running handlers:  172.31.47.66 Running handlers complete  172.31.47.66 Chef Client finished, 10/20 resources updated in 06 seconds |

You can see from the output that the chef-client cookbook set up chef-client as a service on your node.

You can run the [knife status](https://docs.chef.io/knife_status.html) command to display a brief summary of the nodes on your Chef server, including the time of the most recent successful chef-client run.

|  |
| --- |
| knife status 'role:web' --run-list1 minute ago, node1-centos, ["role[web]"], centos 7.4.1708. |

Every 5–6 minutes you'll see that your node performed a recent check-in with the Chef server and ran chef-client.

**4. Next steps**

Now that chef-client is set up to run every 5—6 minutes, now's a great time to experiment with your node. Here are some ideas to start with.

* Repeat the steps where you make a change to the learn\_chef\_httpd cookbook, bump its version in the metadata, and upload your changes to Chef server. Even a small change to the home page template, index.html.erb, is enough to practice the process. Watch your change appear in your web browser the next time chef-client runs.
* Manually log in to your node and stop the Apache service or delete the home page, /var/www/html/index.html. Refresh your browser window or run curl to see that your web server is down. What do you expect to happen the next time chef-client runs?