Chef

Raúl Estrada

Octubre 2020

Server provision software

As mentioned earlier, there are few options for software provisioning. Through this chapter, you will learn how to use Chef and Ansible, focusing on the latter as it is widely used across many companies and is easier to master than Chef.

There are also other options in the market that are valid and good solutions, but we are going to take a special interest in Ansible, which, to me, seems the easiest to learn and extend out of all of them.

Chef

Chef is a very interesting software that follows the bastion host principle to run configurations on our servers. A bastion host is a server placed in our private network that is able to reach our servers directly or via proxy in order to execute the actions needed to set them up with the desired state. This is an option not to be overlooked, as one of the biggest challenges that server provisioning presents is the management of secrets and authorization that, for example, Ansible needs to improve via third-party software such as Ansible Tower from Red Hat.

Chef uses recipes to configure parts of the server. A recipe is basically a set of declarative instructions that define what needs to happen in order to get the server to the desired status. For example, take a look at this:

Copy

```
execute "update-upgrade" do
   command "apt-get update && apt-get upgrade -y"
   action :run
end

package "apache2" do
   action :install
end
```

The preceding code will upgrade our system and then install the Apache2 web server.

This recipe, once finished, gets uploaded into the Chef server from a workstation, and here is the key: in Chef, there are three actors:

- Server
- Workstation
- Nodes

The server is where the recipes and configuration live. It needs to be installed prior to doing any work, and the instructions can be found at https://docs.chef.io/install_server.html.

There are three modalities of the Chef server:

- **Enterprise:** This can be installed inside your infrastructure and it is licensed, so you need to pay depending on the numbers of nodes that it is managing.
- **Open source:** This can also be installed in your infrastructure but **it does not have any support**. It is free and has to be configured and maintained by your company. It is also a cut-down version of the Enterprise Chef.
- **Hosted:** The Chef server is hosted on third-party hardware and you don't need to worry about maintaining and upgrading it. It might not be an option depending on the setup of your company.

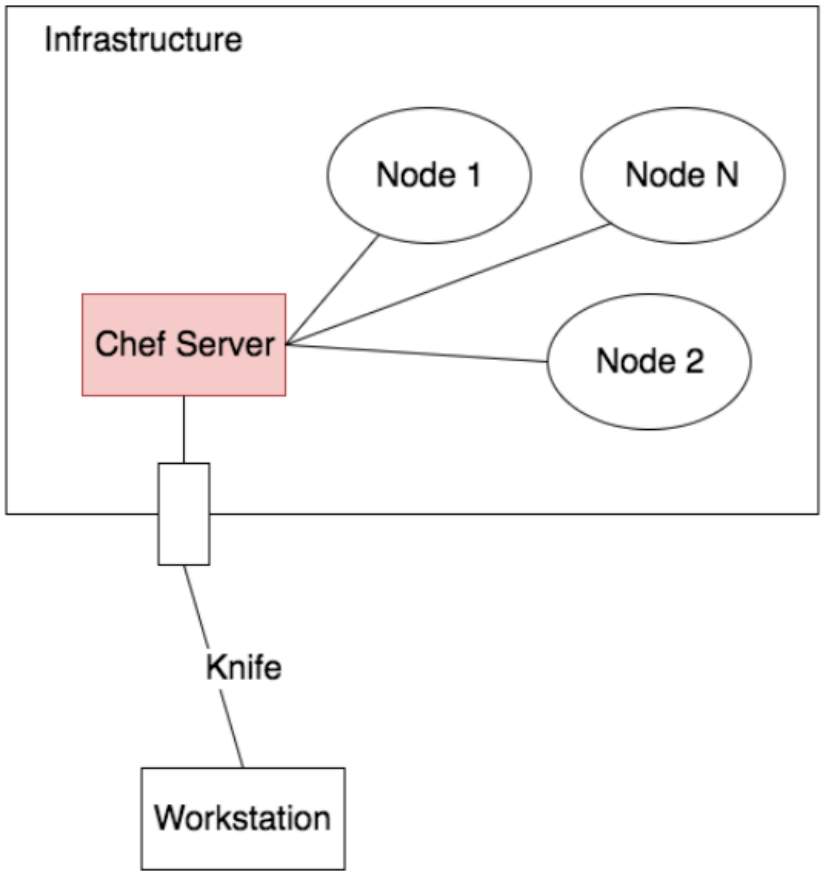
The nodes are the target hosts. Every node is registered in the Chef server and has a run list: a list of recipes that are going to be run on a host when the `chef-client` command is executed.

The workstation is the computer used to configure and upload the Chef server. This computer uses a software called knife that can do everything on the Chef server:

- Configuring roles
- Looking for VMs depending on the roles and other parameters
- Configuring run lists
- Managing secrets

Knife uses cryptographic keys to communicate with the Chef server so all the communication happens in a trusted way.

Now, if we want to picture everything, it looks like that is shown in the following diagram:



As you can see, even though the setup is quite complex (you need to set up a couple of software components) there are obvious benefits: our Chef server is behind the firewall in the demilitarized zone of our infrastructure, but it is managed via a CLI tool so all our secrets and configuration are safe inside our infrastructure.

Chef has a steep learning curve that, once we have gone through the initial learning phase, gets very familiar and easy to add new features and extend the DSL with the power of Ruby and a very well-thought-out interface.