Weighting your configuration manager options: Chef and Salt

Philip Wester

March 2020

1 Introduction

DevOps is the concatenation of the two words "Development" and "Operations", hinting that DevOps is the workflow where Development and Operations blend. The main three phases of DevOps are: Build, Deployment and Operations [1]. We are going to focus on the latter two, Deployment and Operations, how they relate to configuration management. Lastly, we will introduce and compare two of the prominent tools, Chef Infra and Salt.

1.1 Deployment

Deployment is, as the word suggests, the operation of deploying software to machines. The main problem with deployment, that pushes for the use of automated DevOps solutions, is the entanglement of different software and software versions a machine can host. This is not much of a problem if your company only needs to deploy to one machine but if the company grows, this number is likely to grow with it. Instead, let us say you have 100 machines and you now have to install some software X on all of the machines belonging to some machine set $Group1$. Not only will this become a tedious task, repeating the same operations on each individual machine, but it is also very likely that some set $k \in Group1$ will have some sort of compatibility issue with the new software. [1, 2, 3]

1.2 Operations

Operations, or infrastructure operations, can be considered the way you maintain your infrastructure's stability and performance and consists mainly out

of the two categories "Logging" and "Monitoring". While logging might not seem very challenging, it can become critical and difficult to manage in a large scale application. In these cases, using a logging tool and different logging levels (based on importance) might be a better solution . While logging alerts you when something has happened, monitoring might help you advert the crisis before it happens. Monitoring are, as the name suggests, the solutions to monitoring the system health. This can, for example, be how much CPU/RAM or network traffic is being used, allowing you to allocate more before it is too late. [1]

1.3 Configuration management tools

CM tools are generally tools that allow easier configuration of one or two of the phases stated above. It can create a central hub for configuring and handling a vast amount of hosts without having to configure each individual host for logging and installations of certain software. While there are many DevOps tools available, both open source and non-open source, this paper is going to focus on two of them. Namely Chef Infra and Salt. Both Chef Infra and Salt are considered "Configuration management tools", aiding the user in tedious configuration clusters and simplifying the tasks of running programs and updating the systems.

2 Chef

Chef was first released in 2009 and achieved it's first stable releases in 2018 and 2019. Today Chef has a number of products such as Infra, Habitat, Inspec and Automate. They are all answers to different problems, such as Chef Inspec being an answer to the common security threat and Chef Automate handling building issues, but they can be nicely integrated in a chain with one another. We are only going to focus on Chef Infra in this paper and from now on when a Chef component is mentioned, it is the Chef Infra counter part it is referring to. (eg. Chef Server is referring to Chef Infra Server)

The Chef CM topology consists of three types of elements, see Figure 1, [4, 3]:

- Workstation(s)
- Chef Server
- Node(s)

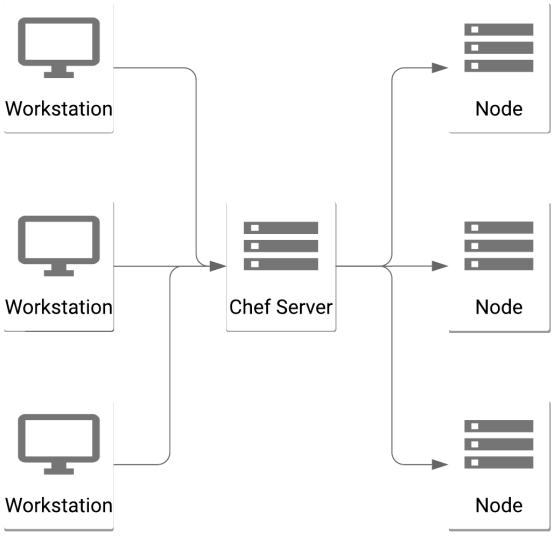

Figure 1: The topology of Chef

2.1 Nodes

When you set up Chef, your main goal is to configure a number of machines. These machines are called *Nodes* and does not need to be physical servers, they can even be cloud instances. Each node in a Chef system needs to have a Chef client and be registered at the Chef Server. [4, 3]

2.2 Chef Server

The Chef Server is the brain and heart of the Chef system. The Chef Server stores configuration files and other meta-data while also working as a controller for the user to access and alter nodes. One of the most important components of the Chef Server is the Bookshelf, which stores and manages Chef code uploaded from the *workstations*. The location on the Chef Server that stores all data objects is refered to as the Chef Server s *Chef repository* and needs to be managed with a control system, like GitHub. To update this repository, a workstation running *Knife* has to push up their local (altered) repository. [4, 3]

2.3 Workstations

A workstation is simply a machine that is used for communication with the Chef Server and updates its *Chef repository*. It is from the workstation(s) that a user can deploy Chef code to the Chef Server and ultimately change the state of nodes and the Chef system. To do this, the workstation will have to be running the Chef tool *Knife*. [4, 3]

2.4 Chef Tools

Each of these elements are built up using a number of components, or Chef Tools. The most important ones being: Bookshelf, Cookbooks, Recipes, Knife, PostgreSQL DB and Chef client. These components cover many different areas such as data storage, data transfers and acting as web interfaces. While some are indirectly used trough the Chef system, others will be used directly by the user. Meaning that you will have to familiarize yourself with them. [4, 3]

2.4.1 Recipes and Cookbooks

A recipe is the smallest component in the Chef management system and is created on a workstation. They consist of Ruby code and pattern matching and described in Chefs documentation as "mostly a collection of resources" [5]. While a recipe can be deployed directly to the Chef Server , you often want to create a Cookbook instead. A Cookbook is more or less a collection of recipes along with other META-data. As re-inventing the wheel is should always be frowned upon in developer communities, Chef offers a platform, Chef Supermarket, where you can share and download recipes and cookbooks from the community. Together with Bookshelf and their cookbooks, recipes are the backbone of the Chef system. [4, 3]

2.4.2 Bookshelf

Cookbooks deployed to the Chef Server is being stored in the Bookshelf. But before a cookbook is stored in the Bookshelf, similar to many Linux package managers, it checks the integrity and dependencies of the cookbook [3]. By doing so, the Bookshelf avoids storing corrupt recipes or the same recipe more than once. [4, 3]

2.4.3 Knife

The main tool on the workstation is *Knife*. Knife is used as a command line-interface to communicate with and update the Chef Server from the local (workstation) Chef repository. With a high amount of Chef commands and Knife sub commands a user can manage the Chef systems different elements and components. [3, 6]

Because Knife is used to access the remote Chef Server , the security aspect is very important. Since Chef 12, Knife uses SSL encryption by default and each workstation has to download their private Key and Certificate from the Chef Server . By default the Chef Server itself creates a self-signed certificate. [7]

2.5 Hosted Chef

Instead of setting up your own Chef Server , you can use Hosted Chef, Chef Incorporation's cloud service (SaaS). Hosted Chef is what it sounds like, Chef Inc. will host an instance of a Chef Server and handles all Chef management for you. "Hosted Chef supports many cloud platforms such as AWS, Windows Azure etc. [3]

2.6 Private Chef

Private Chef is easily confused with Hosted Chef since they function very similar. The main difference is that Private Chef is controlled by individual organizations and does therefore have the same limitations as Hosted Chef. [3]

2.7 Open source Chef

Unlike Hosted Chef, Open source Chef is hosted locally and is, as the name suggests, open source. This means that while you have access to the entire open source community, you will instead have to set up and manage the Chef system yourself. [3]

3 Salt

Salt was first released in 2011 by SaltStack and achieved it's first stable release in 2020. Salt is built around the philosophy that, since a configuration manager exists to simplify the configuration, configuration managers should have as little

and simple configuration, as possible, themselves [8]. SaltStack offers multiple services and software but in this paper we will focus solely on the open source configuration manager Salt.

3.1 Salt topology

The Salt topology has two types of components, the master and the minions, see 2. The Salt master is the component used to remotely configure and control the Salt minions. While it is possible to run Salt and control a minion without connecting or communicating trough a master [9], doing it as a common practice would defeat the purpose of running Salt in the first place.

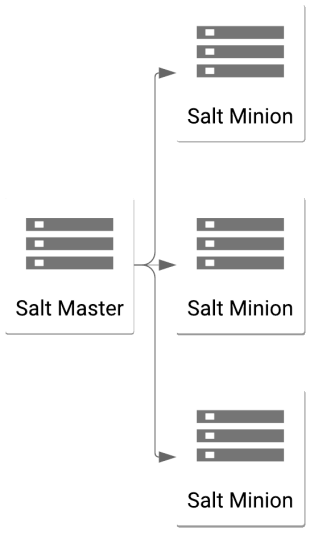

Figure 2: The topology of Salt

3.2 Modules

Salt is built up by a large number of python scripts called modules. These modules handle everything from communication between master and minions to monitoring of disk space. As Salt aspire to need as little configuration as possible itself, it is shipped with a large number of pre-installed modules. Salt will then detect and load necessary modules to memory, without the user needing to configure which modules that suit their needs. For example, Salt was originally intended to be run on Unix systems but as the demand for Salt on

Windows increased, Salt was adapted to work with the windows environment. This was done by the creation of the *Virtual modules*. These modules simulate an environment suitable for the Salt application but they are only ran when the OS is detected by Salt as Windows system. [8, 10]

3.2.1 SLS and customisation

Besides allowing for remote execution, Salt allows for configuration to set up customized reactions to events. For example, it is possible to set up a reaction to install certain packages or send specific data to the database when a new minion is created. Even though Salt is primarily written in Python, the modules can be used either directly from command line with Salt commands or ran automatically from SLS scripts. SLS allows easy creation modification of simple configuration files, meaning that even though the user does not have any Python experience, he/she will still be able to configure Salt after their needs (Although he/she will probably have to learn how to write these SLS files). [9, 10]

3.3 Salt Cloud

SaltStack offers the tool Salt Cloud to allow Salt to handle and configures cloud resources. This solution allows for configuration of cloud servers and services, but does not migrate entire Salt to the cloud. [11]

4 Stating the differences

While both Chef and Salt are configuration management tools, they have a number of differences that are worth noting.

4.1 Topology

An first and obvious difference between Salt and Chef is the difference in their topology. While Chef has three layers (workstation, Chef Server and nodes), Salt only use two (master and minions). This is because Salt does not define any workstation but instead encourages running the commands on the Salt master, whether you ssh or log in directly to that machine is up to you.

This can either be called simple and easy, because you do not need to install and run specific workstation software such as Knife. But it can also be considered

flawed since it leaves to the user to solve the question of how to access the Salt Master securely.

4.2 Syntax

Secondly, configuration in Chef requires some experience in Ruby while Salt use SLS. Since the Salt resources are simple to access and modify with SLS examples from Salt's website, I would argue that learning to write SLS for configuration files is easier than learning to write Ruby recipes, which might make Salt preferable if you haven't got experience with any of them. If you on the other hand already have Ruby experience, this will be an advantage when running Chef as this allows you to use all the power that comes with the famous Ruby language and environment.

4.3 Cloud support

Chef is definitely superior in the aspects of the cloud. It does not only offer good integration and migration to many of the important cloud service vendors such as AWS, IBM, Windows Azure etc. they also have more support and a long history of making these integrations work. While Salt offers "Salt Cloud", it does not allow for the same scale and transparency when migrating to the cloud as Chef. Instead of allowing a full cloud migration as with Chef hosted, Salt will will decouple the cloud servers to your internal Salt system/Salt master.

4.4 Complexity

Both Salt and Chef are easy to install in your system. Although because Salt only offers two layers instead of three in the topology and that the software can be installed very easily with Linux repositories, Salt definitely takes the lead in this field. It is not surprising that the community that lives by the philosophy "to create as little configuration as possible" is the one that gives lightweight and easy-to-use software. Beyond the actual installation of Salt, it is also easy to keep track and understand the different parts of Salt. This is because Salt is built mostly around the modules, and understanding the module system will then give you most of the knowledge you need. That Salt minions can be controlled from the master with ssh, without having Salt-minion installed, is an extra plus.

4.5 Documentation

In my personal opinion, Chef's documentation can be a bit confusing and Salt's documentation might be outdated and sometimes even misleading (Ex. [12]), both of them should focus on making the documentation a better tool for their users. Chef on one hand, has a large and old community which means that there are tons of third party documentation and people to ask. While Salt on the other hand focus on simplicity, meaning that if the documentation is lacking and you put your effort into it, you might be able to figure out the solution yourself.

4.6 Customization

Both Salt and Chef have support for writing your own simple scripts to configure mandatory configuration such as installation and logging in nodes/minions. Salt modules on the other hand allows easy customization of the Salt minions and hosts along with the events and the reactor system. These systems make it possible for Salt to be configured to either notify or run arbitrary scripts whenever an internal event occurs, on either a Salt minion or master. While Chef might not have an event system such as Salt, its Ruby based recipes allow great customization, as long as you are able to write your own Ruby scripts or find suiting plugins on Chef Supermarket.

5 Which one should you use?

By now you probably already have an idea of which one of the two configuration management tools suits you the best. If you want a simple setup and have control over the system fast, or if you want to run a lightweight configuration manager, you should use Salt. If you on the other hand are familiar with Ruby, want to be able to get a lot of support from a large community with many plugins and you want to be able to migrate your system with different cloud services, Chef would be the better option.

References

[1] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE Software*, vol. 33, no. 3, pp. 94–100, 2016.

[2] J.-L. Boulanger, "7 - configuration management," in *Certifiable Software Applications 2*, pp. 87–96, Elsevier Ltd, 2017.

[3] R. Sharma and M. Soni, "Learning chef." https://learning.oreilly.com/library/view/learning-chef/9781783285211/, 2015.

[4] I. Chef Software, "An overview of chef infra." https://docs.chef.io/chef_overview/.

[5] I. Chef Software, "About recipes." https://docs.chef.io/recipes/.

[6] I. Chef Software, "About knife." https://docs.chef.io/workstation/knife/.

[7] I. Chef Software, "Security." https://docs.chef.io/runbook/server_security/.

[8] J. Hall, "Extending saltstack." https://learning.oreilly.com/library/view/extending-saltstack/9781785888618/, 2016.

[9] C. Myers, "Learning saltstack - second edition." https://learning.oreilly.com/library/view/learning-saltstack-/9781785881909/, 2016.

[10] SaltStack, "Salt table of contents." https://docs.saltstack.com/en/latest/contents.html.

[11] SaltStack, "Salt cloud." https://docs.saltstack.com/en/latest/topics/cloud/.

[12] P. Wester, "Incorrect naming in documentation." https://github.com/saltstack/salt/issues/56428.