Ansible

Raúl Estrada

Octubre 2020

easiest to learn and extend. It is also easy to understand and offers a fairly comprehensive open source version that works with all the features from Ansible. You can also buy a license of Ansible Tower (or similar) to run Ansible Playbooks in a bastion host configuration as Chef or Puppet.

Ansible is basically a **domain-specific language** (**DSL**) for executing operations on remote hosts that are defined in an inventory.

Ansible works by running playbooks in the desired servers via SSH, so unlike Chef or Puppet, we don't need to install anything in the remote hosts; we should just be able to SSH into them. A playbook is basically a **Yet Another Markup Language** (**YAML**) with a set of instructions to get the server into the desired state in the same way as if we were executing a Bash script. A Playbook looks like this:

Copy

```
---
-hosts:webservers
vars:
http_port:80
max_clients:200
remote_user:root
tasks:
-name:ensure apache is at the latest version
yum:name=httpd state=latest
-name:write the apache config file
template:src=/srv/httpd.j2 dest=/etc/httpd.conf
notify:
-restart apache
-name:ensure apache is running (and enable it at boot)
service:name=httpd state=started enabled=yes
handlers:
-name:restart apache
service:name=httpd state=restarted
```

Reading through the file will make you understand how easy and straightforward it is to understand what the Playbook doing.

As you can see, in the second line, we are specifying that we want to run this Playbook in the hosts called `webservers` . This can be defined in the other part of Ansible: the inventory. The Ansible inventory is basically a file with the list of hosts in your infrastructure, as follows:

Copy

```
[webservers]
host1
host2

[dbservers]
192.168.0.[1:3]
```

This file is very straightforward but can get really complicated as well:

- The names between brackets are groups
- The groups contain hosts that can be defined with generators or they can just be listed
- Groups can have configuration specific to them or even override variables

In the preceding example, we have two groups: `webservers` and `dbservers` .

Web servers are only two hosts:

```
Host1
Host2
```

Dbservers use a generator and we have three hosts:

```
192.168.0.1
192.168.0.2
192.168.0.3
```

As mentioned earlier, we can also define variables in the inventory. These variables can be scoped on the group and the host. Let's take a look at the following inventory:

Copy

```
[dbservers]
192.168.0.[1:3]

[webservers]
host1 role=master
host2

[dbservers:vars]
timezone=utc
```

As you can see, we have two variables:

- `timezone` : This is applied to all the hosts of the group `dbservers` .
- `role` : This is applied to the host `host1` of the group `webservers` .

This variable can be used in `playbooks` in order to have a specific configuration for specific hosts, as we will see later on in this chapter.

Groups can also be combined into bigger groups:

Copy

```
[dbservers]
192.168.0.[1:3]

[webservers]
host1
host2

[mongoservers]
10.0.0.1
10.0.0.2

[dataservers:child]
mongoservers
dbservers
```

In the preceding inventory, we can find the following:

- `dbservers`
- `mongoservers`
- `webservers`
- `dataservers`
- `all`
- `ungrouped`

Even though we did not specify it, Ansible always has two default groups called `all` and `ungrouped` that are self-descriptive: `all` is all the hosts in the inventory and `ungrouped` is all the hosts that are not specified in any group.

As stated earlier, Ansible does not follow the bastion host architecture as Chef or Puppet, but it follows the client/server architecture: our host needs to be able to reach the destination hosts (the ones on the inventory) in order to work.

This can be inconvenient depending on your infrastructure architecture, but it can be worked around using Ansible Tower or Rundeck to execute Ansible playbooks from inside your demilitarized zone.

In this chapter, we are going to use Ansible to build real production-ready examples in combination with Terraform so that we get a grasp of the real usage of the tools.