Introduction to Ansible

Yubo Cao & Anish Goyal

November 5, 2022

Contents

1 Introduction

Ansible (Figure 1) is currently the simplest and easy-to-use excellent software among operation and maintenance automation tools, which can be used to manage various resources. Users can use Ansible to automatically deploy applications for full deployment of their IT infrastructure.

Ansible can:

- Initialize and configure servers
- **Run CyberPatriot tasks** & **Automatically Security Hardening**
- Performing software updates

Ansible, written in Python, is inferior in sense of performance to other tools, but it is very easy to use and has very good community support. Ansible is a very good tool for beginners to learn automation tools.

Figure 1: Ansible Icon

2 Installation

Ansible can be installed on any Linux distribution (in fact, you can install it on Windows and Mac OS as well).

We only care about installation on Ubuntu and Windows 10.

2.1 Ubuntu

```
sudo apt update
sudo apt install ansible -qq
```

2.2 Windows 10

```
choco install ansible
```

Or, if you don't have Chocolatey installed, you can install it by running:

```
pip install ansible
```

If you don't even have Python or Pip installed, you can install them from Python's official website.

3 Terms

Control Node The machine that runs Ansible. Those machines command the managed nodes, e.g., your laptop.

Managed Node The machine that is managed by Ansible. Those machines are controlled by the control node, e.g., server. However, the control node can also be a managed node, i.e., self-managed.

Inventory A list of managed nodes.

Playbook A list of tasks to be executed on the managed nodes.

Task A single task to be executed on the managed nodes.

Roles A collection of tasks and modules to be executed on the managed nodes.

Module A single module to be executed on the managed nodes. They are like commands in the shell.

4 Usage

4.1 Configuration

The default config file is `/etc/ansible/ansible.cfg`. However, the configuration files take precedence in the following order:

1. `ANSIBLE_CONFIG` (an environment variable)
2. `ansible.cfg` (in the current directory)
3. `~/ansible.cfg` (in the home directory)
4. `/etc/ansible/ansible.cfg` (the default location)

However, we will still use the default location for simplicity.

4.2 Inventory

Ansible uses an inventory file to specify the managed nodes. The default inventory file is `/etc/ansible/hosts`.

Considering the following inventory file:

```
[dev]
192.168.0.1

[school]
192.168.0.2
192.168.0.3

[test]
192.168.0.4

[local]
localhost

[local:vars]
ansible_connection=local
```

In that situation, the control nodes are grouped into 4 groups: `dev`, `school`, `test`, and `local`. The `local` group is a special group that contains the control node itself. This way, we can run commands on the control node itself.

Use `ansible-inventory` to view the inventory file. For example, the above inventory file can be viewed as:

```
ansible-inventory --graph
```

```
root@yubo-virtual-machine:/etc/ansible# ansible-inventory --graph
@all:
  |--@dev:
  |  |--192.168.0.1
  |--@local:
  |  |--localhost
  |--@school:
  |  |--192.168.0.2
  |  |--192.168.0.3
  |--@test:
  |  |--192.168.0.4
  |--@ungrouped:
```

4.2.1 SSH

Since for all remote connections Ansible uses SSH, you need to make sure that you can SSH into the managed nodes without a password. In order to do so, you need to generate a SSH key pair and add the public key to the `authorized_keys` file on the managed nodes.

```
ssh-keygen -t rsa
ssh-copy-id user@host
```

Another way is to set all the host to use the same username and password, and then use the `ansible_user` and `ansible_password` variables in the inventory file.

```
…
[all:vars]
ansible_user=user
ansible_password=password
```

4.3 Command

Ansible can be used to run commands on the managed nodes (and more importantly, modules). Use `-m` to specify the module to be used.

```
ansible all -m ping
```

The above command will run the `ping` module on all the managed nodes. Some output like the following will be shown:

```
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
…
```

Most of module takes the same arguments. For example, the `package` module takes the `name` argument to specify the package to be installed.

```
ansible all -m package -a "name=git state=present"
```

The above command install the `git` package on all the managed nodes (if not present; *indempotence*). The following output will be shown:

```
yubo@yubo-virtual-machine:~$ ansible local -m package -a "name=git state=presen
t"
localhost | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "cache_update_time": 1667624256,
    "cache_updated": false,
    "changed": false
}
```

4.4 Playbook

A playbook is a list of tasks to be executed on the managed nodes. It is written in YAML. The following is an example of a playbook:

```
---
- name: Install Softwares
  hosts: all
  become: yes
  tasks:
    - name: Install git
      package:
        name: git
        state: present
```

The above playbook will install the `git` package on all the managed nodes. Notice that the `become` option is set to `yes`. This means that the tasks will be executed with `sudo` (or `su` if the user is `root`).

In a playbook, there are 4 main sections:

hosts The hosts to run the playbook on.

become Whether to run the tasks with `sudo`.

vars Variables to be used in the playbook.

tasks The tasks to be executed.

handlers The handlers to be executed, after the tasks.

To execute a playbook, use the `ansible-playbook` command:

```
ansible-playbook playbook.yml
```

Since none of our virtual hosts is set up, the above command will fail.

5 Ansible Roles

Ansible roles are a way to structure your Ansible projects. A role is a directory that contains a set of tasks, handlers, templates, files, variables, and other Ansible artifacts. Roles can be shared and reused. Roles can also be created with `ansible-galaxy`.

5.1 List of Roles

```
ansible-galaxy list
```

The above command will list all the roles installed on the control node.

5.2 Download Roles

Go to `https://galaxy.ansible.com/` to search for roles. For example, to install the `dev-sec.os-hardening` role, use the following command:

```
ansible-galaxy install dev-sec.os-hardening
```

5.3 Run Roles

Roles can be run in a playbook. For example, the following playbook will run the `dev-sec.os-hardening` role:

```yaml
---
- name: Harden the system
  hosts: all
  become: yes
  roles:
    - dev-sec.os-hardening
```