DevOps Certification Training

Study Material - Ansible

Ansible automates the management of remote systems and controls their desired state. A basic Ansible environment has three main components:

Control node

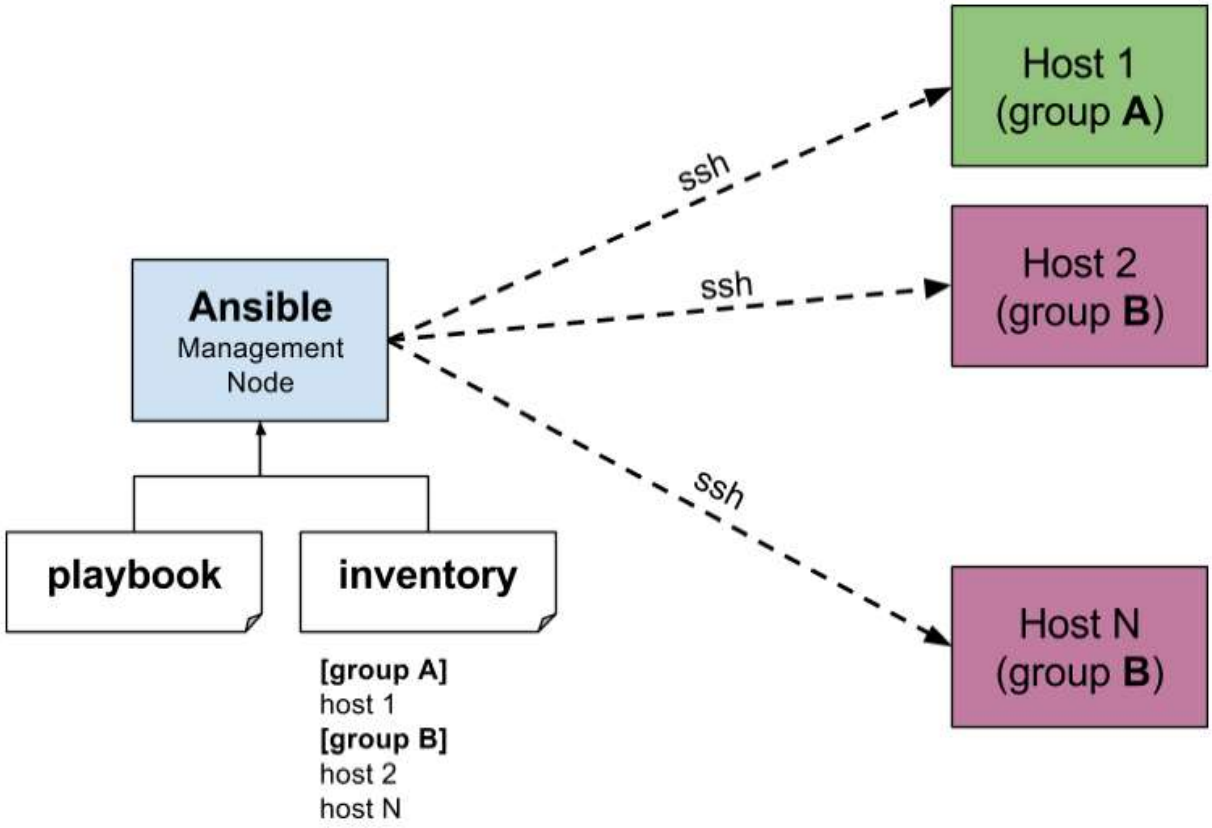
A system on which Ansible is installed. You run Ansible commands such as `ansible` or `ansible-inventory` on a control node.

Managed node

A remote system, or host, that Ansible controls.

Inventory

A list of managed nodes that are logically organized. You create an inventory on the control node to describe host deployments to Ansible.



Ready to start using Ansible? Complete the following steps to get up and running:

Install Ansible. Visit the installation guide for complete details.

```
python3 -m pip install --user ansible
```

Create an inventory by adding the IP address or fully qualified domain name (FQDN) of one or more remote systems to `/etc/ansible/hosts`. The following example adds the IP addresses of three virtual machines in KVM:

```
[myvirtualmachines]
192.0.2.50
192.0.2.51
192.0.2.52
```

3. Verify the hosts in your inventory.

```
ansible all --list-hosts
hosts (1):
192.0.2.50
192.0.2.51
192.0.2.52
```

4. Set up SSH connections so Ansible can connect to the managed nodes.

Add your public SSH key to the `authorized_keys` file on each remote system.

5. Test the SSH connections, for example:

```
ssh username@192.0.2.50
```

If the username on the control node is different on the host, you need to pass the `-u` option with the `ansible` command.

6. Ping the managed nodes.

```
ansible all -m ping
```

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

```
    "changed": false,
    "ping": "pong"
}
```

Congratulations! You are now using Ansible. Continue by learning how to build an inventory.

Building an inventory

Inventories organize managed nodes in centralized files that provide Ansible with system information and network locations. Using an inventory file, Ansible can manage a large number of hosts with a single command. Inventories also help you use Ansible more efficiently by reducing the number of command-line options you need to specify. For example, inventories usually contain the SSH user so you do not need to include the `-u` flag when running Ansible commands. Inventory files can be in `INI` or `YAML` format. For demonstration purposes, this section uses `YAML` format only.

Complete the following steps:

1. Open a terminal window on your control node.
2. Create a new inventory file named `inventory.yaml` in any directory and open it for editing.
3. Add a new group for your hosts then specify the IP address or fully qualified domain name (FQDN) of each managed node with the `ansible_host` field. The following example adds the IP addresses of three virtual machines in KVM:

```
virtualmachines:
  hosts:
    vm01:
      ansible_host: 192.0.2.50
    vm02:
      ansible_host: 192.0.2.51
    vm03:
      ansible_host: 192.0.2.52
```

4. Verify your inventory. If you created your inventory in a directory other than your home directory, specify the full path with the `-i` option.

```
ansible-inventory -i inventory.yaml --list
```

5. Ping the managed nodes in your inventory. In this example, the group name is `virtualmachines` which you can specify with the `ansible` command instead of `all`.

```
ansible virtualmachines -m ping -i inventory.yaml
vm03 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vm02 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
vm01 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Congratulations! You have successfully built an inventory.

Creating a playbook

Playbooks are automation blueprints, in `YAML` format, that Ansible uses to deploy and configure managed nodes.

Playbook

A list of plays that define the order in which Ansible performs operations, from top to bottom, to achieve an overall goal.

Play

An ordered list of tasks that maps to managed nodes in an inventory.

Task

A list of one or more modules that defines the operations that Ansible performs.

Module

A unit of code or binary that Ansible runs on managed nodes. Ansible modules are grouped in collections with a Fully Qualified Collection Name (FQCN) for each module.

In the previous section, you used the `ansible` command to ping hosts in your inventory. Now let's create a playbook that pings your hosts and also prints a "Hello world" message.

Complete the following steps:

1. Open a terminal window on your control node.
2. Create a new playbook file named `playbook.yaml` in any directory and open it for editing.
3. Add the following content to `playbook.yaml`:

```yaml
- name: My first play
  hosts: virtualmachines
  tasks:
   - name: Ping my hosts
     ansible.builtin.ping:
   - name: Print message
     ansible.builtin.debug:
       msg: Hello world
```

4. Run your playbook.

```
ansible-playbook -i inventory.yaml playbook.yaml
```

Ansible returns the following output:

```
PLAY [My first play] ***********************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [vm01]
ok: [vm02]
ok: [vm03]

TASK [Ping my hosts] ***********************************************************************
ok: [vm01]
ok: [vm02]
ok: [vm03]

TASK [Print message] ************************************************************************
ok: [vm01] => {
    "msg": "Hello world"
}
ok: [vm02] => {
    "msg": "Hello world"
}
ok: [vm03] => {
    "msg": "Hello world"
}

PLAY RECAP *********************************************************************************
vm01: ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
vm02: ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
vm03: ok=3    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

In this output you can see:

- The names that you give the play and each task. You should always use descriptive names that make it easy to verify and troubleshoot playbooks.
- The `Gather Facts` task runs implicitly. By default Ansible gathers information about your inventory that it can use in the playbook.
- The status of each task. Each task has a status of `ok` which means it ran successfully.
- The play recap that summarizes results of all tasks in the playbook per host. In this example, there are three tasks so `ok=3` indicates that each task ran successfully.

Congratulations! You have just created your first Ansible playbook.