Build Your Inventory

Running a playbook without an inventory requires several command-line flags. Also, running a playbook against a single device is not a huge efficiency gain over making the same change manually. The next step to harnessing the full power of Ansible is to use an inventory file to organize your managed nodes into groups with information like the `ansible_network_os` and the SSH user. A fully-featured inventory file can serve as the source of truth for your network. Using an inventory file, a single playbook can maintain hundreds of network devices with a single command. This page shows you how to build an inventory file, step by step.

Basic inventory

First, group your inventory logically. Best practice is to group servers and network devices by their What (application, stack or microservice), Where (datacenter or region), and When (development stage):

- **What**: db, web, leaf, spine
- **Where**: east, west, floor_19, building_A
- **When**: dev, test, staging, prod

Avoid spaces, hyphens, and preceding numbers (use `floor_19`, not `19th_floor`) in your group names. Group names are case sensitive.

This tiny example data center illustrates a basic group structure. You can group groups using the syntax `[metagroupname:children]` and listing groups as members of the metagroup. Here, the group `network` includes all leafs and all spines; the group `datacenter` includes all network devices plus all webservers.

```
---

leafs:
  hosts:
    leaf01:
      ansible_host: 10.16.10.11
    leaf02:
      ansible_host: 10.16.10.12

spines:
  hosts:
    spine01:
      ansible_host: 10.16.10.13
    spine02:
      ansible_host: 10.16.10.14

network:
  children:
    leafs:
    spines:

webservers:
  hosts:
    webserver01:
      ansible_host: 10.16.10.15
    webserver02:
      ansible_host: 10.16.10.16

datacenter:
  children:
    network:
    webservers:
```

You can also create this same inventory in INI format.

```
[leafs]
leaf01
leaf02

[spines]
spine01
spine02

[network:children]
leafs
spines

[webservers]
webserver01
webserver02

[datacenter:children]
network
webservers
```

Add variables to the inventory

Next, you can set values for many of the variables you needed in your first Ansible command in the inventory, so you can skip them in the `ansible-playbook` command. In this example, the inventory includes each network device's IP, OS, and SSH user. If your network devices are only accessible by IP, you must add the IP to the inventory file. If you access your network devices using hostnames, the IP is not necessary.

```
---

leafs:
  hosts:
    leaf01:
      ansible_host: 10.16.10.11
      ansible_network_os: vyos.vyos.vyos
      ansible_user: my_vyos_user
    leaf02:
      ansible_host: 10.16.10.12
      ansible_network_os: vyos.vyos.vyos
      ansible_user: my_vyos_user

spines:
  hosts:
    spine01:
      ansible_host: 10.16.10.13
      ansible_network_os: vyos.vyos.vyos
      ansible_user: my_vyos_user
    spine02:
      ansible_host: 10.16.10.14
      ansible_network_os: vyos.vyos.vyos
      ansible_user: my_vyos_user

network:
  children:
    leafs:
    spines:

webservers:
  hosts:
    webserver01:
      ansible_host: 10.16.10.15
      ansible_user: my_server_user
    webserver02:
      ansible_host: 10.16.10.16
      ansible_user: my_server_user

datacenter:
  children:
    network:
    webservers:
```

Group variables within inventory

When devices in a group share the same variable values, such as OS or SSH user, you can reduce duplication and simplify maintenance by consolidating these into group variables:

```
---

leafs:
  hosts:
    leaf01:
      ansible_host: 10.16.10.11
    leaf02:
      ansible_host: 10.16.10.12
  vars:
    ansible_network_os: vyos.vyos.vyos
    ansible_user: my_vyos_user

spines:
  hosts:
    spine01:
      ansible_host: 10.16.10.13
    spine02:
      ansible_host: 10.16.10.14
  vars:
    ansible_network_os: vyos.vyos.vyos
    ansible_user: my_vyos_user

network:
  children:
    leafs:
    spines:

webservers:
  hosts:
    webserver01:
      ansible_host: 10.16.10.15
    webserver02:
      ansible_host: 10.16.10.16
  vars:
    ansible_user: my_server_user
```

```
datacenter:
  children:
    network:
    webservers:
```

Variable syntax

The syntax for variable values is different in inventory, in playbooks, and in the `group_vars` files, which are covered below. Even though playbook and `group_vars` files are both written in YAML, you use variables differently in each.

- In an ini-style inventory file you **must** use the syntax `key=value` for variable values: `ansible_network_os=vyos.vyos.vyos`.
- In any file with the `.yml` or `.yaml` extension, including playbooks and `group_vars` files, you **must** use YAML syntax: `key: value`.
- In `group_vars` files, use the full `key` name: `ansible_network_os: vyos.vyos.vyos`.
- In playbooks, use the short-form `key` name, which drops the `ansible` prefix: `network_os: vyos.vyos.vyos`.

Group inventory by platform

As your inventory grows, you may want to group devices by platform. This allows you to specify platform-specific variables easily for all devices on that platform:

```
---

leafs:
  hosts:
    leaf01:
      ansible_host: 10.16.10.11
    leaf02:
      ansible_host: 10.16.10.12

spines:
  hosts:
    spine01:
      ansible_host: 10.16.10.13
    spine02:
      ansible_host: 10.16.10.14

network:
  children:
    leafs:
    spines:
  vars:
    ansible_connection: ansible.netcommon.network_cli
    ansible_network_os: vyos.vyos.vyos
    ansible_user: my_vyos_user

webservers:
  hosts:
    webserver01:
      ansible_host: 10.16.10.15
    webserver02:
      ansible_host: 10.16.10.16
  vars:
    ansible_user: my_server_user

datacenter:
  children:
    network:
    webservers:
```

With this setup, you can run `first_playbook.yml` with only two flags:

```
ansible-playbook -i inventory.yml -k first_playbook.yml
```

With the `-k` flag, you provide the SSH password(s) at the prompt. Alternatively, you can store SSH and other secrets and passwords securely in your group_vars files with `ansible-vault`. See :ref:`network_vault` for details.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\getting_started\(ansible-devel)(docs)(docsite)(rst)(network)(getting_started)first_inventory.rst`, line 248); *backlink*
>
> Unknown interpreted text role "ref".

Verifying the inventory

You can use the :ref:`ansible-inventory` CLI command to display the inventory as Ansible sees it.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\getting_started\(ansible-devel)(docs)(docsite)(rst)(network)(getting_started)first_inventory.rst`, line 253); *backlink*
>
> Unknown interpreted text role "ref".

```
$ ansible-inventory -i test.yml --list
```

```
{
  "_meta": {
      "hostvars": {
          "leaf01": {
              "ansible_connection": "ansible.netcommon.network_cli",
              "ansible_host": "10.16.10.11",
              "ansible_network_os": "vyos.vyos.vyos",
              "ansible_user": "my_vyos_user"
          },
          "leaf02": {
              "ansible_connection": "ansible.netcommon.network_cli",
              "ansible_host": "10.16.10.12",
              "ansible_network_os": "vyos.vyos.vyos",
              "ansible_user": "my_vyos_user"
          },
          "spine01": {
              "ansible_connection": "ansible.netcommon.network_cli",
              "ansible_host": "10.16.10.13",
              "ansible_network_os": "vyos.vyos.vyos",
              "ansible_user": "my_vyos_user"
          },
          "spine02": {
              "ansible_connection": "ansible.netcommon.network_cli",
              "ansible_host": "10.16.10.14",
              "ansible_network_os": "vyos.vyos.vyos",
              "ansible_user": "my_vyos_user"
          },
          "webserver01": {
              "ansible_host": "10.16.10.15",
              "ansible_user": "my_server_user"
          },
          "webserver02": {
              "ansible_host": "10.16.10.16",
              "ansible_user": "my_server_user"
          }
      }
  },
  "all": {
      "children": [
          "datacenter",
          "ungrouped"
      ]
  },
  "datacenter": {
      "children": [
          "network",
          "webservers"
      ]
  },
  "leafs": {
      "hosts": [
          "leaf01",
          "leaf02"
      ]
  },
  "network": {
      "children": [
          "leafs",
          "spines"
      ]
  },
  "spines": {
      "hosts": [
          "spine01",
          "spine02"
      ]
  },
  "webservers": {
      "hosts": [
          "webserver01",
          "webserver02"
      ]
  }
}
```

Protecting sensitive variables with `ansible-vault`

The `ansible-vault` command provides encryption for files and/or individual variables like passwords. This tutorial will show you how to encrypt a single SSH password. You can use the commands below to encrypt other sensitive information, such as database passwords, privilege-escalation passwords and more.

First you must create a password for ansible-vault itself. It is used as the encryption key, and with this you can encrypt dozens of different passwords across your Ansible project. You can access all those secrets (encrypted values) with a single password (the ansible-vault password) when you run your playbooks. Here's a simple example.

1. Create a file and write your password for ansible-vault to it:

```
echo "my-ansible-vault-pw" > ~/my-ansible-vault-pw-file
```

2. Create the encrypted ssh password for your VyOS network devices, pulling your ansible-vault password from the file you just created:

```
ansible-vault encrypt_string --vault-id my_user@~/my-ansible-vault-pw-file 'VyOS_SSH_password' --name 'ansible_p
```

If you prefer to type your ansible-vault password rather than store it in a file, you can request a prompt:

```
ansible-vault encrypt_string --vault-id my_user@prompt 'VyOS_SSH_password' --name 'ansible_password'
```

and type in the vault password for `my_user`.

The :option:`--vault-id <ansible-playbook --vault-id>` flag allows different vault passwords for different users or different levels of access. The output includes the user name `my_user` from your `ansible-vault` command and uses the YAML syntax `key: value`:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\getting_started\(ansible-devel)(docs)(docsite)(rst)(network)(getting_started)first_inventory.rst, line 362); *backlink***
>
> Unknown interpreted text role "option".

```
ansible_password: !vault |
       $ANSIBLE_VAULT;1.2;AES256;my_user
       66386134653765386232383236303063623663343437643766386435663632343266393064373933
       36616661323633393036393535383166626166638356631650a31633831666366643938313835303 2
       63393934343937373637306162366265383461316334383132626462656463363630613832313562
       38376462666638356 40a31316434353531666665303135376361303765636261353563353838653 9
       65656439626166666363323435613131643066353762333232326232323565376635
Encryption successful
```

This is an example using an extract from a YAML inventory, as the INI format does not support inline vaults:

```
...

vyos: # this is a group in yaml inventory, but you can also do under a host
  vars:
    ansible_connection: ansible.netcommon.network_cli
    ansible_network_os: vyos.vyos.vyos
    ansible_user: my_vyos_user
    ansible_password:  !vault |
         $ANSIBLE_VAULT;1.2;AES256;my_user
         66386134653765386232383236303063623663343437643766386435663632343266393064373933
         36616661323633393036393535383166626166638356631650a31633831666366643938313835303 2
         63393934343937373637306162366265383461316334383132626462656463363630613832313562
         38376462666638356 40a31316434353531666665303135376361303765636261353563353838653 9
         65656439626166666363323435613131643066353762333232326232323565376635

 ...
```

To use an inline vaulted variables with an INI inventory you need to store it in a 'vars' file in YAML format, it can reside in host_vars/ or group_vars/ to be automatically picked up or referenced from a play via `vars_files` or `include_vars`.

To run a playbook with this setup, drop the `-k` flag and add a flag for your `vault-id`:

```
ansible-playbook -i inventory --vault-id my_user@~/my-ansible-vault-pw-file first_playbook.yml
```

Or with a prompt instead of the vault password file:

```
ansible-playbook -i inventory --vault-id my_user@prompt first_playbook.yml
```

To see the original value, you can use the debug module. Please note if your YAML file defines the *ansible_connection* variable (as we used in our example), it will take effect when you execute the command below. To prevent this, please make a copy of the file without the ansible_connection variable.

```
cat vyos.yml | grep -v ansible_connection >> vyos_no_connection.yml

ansible localhost -m debug -a var="ansible_password" -e "@vyos_no_connection.yml" --ask-vault-pass
Vault password:

localhost | SUCCESS => {
    "ansible_password": "VyOS_SSH_password"
}
```

> **Warning**
>
> Vault content can only be decrypted with the password that was used to encrypt it. If you want to stop using one password and move to a new one, you can update and re-encrypt existing vault content with `ansible-vault rekey myfile`, then provide the old password and the new password. Copies of vault content still encrypted with the old password can still be decrypted with old password.

For more details on building inventory files, see :ref:`the introduction to inventory<intro_inventory>`; for more details on ansible-vault, see :ref:`the full Ansible Vault documentation<vault>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\getting_started\(ansible-devel)(docs)(docsite)(rst)(network)(getting_started)first_inventory.rst, line 429); *backlink***
>
> Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\getting_started\(ansible-devel)(docs)(docsite)(rst)(network)(getting_started)first_inventory.rst, line 429); *backlink*

Unknown interpreted text role "ref".

Now that you understand the basics of commands, playbooks, and inventory, it's time to explore some more complex Ansible Network examples.