Ansible Managed device information

Inventory

Provides/generates a list of ansible managed devices. Allows hierarhical inventory item grouping and a richer device description with the help of variables.

Inventory is passed to the playbook by flag `-i`,`--inventory` or by setting ansible configuration in `ansible.cfg`.

Formating

Supported inventroy file formats:

- TOML
- **YAML**
- Python (scripts)

First inventory file:

```
cat <<'EOF' > hosts
ungrouped:
  hosts:
    N9K-1:
EOF
```

Inventory formating and inspection can be done with comand `ansible-inventory --list`.

```
{
    "_meta": {
        "hostvars": {}
    },
    "all": {
        "children": [
            "ungrouped"
        ]
    },
    "ungrouped": {
        "hosts": [
            "N9K-1"
        ]
    }
}
```

Grouping

At each step check the output of `ansible-inventory --list`. Creating 3 groups **A,B,C**.

```
cat <<'EOF' > hosts
A:
  hosts:
    N9K-1:
B:
  hosts:
    N9K-2:
C:
  hosts:
    N9K-[1:9:2]:
    N9K-[10:12]:
EOF
```

Adding a merged group consisting **B** and **C** group members to group **BC**:

```
cat <<'EOF' >> hosts
BC:
  children:
    B:
    C:
EOF
```

Multiple files

You can arragne Your inventory in multiple files stored in a directory and point Your inventory collection to that directory.

```
mkdir inventory
cat <<'EOF' > inventory/A.yml
A:
  hosts:
    N9K-1:
EOF
cat <<'EOF' > inventory/B.yml
B:
  hosts:
    N9K-2:
EOF
```

Scripts

Available inventory collection methods are (inventory sources):

- Simple comma separated IP/hostname list
- YAML file (used in examples)
- Constructed grouping based on jinja2 expressions and additional variables from jinja2 expressions. Adds logics to currently existing inventory.
- INI file (first static file type)

- Virtualbox
- **Script**

Simple example of an inventory script (although not implemented: link).

Some "scripts" are matured and valued as inventory plugins. These are usually predeveloped scripts for standard itegrations with software suites or compute environments: inventory software (Infoblox, Netbox, Zabbix), environments (VMware, AWS, GCP, Azure, Kubernetes).

An interesting inventory plugin in this network topic might NMAP inventory plugin:

```
mkdir nmap-ansible-inventory
cd nmap-ansible-inventory
```

```
cat <<'EOF' > ansible.cfg
[defaults]
inventory = ./nmap-inventory.yml
host_key_checking = False
interpreter_python = auto
[inventory]
enable_plugins = community.general.nmap
EOF
```

```
cat <<'EOF' > nmap-inventory.yml
plugin: community.general.nmap
strict: false
address: 192.168.227.90-99
port: 22,443
ipv6: false
groups:
  sshers: "ports | selectattr('service', 'equalto', 'ssh')"
  apiers: "ports | selectattr('service', 'equalto', 'https')"
EOF
```

Try it out: `ansible-inventory --list` Read about the plugin: `ansible-doc -t inventory community.general.nmap` *Note: at the time of writing nmap plugin code is broken - needs changes in python file (get_option('key') and not get_option['key'])

Variables

Inventory is the main information needed to start automatically interacting with managed devices. Variables bring additional context that can be used for determining the intended state of the (network) devices. Variables can be set for hosts individually and for groups of devices (decouples one to one mapping of device and variables).

Host

Host variables can easily be set in inventory file(s) inline with the device or in a separate file in hostvars directory.

Directly into inventory file:

```
cd ~/auto_lab/
cat <<'EOF' > hosts
A:
  hosts:
    N9K-1:
      var1_1: inventory_file
B:
  hosts:
    N9K-2:
      var2_1: inventory_file
C:
  hosts:
    N9K-[1:9:2]:
      varX_1: inventory_file
BC:
  children:
    B:
    C:
EOF
```

Or by creating a host_vars file:

```
mkdir host_vars
cat <<'EOF' > host_vars/N9K-1.yml
var1_2: host_vars_file
EOF
```

Try it out: `ansible-inventory --list`

Group

Exactly the same applies to groups. Provding only altered sections for brevity:

```
# From hosts file
BC:
  vars:
    var3_1: inventory_file
  children:
    B:
    C:
```

And in a separate group_vars file:

```
```

```
mkdir group_vars
cat <<'EOF' > group_vars/BC.yml
var3_2: group_vars_file
EOF
```

Try it out: `ansible-inventory --list`

Vault

In simple words, Ansible vault allows encrypting any file that is used by ansible with symmetric encryption so that it would be unreadable when stored, but usable after decryption, when it used.

Security and encryptions is always an important and comprehensive topic. Please take time to dig and test out the necessary features. Here we provide only an introductionary example.

Create a host_vars inventory file that is encrypted:

```
cat <<'EOF' > .password_file
AABBCCDDEEFFGGHHIIJJKK11223344556677889900
EOF
```

```
cat <<'EOF' >> ansible.cfg
vault_password_file = .password_file
EOF
```

Create the file:

```
ansible-vault create host_vars/N9K-2.yml
```

In editing mode (You are using vi 😉) enter this line, save and quit:

```
var2_2: encrypted_host_vars_file
```

vi hints:

- 'i' to insert text
- 'esc' key to exit insert mode
- ':w' + enter to save
- ':q' + enter to exit vi

Try it out: `ansible-inventory --list`

Links

- https://docs.ansible.com/ansible/latest/inventory_guide/intro_inventory.html
- https://www.jeffgeerling.com/blog/creating-custom-dynamic-inventories-ansible
-