Run Your First Command and Playbook

Put the concepts you learned to work with this quick tutorial. Install Ansible, execute a network configuration command manually, execute the same command with Ansible, then create a playbook so you can execute the command any time on multiple network devices.

Prerequisites

Before you work through this tutorial you need:

- Ansible 2.10 (or higher) installed
- One or more network devices that are compatible with Ansible
- Basic Linux command line knowledge
- Basic knowledge of network switch & router configuration

Install Ansible

Install Ansible using your preferred method. See :ref:`installation_guide`. Then return to this tutorial.

> **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_playbook.rst, line 26); *backlink*
>
> Unknown interpreted text role "ref".

Confirm the version of Ansible (must be >= 2.10):

```
ansible --version
```

Establish a manual connection to a managed node

To confirm your credentials, connect to a network device manually and retrieve its configuration. Replace the sample user and device name with your real credentials. For example, for a VyOS router:

```
ssh my_vyos_user@vyos.example.net
show config
exit
```

This manual connection also establishes the authenticity of the network device, adding its RSA key fingerprint to your list of known hosts. (If you have connected to the device before, you have already established its authenticity.)

Run your first network Ansible command

Instead of manually connecting and running a command on the network device, you can retrieve its configuration with a single, stripped-down Ansible command:

```
ansible all -i vyos.example.net, -c ansible.netcommon.network_cli -u my_vyos_user -k -m vyos.vyos.vyos_facts -e ansible_netw
```

The flags in this command set seven values:

- the host group(s) to which the command should apply (in this case, all)
- the inventory (-i, the device or devices to target - without the trailing comma -i points to an inventory file)
- the connection method (-c, the method for connecting and executing ansible)
- the user (-u, the username for the SSH connection)
- the SSH connection method (-k, please prompt for the password)
- the module (-m, the Ansible module to run, using the fully qualified collection name (FQCN))
- an extra variable (-e, in this case, setting the network OS value)

NOTE: If you use `ssh-agent` with ssh keys, Ansible loads them automatically. You can omit `-k` flag.

> **Note**
>
> If you are running Ansible in a virtual environment, you will also need to add the variable `ansible_python_interpreter=/path/to/venv/bin/python`

Create and run your first network Ansible Playbook

If you want to run this command every day, you can save it in a playbook and run it with `ansible-playbook` instead of `ansible`. The playbook can store a lot of the parameters you provided with flags at the command line, leaving less to type at the command line. You need two files for this - a playbook and an inventory file.

1. Download :download:`first_playbook.yml <sample_files/first_playbook.yml>`, which looks like this:

> **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_playbook.rst, line 79); *backlink*
>
> Unknown interpreted text role "download".

> **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_playbook.rst, line 81)
>
> Unknown directive type "literalinclude".
>
> ```
> .. literalinclude:: sample_files/first_playbook.yml
>    :language: YAML
> ```

The playbook sets three of the seven values from the command line above: the group (`hosts: all`), the connection method (`connection: ansible.netcommon.network_cli`) and the module (in each task). With those values set in the playbook, you can omit them on the command line. The playbook also adds a second task to show the config output. When a module runs in a playbook, the output is held in memory for use by future tasks instead of written to the console. The debug task here lets you see the results in your shell.

2. Run the playbook with the command:

```
ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos first_playbook.yml
```

The playbook contains one play with two tasks, and should generate output like this:

```
$ ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos first_playbook.yml

PLAY [First Playbook]
**********************************************************************************************************************************

TASK [Get config for VyOS devices]
**********************************************************************************************************************************
ok: [vyos.example.net]

TASK [Display the config]
**********************************************************************************************************************************
ok: [vyos.example.net] => {
    "msg": "The hostname is vyos and the OS is VyOS 1.1.8"
}
```

3. Now that you can retrieve the device config, try updating it with Ansible. Download :download:`first_playbook_ext.yml <sample_files/first_playbook_ext.yml>`, which is an extended version of the first playbook:

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_playbook.rst`, **line 111**); *backlink*

Unknown interpreted text role "download".

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_playbook.rst`, **line 113**)

Unknown directive type "literalinclude".

```
.. literalinclude:: sample_files/first_playbook_ext.yml
   :language: YAML
```

The extended first playbook has four tasks in a single play. Run it with the same command you used above. The output shows you the change Ansible made to the config:

```
$ ansible-playbook -i vyos.example.net, -u ansible -k -e ansible_network_os=vyos.vyos.vyos first_playbook_ext.yml

PLAY [First Playbook]
**********************************************************************************************************************************

TASK [Get config for VyOS devices]
**********************************************************************************************************************************
ok: [vyos.example.net]

TASK [Display the config]
**********************************************************************************************************************************
ok: [vyos.example.net] => {
    "msg": "The hostname is vyos and the OS is VyOS 1.1.8"
}

TASK [Update the hostname]
**********************************************************************************************************************************
changed: [vyos.example.net]

TASK [Get changed config for VyOS devices]
**********************************************************************************************************************************
ok: [vyos.example.net]

TASK [Display the changed config]
**********************************************************************************************************************************
ok: [vyos.example.net] => {
    "msg": "The new hostname is vyos-changed and the OS is VyOS 1.1.8"
}

PLAY RECAP
**********************************************************************************************************************************
vyos.example.net           : ok=5    changed=1    unreachable=0    failed=0
```

Gathering facts from network devices

The `gather_facts` keyword now supports gathering network device facts in standardized key/value pairs. You can feed these network facts into further tasks to manage the network device.

You can also use the new `gather_network_resources` parameter with the network `*_facts` modules (such as :ref:`arista.eos.eos_facts <ansible_collections.arista.eos.eos_facts_module>`) to return just a subset of the device configuration, as shown below.

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_playbook.rst`, **line 162**); *backlink*

Unknown interpreted text role "ref".

```
- hosts: arista
  gather_facts: True
  gather_subset: interfaces
  module_defaults:
    arista.eos.eos_facts:
```

```
      gather_network_resources: interfaces
```

The playbook returns the following interface facts:

```
"network_resources": {
      "interfaces": [
          {
              "description": "test-interface",
              "enabled": true,
              "mtu": "512",
              "name": "Ethernet1"
          },
          {
              "enabled": true,
              "mtu": "3000",
              "name": "Ethernet2"
          },
          {
              "enabled": true,
              "name": "Ethernet3"
          },
          {
              "enabled": true,
              "name": "Ethernet4"
          },
          {
              "enabled": true,
              "name": "Ethernet5"
          },
          {
              "enabled": true,
              "name": "Ethernet6"
          },
      ]
  }
```

Note that this returns a subset of what is returned by just setting `gather_subset: interfaces`.

You can store these facts and use them directly in another task, such as with the :ref:`eos_interfaces <ansible_collections.arista.eos.eos_interfaces_module>` resource module.

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_playbook.rst`, line 212); *backlink*

Unknown interpreted text role "ref".