Ansible Control Node, Inventory & Managed Nodes

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Session - 11 Agenda:

1. Ansible Facts:
(a) Ansible Facts Introduction
(b) Custom Facts in Ansible
(c) Magic Variables

Ansible Facts:

Ansible facts are system properties that are collected by Ansible when it executes on a remote system. The facts contain useful details such as storage and network configuration about a target system. They may be outputted to a file as a type of system report, or they may be used during Ansible playbook execution to make runtime decisions.

Variables related to remote systems are called facts. With facts, you can use the behavior or state of one system as configuration on other systems. Facts are the kind of collection of ansible variables which we are not by default defined/set/pass by the user anywhere in the ansible environment, but user can create custom facts on the managed nodes if necessary.

Let's execute ansible Ad-hoc command using setup module to print all the information related to the managed nodes.

$ ansible all -m setup

You can also filter facts as below to retrieve only required information.

$ ansible all -m setup -a 'filter=ansible_hostname'

Setup module is used to gather the system information from the managed nodes. Information which is gathered by the setup module from the managed nodes for further use is known as ansible facts.

Let's print the kernel version information of all the managed nodes using ansible Ad-hoc command.

$ ansible all -m setup -a 'filter=*kernel*'

If you know the exact name of the fact, you can mention the name without asterisk.

$ ansible all -m setup -a 'filter=ansible_kernel'

To append the output in a file, we can use re-directional operator.

$ ansible all -m setup -a 'filter=ansible_kernel' > kernel_report.txt

$ cat kernel_report.txt

Let's create a playbook for checking the kernel version of the managed nodes and appending the output in a file name as kernel_version.txt

$ mkdir playbooks
$ cd playbooks

Let's create a file without mentioning any task in it.

$ vim ansible_facts.yml

```
- name: Ansible facts use case example playbook.
  hosts: all
  become: true
  tasks:
...
```

Check the playbook for syntax errors.

$ ansible-playbook ansible_facts.yml --syntax-check

Now, execute the playbook and see that it will automatically gather the facts from the managed nodes.

$ ansible-playbook ansible_facts.yml

Here you can see that gathering facts task is automatically running without even mentioning it in the playbook. There is no need to use setup module for this task because it is native (default) in ansible to collect managed nodes information.

Let's complete the playbook for the above task (to gather the kernel information).

$ vim ansible_facts.yml

```
- name: Ansible facts use case example playbook.
  hosts: all
  become: true
  tasks:
    - name: Capture the kernel version information.
      copy:
        content: "The available kernel version is {{ ansible_kernel }}"
        dest: /tmp/kernel_version.txt
...
```

Check the playbook for syntax errors.

$ ansible-playbook ansible_facts.yml --syntax-check

Now, execute the playbook.

$ ansible-playbook ansible_facts.yml

Verify the same with the ansible Ad-hoc command, open the file created by the

playbook earlier to see the kernel information.
$ ansible all -m command -a 'cat /tmp/kernel_version.txt'

Let's create another playbook to make an entry in the /etc/hosts file for all the managed nodes so that we can ping the machines using their respective hostnames instead of IP Address.
$ cat /etc/hosts

Also check the details of the /etc/hosts on the managed nodes using Ad-hoc command.
$ ansible all -m command -a 'cat /etc/hosts'

Let's print all the system information using setup module in Ad-hoc command.
$ ansible all -m setup

It prints the output in json format. We can also filter the desired information from this output.
$ ansible all -m setup -a 'filter=*fqdn*'
$ ansible all -m setup -a 'filter=*ipv4*'

Now, we can create the playbook to make an entry in the /etc/hosts file by using the ansible facts.

```
$ vim ansible_facts2.yml
- name: Ansible playbook for making an entry in the /etc/hosts file.
  hosts: all
  become: true
  tasks:
    - name: Append the IP address and hostname information.
      lineinfile:
         line: "{{ ansible_default_ipv4.address }} {{ ansible_default_ipv4.macaddress }}"
         dest: /etc/hosts
...
```

```
$ vim ansible_facts2.yml
- name: Ansible playbook for making an entry in the /etc/hosts file.
  hosts: all
  become: true
  tasks:
    - name: Append the IP address and hostname information.
      lineinfile:
         line: "{{ ansible_default_ipv4.address }} {{ ansible_fqdn }}"
         dest: /etc/hosts
...
```

Now, execute the playbook to make an entry in the /etc/hosts file on the managed nodes for the IP addresses and hostnames.

$ ansible-playbook ansible_facts2.yml

Verify the same using Ad-hoc command.

$ ansible all -m command -a 'cat /etc/hosts'

There is one more method of defining facts in playbook file. We can also use buit-in variables or magics variable as well if needed.

```
$ vim ansible_facts3.yml
- name: Ansible playbook for making an entry in the /etc/hosts file.
  hosts: all
  become: true
  tasks:
    - name: Append the ip address and hostname information.
      lineinfile:
        line: "{{ ansible_default_ipv4['address'] }} {{ ansible_fqdn }} {{ ansible_hostname }}"
        dest: /etc/hosts
...
```

Now, execute the playbook to make an entry in the /etc/hosts file on the managed nodes for the IP addresses and hostnames.

$ ansible-playbook ansible_facts3.yml

Verify the same using Ad-hoc command.

$ ansible all -m command -a 'cat /etc/hosts'

Disable/Enable gather facts module in Ansible:

Let's create a simple playbook to print Hello World.

```
$ vim hello.yml
---
- name: Hello World Printing Playbook
  hosts: all
  tasks:
    - name: Printing Hello World
      debug:
        msg: Hello Ansible World
...
```

Let's execute this playbook.

$ ansible-playbook hello.yml

Ansible Control Node, Inventory & Managed Nodes

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Managing Facts in Playbooks:

In case if we want to disable/enable gathering facts in any ansible playbook we can use gather_facts option.

$ vim hello.yml

```
---
- name: Hello World Printing Playbook
  hosts: all
  gather_facts: false
  tasks:
    - name: Printing Hello World
      debug:
        msg: Hello Ansible World
...
```

Let's execute this playbook.

$ ansible-playbook hello.yml

Managing Facts Globally in Ansible for all the playbooks.

We can use gathering option in the defaults section of the ansible configuration file (/etc/ansible/ansible.cfg), we can set its value to explicit if we want to disable facts gathering while executing the ansible playbooks and we can set its value to implicit if we want to enable facts gathering while executing the ansible playbooks.

$ sudo vim /etc/ansible/ansible.cfg

```
[defaults]
gathering = explicit
```

But we can override this by mention the gather_facts option in the playbook. Because the priority of the playbook option is higher.

$ vim hello.yml

```
---
- name: Hello World Printing Playbook
  hosts: all
  gather_facts: true
  tasks:
    - name: Printing Hello World
      debug:
        msg: Hello Ansible World
...
```

Let's execute this playbook and verify that it gathers the facts.

$ ansible-playbook hello.yml

Ansible Control Node, Inventory & Managed Nodes

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Let's re-execute the first playbook to capture the kernel information. This time it will not gather the facts and will show an error because of missing variable values.

$ ansible-playbook ansible_facts.yml

Now, we can either use gather_facts option to run the playbook or mention implicit in the ansible configuration file.

```
$ vim ansible_facts.yml
- name: Ansible facts use case example playbook.
  hosts: all
  become: true
  gather_facts: true
  tasks:
    - name: Capture the kernel version information.
      copy:
        content: "The available kernel version is {{ ansible_kernel }}"
        dest: /tmp/kernel_version.txt
...
```

Now, we can execute the playbook without any errors.

$ ansible-playbook ansible_facts.yml

OR

```
$ sudo vim /etc/ansible/ansible.cfg
[defaults]
gathering = implicit

$ vim ansible_facts.yml
- name: Ansible facts use case example playbook.
  hosts: all
  become: true
  tasks:
    - name: Capture the kernel version information.
      copy:
        content: "The available kernel version is {{ ansible_kernel }}"
        dest: /tmp/kernel_version.txt
...
```

Now, we can execute the playbook without any errors.

$ ansible-playbook ansible_facts.yml

Ansible Control Node, Inventory & Managed Nodes

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Custom Facts:

We can configure custom facts inside the managed hosts and these facts will be retrieved by setup together with ansible facts. Custom facts can be used in managed hosts to control the play based on custom values.

You can setup local custom variables in facts.d directory – /etc/ansible/facts.d. If the directory doesn't exist, create it. You can also use JSON format for custom facts files.

To create a custom facts file on the managed nodes, follow the steps below.

1. make the /etc/ansible/facts.d directory

$ ssh node1
$ sudo mkdir -p /etc/ansible/facts.d/
$ sudo vim /etc/ansible/facts.d/httpd.fact
[basic]
package=httpd
service=httpd
state=started
enabled=true

2. Use the setup module to verify the custom facts and search for "ansible_local" string.

$ ansible node1 -m setup -a "filter=ansible_local"

```
"ansible_local": {
        "httpd": {
            "basic": {
                "enabled": "true",
                "package": "httpd",
                "service": "httpd",
                "state": "started"
```

Now you can see that the custom facts have been created on the control node. Similarly, you will do the same to create custom facts on all the managed hosts.

$ ansible node1 -m setup | less

```
 "ansible_local": {
        "httpd": {
            "basic": {
                "enabled": "true",
                "package": "httpd",
                "service": "httpd",
                "state": "started"
```

Now you can see the custom facts created on the managed host.

Ansible facts file can be written in INI or JSON format but cannot be written in YAML format. Having understood how to create custom facts, let's look at how to use playbooks with custom facts.

To use a playbook with the custom facts we have created above, follow the steps below, let's create a playbook.

$ vim custom_facts.yml

```
---
- name: Install httpd
  hosts: node1
  become: true
  tasks:
    - name: Install httpd
      yum:
        name: "{{ ansible_facts['ansible_local']['httpd']['basic']['package'] }}"
        state: latest
    - name: Start and enable httpd
      service:
        name: "{{ ansible_facts['ansible_local']['httpd']['basic']['service'] }}"
        state: "{{ ansible_facts['ansible_local']['httpd']['basic']['state'] }}"
        enabled: "{{ ansible_facts['ansible_local']['httpd']['basic']['enabled'] }}"
...
```

Let's run the playbook.

$ ansible-playbook custom_facts.yml

The custom fact created on node1 host above was manually created. In real life environment, it is smart to create /etc/ansible/facts.d/httpd using a playbook.

Let's verify the task status using ansible ad-hoc command.

$ ansible node1 -m command -a 'systemctl status httpd'

Magic Variables:

Magic variables are special variables that cannot be set directly by a user neither by facts but automatically set by ansible by default. Magic variables is also useful to gather information of managed hosts.

There are numbers of magic variables but the most common ones are hostvars, groups, group_names, inventory_hostname, etc. Click here to see other magic variables in Ansible.

Let's use magic_variables.yml we created above as an example of one of the ways to refer to a magic variable in a playbook.

```
$ vim magic_variables.yml
- name: gather facts
  hosts: node1
  tasks:
    - name: gather all facts for {{ inventory_hostname }}
      debug:
        var: ansible_facts
…
```

Let's execute the playbook.

```
$ ansible-playbook magic_variables.yml
```

Thanks