Ansible Tutorial Part 4 – Ansible Facts

Ansible facts are variables that are automatically discovered by Ansible on a managed host. Facts contain host-specific information that can be used just like regular variables in plays, conditions, loops or any other statement that depends on a value collected from a managed host.

Some of the facts gathered from a managed host might include –

- The Hostname
- The Kernel Version
- The Network Interfaces
- The IP Addresses
- The version of the Operating System
- Various environment variables
- The number of CPUs
- The available or free memory
- The available disk space

Facts are a convenient way to retrieve the state of a managed host and to determine what action to take based on that state. For example –

- A server can be restarted by a conditional task which is run based on a fact containing the managed host's current kernel version.
- The MYSQL configuration file can be customized depending on the available memory reported by a fact.
- The IPv4 address used in a configuration file can be set based on the value of a fact.

Normally every play runs the **setup** module automatically before the first task in order to gather facts. This is reported as the **Gathering Facts** task in Ansible 2.3 and later or imply as **setup** in older versions of Ansible. By default, you do not need to have a task to run setup in your play. It is normally run automatically for you. One way to see what facts are gathered for your managed hosts is to run a short playbook that gathers facts and uses the **debug** module to print the value of the **ansible_facts** variable

```
- name: Fact dump
  hosts: all
  tasks:
    - name: Print all facts
      debug:
        var: ansible_facts
```

Example of Ansible Facts

Fact	Variable
Short host name	`ansible_facts['hostname']`
Fully qualified domain name	`ansible_facts['fqdn']`
Main IPv4 address (based on routing)	`ansible_facts['default_ipv4']['address']`
List of the names of all network interfaces	`ansible_facts['interfaces']`
Size of the `/dev/vda1` disk partition	`ansible_facts['devices']['vda']['partitions']['vda1']['size']`
List of DNS servers	`ansible_facts['dns']['nameservers']`
Version of the currently running kernel	`ansible_facts['kernel']`

Remember when a variable's value is a hash/directory, there are two syntaxes that can be used to retrieve the value. To take two examples from the preceding table –

- `ansible_facts['default_ipv4']['address']` can also be written `ansible_facts.default_ipv4.address`
- `ansible_facts['dns']['nameservers']` can also be written `ansible_facts.dns.nameservers`

When a fact is used in playbook, Ansible dynamically substitutes the variable name for the fact with the corresponding value –

```
---
- hosts: all
  tasks:
  - name: Prints various Ansible facts
    debug:
      msg: >
        The default IPv4 address of {{ ansible_facts.fqdn }}
        is {{ ansible_facts.default_ipv4.address }}
```

The following output shows how Ansible was able to query the managed node and dynamically use the system information to update the variable. Facts can also be used to create dynamic groups of hosts that match particular criteria.

```
[user@demo ~]$ ansible-playbook playbook.yml
PLAY ***************************************************************************

TASK [Gathering Facts] *********************************************************
ok: [demo1.example.com]

TASK [Prints various Ansible facts] ********************************************
ok: [demo1.example.com] => {
    "msg": "The default IPv4 address of demo1.example.com is
            172.25.250.10"
}

PLAY RECAP *********************************************************************
demo1.example.com    : ok=2    changed=0    unreachable=0    failed=0
```

Ansible Facts Injected as Variables

Before Ansible 2.5, facts were injected as individual variables prefixed with the string **ansible_** instead of being part of the **ansible_facts** variable. For example, the ansible_facts['distribution'] fact would have been called **ansible_distribution.**

Many older playbooks still use facts injected as variables instead of the new syntax that is namespaced under the **ansible_facts** variable. You can use an ad hoc command to run the **setup** module to print the value of all facts in this form. In the following example , an ad hoc command is used to run the setup module on the managed host demo1.example.com

```
[user@demo ~]$ ansible demo1.example.com -m setup
demo1.example.com | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "172.25.250.10"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::5054:ff:fe00:fa0a"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "01/01/2011",
        "ansible_bios_version": "0.5.1",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-327.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "console": "tty50,115200n8",
            "crashkernel": "auto",
            "net.ifnames": "0",
            "no_timer_check": true,
            "ro": true,
            "root": "UUID=2460ab6e-e869-4011-acae-31b2e8c05a3b"
        }
```

The following table compares the old and new fact names –

ansible_facts form	Old fact variable form
ansible_facts['hostname']	ansible_hostname
ansible_facts['fqdn']	ansible_fqdn
ansible_facts['default_ipv4']['address']	ansible_default_ipv4['address']
ansible_facts['interfaces']	ansible_interfaces
ansible_facts['devices']['vda']['partitions']['vda1']['size']	ansible_devices['vda']['partitions']['vda1']['size']
ansible_facts['dns']['nameservers']	ansible_dns['nameservers']
ansible_facts['kernel']	ansible_kernel

Important

Currently, Ansible recognizes both the new fact naming system (using ansible_facts) and the old pre-2.5 "facts injected as separate variables" naming system.
You can turn off the old naming system by setting the **inject_facts_as_vars** parameter in the [default] section of the Ansible configuration file to false. The default setting is currently true.

Turning off Fact Gathering

Sometimes, you do not want to gather facts for your play. There are a couple of reasons why this might be the case. It might be that you are not using any facts and want to speed up the play or reduce load caused by the play on the managed hosts. It might be that the managed hosts can not run the setup module for some reason or need to install prerequisite software before gathering facts.
To disable fact gathering for a play, set the **gather_facts** keyword to no –

```
---
- name: This play gathers no facts automatically
  hosts: large_farm
  gather_facts: no
```

Even if gather_facts : no is set for a play, you can manually gather facts at any time by running a task that uses the setup module

```
  tasks:
    - name: Manually gather facts
      setup:
...output omitted...
```

Creating Custom Facts

Administrators can create custom facts which are stored locally on each managed host. These facts are integrated into the list of standard facts gathered by the setup module when it runs on the managed host. These allow the managed host to provide arbitrary variables to Ansible which can be used to adjust the behavior of plays.
Custom facts can be defined in a static file, formatted **as an INI file or using JSON**. They can also be executable scripts which generate JSON output, just like a dynamic inventory script.
Custom facts allow administrators to define certain values for managed hosts which plays might use to populate configuration files or conditionally run tasks. Dynamic custom facts allow the values for these facts or even which facts are provided, to be determined programmatically when the play is run.
By default, the setup module loads custom facts from files and scripts in each managed host's **/etc/ansible/facts.d** directory. The name of each file or script must end in **. fact** in order to be used.
Dynamic custom facts scripts must output JSON-formatted facts and must be executable.
This is an example of a static custom facts file written in INI format. An INI-formatted custom facts file contains a top level defined by a section, followed by the key-value pairs of the facts to define:

```
[packages]
web_package = httpd
db_package = mariadb-server

[users]
user1 = joe
user2 = jane
```

The same facts could be provided in JSON format. The following JSON facts are equivalent to the facts specified by the INI format in the preceding example. The JSON data could be stored in a static text file or printed output by an executable script:

```
{
  "packages": {
    "web_package": "httpd",
    "db_package": "mariadb-server"
  },
  "users": {
    "user1": "joe",
    "user2": "jane"
  }
}
```

Custom facts are stored by the setup module in the **ansible_facts.ansible_local** variable. Facts are organized based on the name of the file that defined them. For example assume the preceding custom facts are produced by a file saved as **/etc/ansible/facts.d/custom.fact** on the managed host. In that case, the value of **ansible_facts.ansible_local[‘custom’][‘users’][‘user1’]** is joe.

You can inspect the structure of your custom facts by running the setup module on the managed host with an ad hoc command

```
[user@demo ~]$ ansible demo1.example.com -m setup
demo1.example.com | SUCCESS => {
    "ansible_facts": {
...output omitted...
        "ansible_local": {
            "custom": {
                "packages": {
                    "db_package": "mariadb-server",
                    "web_package": "httpd"
                },
                "users": {
                    "user1": "joe",
                    "user2": "jane"
                }
            }
        },
...output omitted...
    },
    "changed": false
}
```

Custom facts can be used the same way as default facts in the playbooks

```
[user@demo ~]$ cat playbook.yml
---
- hosts: all
  tasks:
  - name: Prints various Ansible facts
    debug:
      msg: >
           The package to install on {{ ansible_facts['fqdn'] }}
           is {{ ansible_facts['ansible_local']['custom']['packages']['web_package'] }}

[user@demo ~]$ ansible-playbook playbook.yml
PLAY ***************************************************************************

TASK [Gathering Facts] *********************************************************
ok: [demo1.example.com]

TASK [Prints various Ansible facts] ********************************************
ok: [demo1.example.com] => {
    "msg": "The package to install on demo1.example.com  is httpd"
}

PLAY RECAP *********************************************************************
demo1.example.com    : ok=2    changed=0    unreachable=0    failed=0
```

Using Magic Variables

Some variables are not facts or configured through the setup module but are also automatically set by Ansible. These magic variables can also be useful to get information specific to a particular managed host.
Four of the most useful magic variables are –
hostvars – Contains the variables for managed hosts and can be used to get the values for another managed host's variables. It does not include the managed host's facts if they have not yet been gathered for that host.
group_names – Lists all groups the current managed host is in.
groups – Lists all groups and hosts in the inventory.
inventory_hostname – Contains the hostname for the current managed host as configured in the inventory. This may be different from the host name reported by facts for various reasons.

Submit a Comment

Your email address will not be published. Required fields are marked *
Save my name, email, and website in this browser for the next time I comment.