Red Hat Enterprise Linux Automation with Ansible

FEEDBACK TRANSLATIONS CERTIFICATE OF ATTENDANCE

Managing Facts ☆

Objectives

After completing this section, you should be able to reference data about managed hosts using Ansible facts, and configure custom facts on managed hosts.

Describing 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, conditionals, loops, or any other statement that depends on a value collected from a managed host.

Some of the facts gathered for a managed host might include:

- The host name
- 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 simply 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
```

When you run the playbook, the facts are displayed in the job output:

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

PLAY [Fact dump] ************************************************************

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

TASK [Print all facts] ******************************************************
ok: [demo1.example.com] => {
    "ansible_facts": {
        "all_ipv4_addresses": [
            "172.25.250.10"
        ],
        "all_ipv6_addresses": [
           "fe80::5054:ff:fe00:fa0a"
        ],
        "ansible_local": {},
        "apparmor": {
            "status": "disabled"
        },
        "architecture": "x86_64",
        "bios_date": "01/01/2011",
        "bios_version": "0.5.1",
        "cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-327.el7.x86_64",
            "LANG": "en_US.UTF-8",
            "console": "ttyS0,115200n8",
            "crashkernel": "auto",
            "net.ifnames": "0",
            "no_timer_check": true,
            "ro": true,
            "root": "UUID=2460ab6e-e869-4011-acae-31b2e8c05a3b"
        },
...output omitted...
```

The playbook displays the content of the `ansible_facts` variable in JSON format as a hash/dictionary of variables. You can browse the output to see what facts are gathered, to find facts that you might want to use in your plays.

The following table shows some facts which might be gathered from a managed node and may be useful in a playbook:

Table 3.3. Examples 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']`

> **NOTE**
>
> Remember that when a variable's value is a hash/dictionary, 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 a 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": "ttyS0,115200n8",
            "crashkernel": "auto",
            "net.ifnames": "0",
            "no_timer_check": true,
            "ro": true,
            "root": "UUID=2460ab6e-e869-4011-acae-31b2e8c05a3b"
        }
...output omitted...
```

The following table compares the old and new fact names.

Table 3.4. Comparison of Selected Ansible 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`.

The default value of `inject_facts_as_vars` will probably change to `false` in a future version of Ansible. If it is set to `false`, you can only reference Ansible facts using the new `ansible_facts.*` naming system. In that case, attempts to reference facts through the old namespace results in the following error:

```
...output omitted...
TASK [Show me the facts] ********************************************
fatal: [demo.example.com]: FAILED! => {"msg": "The task includes an option with an undefined variable. The
error was: 'ansible_distribution' is undefined\n\nThe error appears to have been in
 '/home/student/demo/playbook.yml': line 5, column 7, but may\nbe elsewhere in the file depending on the ex
act syntax problem.\n\nThe offending line appears to be:\n\n  tasks:\n    - name: Show me the facts\n
^ here\n"}
...output omitted...
```

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 cannot run the `setup` module for some reason, or need to install some 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.

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 fact 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 to standard output by an executable script:

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

> **NOTE**
>
> Custom fact files cannot be in YAML format like a playbook. JSON format is the closest equivalent.

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 that 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 hosts 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 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 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 host name for the current managed host as configured in the inventory. This may be different from the host name reported by facts for various reasons.

There are a number of other "magic variables" as well. For more information, see https://docs.ansible.com/ansible/2.9/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable (https://docs.ansible.com/ansible/2.9/user_guide/playbooks_variables.html#variable-precedence-where-should-i-put-a-variable). One way to get insight into their values is to use the `debug` module to report on the contents of the `hostvars` variable for a particular host:

```
[user@demo ~]$ ansible localhost -m debug -a 'var=hostvars["localhost"]'
localhost | SUCCESS => {
    "hostvars[\"localhost\"]": {
        "ansible_check_mode": false,
        "ansible_connection": "local",
        "ansible_diff_mode": false,
        "ansible_facts": {},
        "ansible_forks": 5,
        "ansible_inventory_sources": [
            "/home/student/demo/inventory"
        ],
        "ansible_playbook_python": "/usr/bin/python2",
        "ansible_python_interpreter": "/usr/bin/python2",
        "ansible_verbosity": 0,
        "ansible_version": {
            "full": "2.7.0",
            "major": 2,
            "minor": 7,
            "revision": 0,
            "string": "2.7.0"
        },
        "group_names": [],
        "groups": {
            "all": [
                "serverb.lab.example.com"
            ],
            "ungrouped": [],
            "webservers": [
                "serverb.lab.example.com"
            ]
        },
        "inventory_hostname": "localhost",
        "inventory_hostname_short": "localhost",
        "omit": "__omit_place_holder__18d132963728b2cbf7143dd49dc4bf5745fe5ec3",
        "playbook_dir": "/home/student/demo"
    }
}
```

REFERENCES

setup - Gathers facts about remote hosts — Ansible Documentation (https://docs.ansible.com/ansible/2.9/modules/setup_module.html)

Variables discovered from systems: Facts — Ansible Documentation (https://docs.ansible.com/ansible/2.9/user_guide/playbooks_variables.html#variables-discovered-from-systems-facts)

← PREVIOUS (/ROL/APP/COURSES/RH294-8.4/PAGES/CH03S04) → NEXT (/ROL/APP/COURSES/RH294-8.4/PAGES/CH03S06)

Privacy Policy (http://s.bl-1.com/h/cZrgWbQn?url=https://www.redhat.com/en/about/privacy-policy?extIdCarryOver=true&sc_cid=701f2000001D8QoAAK)

Red Hat Training Policies (http://s.bl-1.com/h/cZrb2DXG?url=https://www.redhat.com/en/about/red-hat-training-policies)

Terms of Use (https://www.redhat.com/en/about/terms-use)

All policies and guidelines (https://www.redhat.com/en/about/all-policies-guidelines)

Release Notes (https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/Red-Hat-Learning-Subscription-Release-Notes/ba-p/22952)

Cookie Preferences