Discovering variables: facts and magic variables

With Ansible you can retrieve or discover certain variables containing information about your remote systems or about Ansible itself. 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. For example, you can use the IP address of one system as a configuration value on another system. Variables related to Ansible are called magic variables.

Ansible facts

Ansible facts are data related to your remote systems, including operating systems, IP addresses, attached filesystems, and more. You can access this data in the `ansible_facts` variable. By default, you can also access some Ansible facts as top-level variables with the `ansible_` prefix. You can disable this behavior using the INJECT_FACTS_AS_VARS (../reference_appendices/config.html#inject-facts-as-vars) setting. To see all available facts, add this task to a play:

```
- name: Print all available facts
  ansible.builtin.debug:
    var: ansible_facts
```

To see the 'raw' information as gathered, run this command at the command line:

```
ansible <hostname> -m ansible.builtin.setup
```

Facts include a large amount of variable data, which may look like this:

Search this site

```
{
    "ansible_all_ipv4_addresses": [
        "REDACTED IP ADDRESS"
    ],
    "ansible_all_ipv6_addresses": [
        "REDACTED IPV6 ADDRESS"
    ],
    "ansible_apparmor": {
        "status": "disabled"
    },
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "11/28/2013",
    "ansible_bios_version": "4.1.5",
    "ansible_cmdline": {
        "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-862.14.4.el7.x86_64",
        "console": "ttyS0,115200",
        "no_timer_check": true,
        "nofb": true,
        "nomodeset": true,
        "ro": true,
        "root": "LABEL=cloudimg-rootfs",
        "vga": "normal"
    },
    "ansible_date_time": {
        "date": "2018-10-25",
        "day": "25",
        "epoch": "1540469324",
        "hour": "12",
        "iso8601": "2018-10-25T12:08:44Z",
        "iso8601_basic": "20181025T120844109754",
        "iso8601_basic_short": "20181025T120844",
        "iso8601_micro": "2018-10-25T12:08:44.109968Z",
        "minute": "08",
        "month": "10",
        "second": "44",
        "time": "12:08:44",
        "tz": "UTC",
        "tz_offset": "+0000",
        "weekday": "Thursday",
        "weekday_number": "4",
        "weeknumber": "43",
        "year": "2018"
    },
    "ansible_default_ipv4": {
        "address": "REDACTED",
        "alias": "eth0",
        "broadcast": "REDACTED",
        "gateway": "REDACTED",
        "interface": "eth0",
        "macaddress": "REDACTED",
        "mtu": 1500,
        "netmask": "255.255.255.0",
        "network": "REDACTED",
        "type": "ether"
    },
    "ansible_default_ipv6": {},
    "ansible_device_links": {
        "ids": {},
        "labels": {
            "xvda1": [
                "cloudimg-rootfs"
            ],
            "xvdd": [
```

Search this site

```
                "config-2"
            ]
        },
        "masters": {},
        "uuids": {
            "xvda1": [
                "cac81d61-d0f8-4b47-84aa-b48798239164"
            ],
            "xvdd": [
                "2018-10-25-12-05-57-00"
            ]
        }
    },
    "ansible_devices": {
        "xvda": {
            "holders": [],
            "host": "",
            "links": {
                "ids": [],
                "labels": [],
                "masters": [],
                "uuids": []
            },
            "model": null,
            "partitions": {
                "xvda1": {
                    "holders": [],
                    "links": {
                        "ids": [],
                        "labels": [
                            "cloudimg-rootfs"
                        ],
                        "masters": [],
                        "uuids": [
                            "cac81d61-d0f8-4b47-84aa-b48798239164"
                        ]
                    },
                    "sectors": "83883999",
                    "sectorsize": 512,
                    "size": "40.00 GB",
                    "start": "2048",
                    "uuid": "cac81d61-d0f8-4b47-84aa-b48798239164"
                }
            },
            "removable": "0",
            "rotational": "0",
            "sas_address": null,
            "sas_device_handle": null,
            "scheduler_mode": "deadline",
            "sectors": "83886080",
            "sectorsize": "512",
            "size": "40.00 GB",
            "support_discard": "0",
            "vendor": null,
            "virtual": 1
        },
        "xvdd": {
            "holders": [],
            "host": "",
            "links": {
                "ids": [],
                "labels": [
                    "config-2"
                ],
```

Search this site

```
                "masters": [],
                "uuids": [
                    "2018-10-25-12-05-57-00"
                ]
            },
            "model": null,
            "partitions": {},
            "removable": "0",
            "rotational": "0",
            "sas_address": null,
            "sas_device_handle": null,
            "scheduler_mode": "deadline",
            "sectors": "131072",
            "sectorsize": "512",
            "size": "64.00 MB",
            "support_discard": "0",
            "vendor": null,
            "virtual": 1
        },
        "xvde": {
            "holders": [],
            "host": "",
            "links": {
                "ids": [],
                "labels": [],
                "masters": [],
                "uuids": []
            },
            "model": null,
            "partitions": {
                "xvde1": {
                    "holders": [],
                    "links": {
                        "ids": [],
                        "labels": [],
                        "masters": [],
                        "uuids": []
                    },
                    "sectors": "167770112",
                    "sectorsize": 512,
                    "size": "80.00 GB",
                    "start": "2048",
                    "uuid": null
                }
            },
            "removable": "0",
            "rotational": "0",
            "sas_address": null,
            "sas_device_handle": null,
            "scheduler_mode": "deadline",
            "sectors": "167772160",
            "sectorsize": "512",
            "size": "80.00 GB",
            "support_discard": "0",
            "vendor": null,
            "virtual": 1
        }
    },
    "ansible_distribution": "CentOS",
    "ansible_distribution_file_parsed": true,
    "ansible_distribution_file_path": "/etc/redhat-release",
    "ansible_distribution_file_variety": "RedHat",
    "ansible_distribution_major_version": "7",
    "ansible_distribution_release": "Core",
```

Search this site

```
        "ansible_distribution_version": "7.5.1804",
        "ansible_dns": {
            "nameservers": [
                "127.0.0.1"
            ]
        },
        "ansible_domain": "",
        "ansible_effective_group_id": 1000,
        "ansible_effective_user_id": 1000,
        "ansible_env": {
            "HOME": "/home/zuul",
            "LANG": "en_US.UTF-8",
            "LESSOPEN": "||/usr/bin/lesspipe.sh %s",
            "LOGNAME": "zuul",
            "MAIL": "/var/mail/zuul",
            "PATH": "/usr/local/bin:/usr/bin",
            "PWD": "/home/zuul",
            "SELINUX_LEVEL_REQUESTED": "",
            "SELINUX_ROLE_REQUESTED": "",
            "SELINUX_USE_CURRENT_RANGE": "",
            "SHELL": "/bin/bash",
            "SHLVL": "2",
            "SSH_CLIENT": "REDACTED 55672 22",
            "SSH_CONNECTION": "REDACTED 55672 REDACTED 22",
            "USER": "zuul",
            "XDG_RUNTIME_DIR": "/run/user/1000",
            "XDG_SESSION_ID": "1",
            "_": "/usr/bin/python2"
        },
        "ansible_eth0": {
            "active": true,
            "device": "eth0",
            "ipv4": {
                "address": "REDACTED",
                "broadcast": "REDACTED",
                "netmask": "255.255.255.0",
                "network": "REDACTED"
            },
            "ipv6": [
                {
                    "address": "REDACTED",
                    "prefix": "64",
                    "scope": "link"
                }
            ],
            "macaddress": "REDACTED",
            "module": "xen_netfront",
            "mtu": 1500,
            "pciid": "vif-0",
            "promisc": false,
            "type": "ether"
        },
        "ansible_eth1": {
            "active": true,
            "device": "eth1",
            "ipv4": {
                "address": "REDACTED",
                "broadcast": "REDACTED",
                "netmask": "255.255.224.0",
                "network": "REDACTED"
            },
            "ipv6": [
                {
                    "address": "REDACTED",
```

Search this site

```
                "prefix": "64",
                "scope": "link"
            }
        ],
        "macaddress": "REDACTED",
        "module": "xen_netfront",
        "mtu": 1500,
        "pciid": "vif-1",
        "promisc": false,
        "type": "ether"
    },
    "ansible_fips": false,
    "ansible_form_factor": "Other",
    "ansible_fqdn": "centos-7-rax-dfw-0003427354",
    "ansible_hostname": "centos-7-rax-dfw-0003427354",
    "ansible_interfaces": [
        "lo",
        "eth1",
        "eth0"
    ],
    "ansible_is_chroot": false,
    "ansible_kernel": "3.10.0-862.14.4.el7.x86_64",
    "ansible_lo": {
        "active": true,
        "device": "lo",
        "ipv4": {
            "address": "127.0.0.1",
            "broadcast": "host",
            "netmask": "255.0.0.0",
            "network": "127.0.0.0"
        },
        "ipv6": [
            {
                "address": "::1",
                "prefix": "128",
                "scope": "host"
            }
        ],
        "mtu": 65536,
        "promisc": false,
        "type": "loopback"
    },
    "ansible_local": {},
    "ansible_lsb": {
        "codename": "Core",
        "description": "CentOS Linux release 7.5.1804 (Core)",
        "id": "CentOS",
        "major_release": "7",
        "release": "7.5.1804"
    },
    "ansible_machine": "x86_64",
    "ansible_machine_id": "2db133253c984c82aef2fafcce6f2bed",
    "ansible_memfree_mb": 7709,
    "ansible_memory_mb": {
        "nocache": {
            "free": 7804,
            "used": 173
        },
        "real": {
            "free": 7709,
            "total": 7977,
            "used": 268
        },
        "swap": {
                "cached": 0,
                "free": 0,
                "total": 0,
                "used": 0
            }
        },
        "ansible_memtotal_mb": 7977,
        "ansible_mounts": [
            {
                "block_available": 7220998,
                "block_size": 4096,
                "block_total": 9817227,
                "block_used": 2596229,
                "device": "/dev/xvda1",
                "fstype": "ext4",
                "inode_available": 10052341,
                "inode_total": 10419200,
                "inode_used": 366859,
                "mount": "/",
                "options": "rw,seclabel,relatime,data=ordered",
                "size_available": 29577207808,
                "size_total": 40211361792,
                "uuid": "cac81d61-d0f8-4b47-84aa-b48798239164"
            },
            {
                "block_available": 0,
                "block_size": 2048,
                "block_total": 252,
                "block_used": 252,
                "device": "/dev/xvdd",
                "fstype": "iso9660",
                "inode_available": 0,
                "inode_total": 0,
                "inode_used": 0,
                "mount": "/mnt/config",
                "options": "ro,relatime,mode=0700",
                "size_available": 0,
                "size_total": 516096,
                "uuid": "2018-10-25-12-05-57-00"
            }
        ],
        "ansible_nodename": "centos-7-rax-dfw-0003427354",
        "ansible_os_family": "RedHat",
        "ansible_pkg_mgr": "yum",
        "ansible_processor": [
            "0",
            "GenuineIntel",
            "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
            "1",
            "GenuineIntel",
            "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
            "2",
            "GenuineIntel",
            "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
            "3",
            "GenuineIntel",
            "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
            "4",
            "GenuineIntel",
            "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
            "5",
            "GenuineIntel",
            "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
            "6",
        "GenuineIntel",
        "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz",
        "7",
        "GenuineIntel",
        "Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz"
    ],
    "ansible_processor_cores": 8,
    "ansible_processor_count": 8,
    "ansible_processor_nproc": 8,
    "ansible_processor_threads_per_core": 1,
    "ansible_processor_vcpus": 8,
    "ansible_product_name": "HVM domU",
    "ansible_product_serial": "REDACTED",
    "ansible_product_uuid": "REDACTED",
    "ansible_product_version": "4.1.5",
    "ansible_python": {
        "executable": "/usr/bin/python2",
        "has_sslcontext": true,
        "type": "CPython",
        "version": {
            "major": 2,
            "micro": 5,
            "minor": 7,
            "releaselevel": "final",
            "serial": 0
        },
        "version_info": [
            2,
            7,
            5,
            "final",
            0
        ]
    },
    "ansible_python_version": "2.7.5",
    "ansible_real_group_id": 1000,
    "ansible_real_user_id": 1000,
    "ansible_selinux": {
        "config_mode": "enforcing",
        "mode": "enforcing",
        "policyvers": 31,
        "status": "enabled",
        "type": "targeted"
    },
    "ansible_selinux_python_present": true,
    "ansible_service_mgr": "systemd",
    "ansible_ssh_host_key_ecdsa_public": "REDACTED KEY VALUE",
    "ansible_ssh_host_key_ed25519_public": "REDACTED KEY VALUE",
    "ansible_ssh_host_key_rsa_public": "REDACTED KEY VALUE",
    "ansible_swapfree_mb": 0,
    "ansible_swaptotal_mb": 0,
    "ansible_system": "Linux",
    "ansible_system_capabilities": [
        ""
    ],
    "ansible_system_capabilities_enforced": "True",
    "ansible_system_vendor": "Xen",
    "ansible_uptime_seconds": 151,
    "ansible_user_dir": "/home/zuul",
    "ansible_user_gecos": "",
    "ansible_user_gid": 1000,
    "ansible_user_id": "zuul",
    "ansible_user_shell": "/bin/bash",
    "ansible_user_uid": 1000,
    "ansible_userspace_architecture": "x86_64",
    "ansible_userspace_bits": "64",
    "ansible_virtualization_role": "guest",
    "ansible_virtualization_type": "xen",
    "gather_subset": [
        "all"
    ],
    "module_setup": true
}
```

You can reference the model of the first disk in the facts shown above in a template or playbook as:

```
{{ ansible_facts['devices']['xvda']['model'] }}
```

To reference the system hostname:

```
{{ ansible_facts['nodename'] }}
```

You can use facts in conditionals (see Conditionals (playbooks_conditionals.html#playbooks-conditionals)) and also in templates. You can also use facts to create dynamic groups of hosts that match particular criteria, see the group_by module (../collections/ansible/builtin/group_by_module.html#group-by-module) documentation for details.

Note

Because `ansible_date_time` is created and cached when Ansible gathers facts before each playbook run, it can get stale with long-running playbooks. If your playbook takes a long time to run, use the `pipe` filter (for example, `lookup('pipe', 'date +%Y-%m-%d.%H:%M:%S')`) or now() (playbooks_templating_now.html#templating-now) with a Jinja 2 template instead of `ansible_date_time`.

Package requirements for fact gathering

On some distros, you may see missing fact values or facts set to default values because the packages that support gathering those facts are not installed by default. You can install the necessary packages on your remote hosts using the OS package manager. Known dependencies include:

Search this site

- Linux Network fact gathering - Depends on the `ip` binary, commonly included in the `iproute2` package.

Caching facts

Like registered variables, facts are stored in memory by default. However, unlike registered variables, facts can be gathered independently and cached for repeated use. With cached facts, you can refer to facts from one system when configuring a second system, even if Ansible executes the current play on the second system first. For example:

```
{{ hostvars['asdf.example.com']['ansible_facts']['os_family'] }}
```

Caching is controlled by the cache plugins. By default, Ansible uses the memory cache plugin, which stores facts in memory for the duration of the current playbook run. To retain Ansible facts for repeated use, select a different cache plugin. See Cache plugins (../plugins/cache.html#cache-plugins) for details.

Fact caching can improve performance. If you manage thousands of hosts, you can configure fact caching to run nightly, then manage configuration on a smaller set of servers periodically throughout the day. With cached facts, you have access to variables and information about all hosts even when you are only managing a small number of servers.

Disabling facts

By default, Ansible gathers facts at the beginning of each play. If you do not need to gather facts (for example, if you know everything about your systems centrally), you can turn off fact gathering at the play level to improve scalability. Disabling facts may particularly improve performance in push mode with very large numbers of systems, or if you are using Ansible on experimental platforms. To disable fact gathering:

```
- hosts: whatever
  gather_facts: false
```

Adding custom facts

The setup module in Ansible automatically discovers a standard set of facts about each host. If you want to add custom values to your facts, you can write a custom facts module, set temporary facts with a `ansible.builtin.set_fact` task, or provide permanent custom facts using the facts.d directory.

facts.d or local facts

New in version 1.3.

Search this site

You can add static custom facts by adding static files to facts.d, or add dynamic facts by adding executable scripts to facts.d. For example, you can add a list of all users on a host to your facts by creating and running a script in facts.d.

To use facts.d, create an `/etc/ansible/facts.d` directory on the remote host or hosts. If you prefer a different directory, create it and specify it using the `fact_path` play keyword. Add files to the directory to supply your custom facts. All file names must end with `.fact`. The files can be JSON, INI, or executable files returning JSON.

To add static facts, simply add a file with the `.fact` extension. For example, create `/etc/ansible/facts.d/preferences.fact` with this content:

```
[general]
asdf=1
bar=2
```

Note

Make sure the file is not executable as this will break the `ansible.builtin.setup` module.

The next time fact gathering runs, your facts will include a hash variable fact named `general` with `asdf` and `bar` as members. To validate this, run the following:

```
ansible <hostname> -m ansible.builtin.setup -a "filter=ansible_local"
```

And you will see your custom fact added:

```
{
    "ansible_local": {
        "preferences": {
            "general": {
                "asdf" : "1",
                "bar"  : "2"
            }
        }
    }
}
```

The ansible_local namespace separates custom facts created by facts.d from system facts or variables defined elsewhere in the playbook, so variables will not override each other. You can access this custom fact in a template or playbook as:

```
{{ ansible_local['preferences']['general']['asdf'] }}
```

Note

The key part in the key=value pairs will be converted into lowercase inside the ansible_local variable. Using the example above, if the ini file contained `XYZ=3` in the `[general]` section, then you should expect to access it as: `{{ ansible_local['preferences']['general']['xyz'] }}` and not `{{ ansible_local['preferences']['general']['XYZ'] }}`. This is because Ansible uses Python's ConfigParser (https://docs.python.org/3/library/configparser.html) which passes all option names through the optionxform (https://docs.python.org/3/library/configparser.html#ConfigParser.RawConfigParser.optionxform) method and this method's default implementation converts option names to lower case.

You can also use facts.d to execute a script on the remote host, generating dynamic custom facts to the ansible_local namespace. For example, you can generate a list of all users that exist on a remote host as a fact about that host. To generate dynamic custom facts using facts.d:

1. Write and test a script to generate the JSON data you want.
2. Save the script in your facts.d directory.
3. Make sure your script has the `.fact` file extension.
4. Make sure your script is executable by the Ansible connection user.
5. Gather facts to execute the script and add the JSON output to ansible_local.

By default, fact gathering runs once at the beginning of each play. If you create a custom fact using facts.d in a playbook, it will be available in the next play that gathers facts. If you want to use it in the same play where you created it, you must explicitly re-run the setup module. For example:

```
- hosts: webservers
  tasks:

    - name: Create directory for ansible custom facts
      ansible.builtin.file:
        state: directory
        recurse: true
        path: /etc/ansible/facts.d

    - name: Install custom ipmi fact
      ansible.builtin.copy:
        src: ipmi.fact
        dest: /etc/ansible/facts.d

    - name: Re-read facts after adding custom fact
      ansible.builtin.setup:
        filter: ansible_local
```

If you use this pattern frequently, a custom facts module would be more efficient than facts.d.

Information about Ansible: magic variables

You can access information about Ansible operations, including the python version being used, the hosts and groups in inventory, and the directories for playbooks and roles, using “magic” variables. Like connection variables, magic variables are Special Variables (../reference_appendices/special_variables.html#special-variables). Magic variable names are reserved - do not set variables with these names. The variable `environment` is also reserved.

The most commonly used magic variables are `hostvars`, `groups`, `group_names`, and `inventory_hostname`. With `hostvars`, you can access variables defined for any host in the play, at any point in a playbook. You can access Ansible facts using the `hostvars` variable too, but only after you have gathered (or cached) facts. Note that variables defined at play objects are not defined for specific hosts and therefore are not mapped to hostvars.

If you want to configure your database server using the value of a ‘fact’ from another node, or the value of an inventory variable assigned to another node, you can use `hostvars` in a template or on an action line:

```
{{ hostvars['test.example.com']['ansible_facts']['distribution'] }}
```

With `groups`, a list of all the groups (and hosts) in the inventory, you can enumerate all hosts within a group. For example:

```
{% for host in groups['app_servers'] %}
   # something that applies to all app servers.
{% endfor %}
```

Search this site

You can use `groups` and `hostvars` together to find all the IP addresses in a group.

```
{% for host in groups['app_servers'] %}
   {{ hostvars[host]['ansible_facts']['eth0']['ipv4']['address'] }}
{% endfor %}
```

You can use this approach to point a frontend proxy server to all the hosts in your app servers group, to set up the correct firewall rules between servers, and so on. You must either cache facts or gather facts for those hosts before the task that fills out the template.

With `group_names`, a list (array) of all the groups the current host is in, you can create templated files that vary based on the group membership (or role) of the host:

```
{% if 'webserver' in group_names %}
   # some part of a configuration file that only applies to webservers
{% endif %}
```

You can use the magic variable `inventory_hostname`, the name of the host as configured in your inventory, as an alternative to `ansible_hostname` when fact-gathering is disabled. If you have a long FQDN, you can use `inventory_hostname_short`, which contains the part up to the first period, without the rest of the domain.

Other useful magic variables refer to the current play or playbook. These vars may be useful for filling out templates with multiple hostnames or for injecting the list into the rules for a load balancer.

`ansible_play_hosts` is the list of all hosts still active in the current play.

`ansible_play_batch` is a list of hostnames that are in scope for the current 'batch' of the play.

The batch size is defined by `serial`, when not set it is equivalent to the whole play (making it the same as `ansible_play_hosts`).

`ansible_playbook_python` is the path to the python executable used to invoke the Ansible command line tool.

`inventory_dir` is the pathname of the directory holding Ansible's inventory host file.

`inventory_file` is the pathname and the filename pointing to the Ansible's inventory host file.

`playbook_dir` contains the playbook base directory.

`role_path` contains the current role's pathname and only works inside a role.

`ansible_check_mode` is a boolean, set to `True` if you run Ansible with `--check`.

Ansible version

New in version 1.8.

To adapt playbook behavior to different versions of Ansible, you can use the variable `ansible_version`, which has the following structure:

```
{
    "ansible_version": {
        "full": "2.10.1",
        "major": 2,
        "minor": 10,
        "revision": 1,
        "string": "2.10.1"
    }
}
```