Ansible Variables and Facts

What Constitutes a Valid Variable Name?

A variable name includes letters, numbers, underscores or a mix of either 2 or all of them. However, bear in mind that a variable name must always begin with a letter and should not contain spaces.

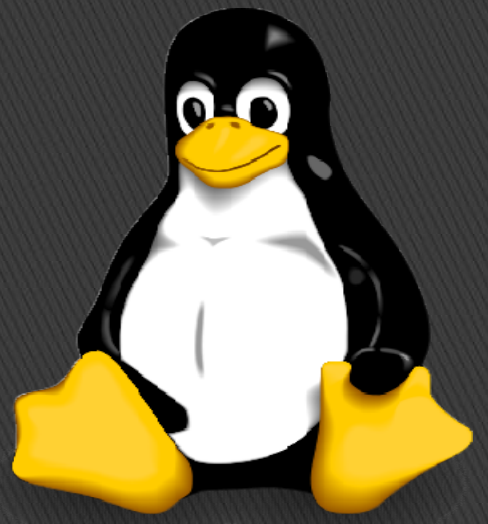
Let's take a look a few examples of valid and unacceptable variable names:

Valid Variable Name Examples:

- football
- foot_ball
- football20
- foot_ball20

Non-valid Variable Name Examples:

- foot ball
- 20
- foot-ball



Playbook Variables

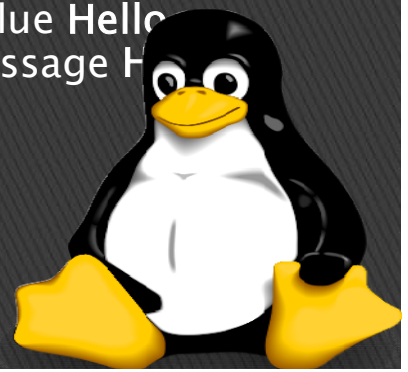
- **Playbook** variables are quite easy and straightforward. To define a variable in a **playbook**, simply use the keyword **vars** before writing your variables with indentation.

- To access the value of the variable, place it between the double curly braces enclosed with quotation marks.
- Here's a simple playbook example:

```
- hosts: all
  vars:
    greeting: Hello world!

  tasks:
  - name: Ansible Basic Variable Example
    debug:
          msg: "{{ greeting }}“
```

- In the above playbook, the **greeting** variable is substituted by the value **Hello world!** when the playbook is run. The playbook simply prints the message **Hello world!** when executed.

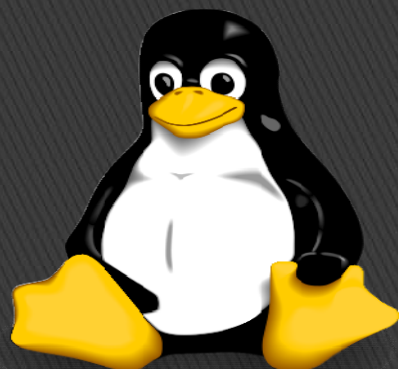


list or an array of variables

- The playbook below shows a variable called **continents.** The variable holds 5 different values – continent names. Each of these values can easily be accessed using **index 0** as the first variable.
- The example of the playbook below retrieves and displays **Asia (Index 1).**

```
- hosts: all
  vars:
   continents:
     - Africa
     - Asia
     - South America
     - North America
     - Europe

  tasks:
   - name: Ansible List variable Example
     debug:
         msg: "{{ continents [1] }}"
```



Array of variables

- The variable list can similarly be structured as shown:

```
vars:
   Continents: [Africa, Asia, South America, North America, Europe]
```

- To list all the items on the list, use the **with_items** module. This will loop through all the values in the array.

```
- hosts: all
  vars:
   continents: [Africa, Asia, South America, North America, Europe]

  tasks:
  - name: Ansible array variables example
    debug:
      msg: "{{ continents }}"
```

Dictionary variable

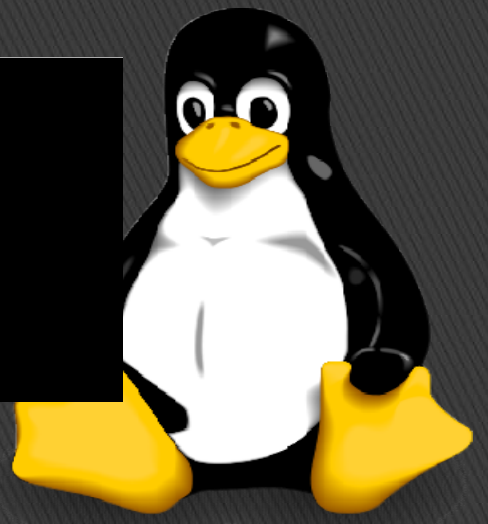
Dictionary variables are additionally supported in the playbook.

```
- name: dictionary variable
  hosts: web
  vars:
     students:
          - Mark:
               city: Melbourne
               address: 0045-0987-8642
          - Angela:
               city: Sydney
               address: 3456-7685-9087
  tasks:
     - debug:
          var: students
```

Special Variables

- Ansible provides a list of predefined variables that can be referenced in Jinja2 templates and playbooks but cannot be altered or defined by the user.
- Collectively, the list of Ansible predefined variables is referred to as Ansible facts and these are gathered when a playbook is executed.
- To get a list of all the Ansible variables, use the setup module in the Ansible ad-hoc command as shown below:
- # ansible -m setup hostname
- This displays the output in JSON format as shown:
- # ansible -m setup localhost

```
[root@rhel-8 ~]# ansible -m setup localhost
localhost | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "173.82.120.14"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::216:3eff:feac:5a7a"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "04/01/2014",
        "ansible_bios_version": "1.11.0-2.el7",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-3.10.0-1062.1.1.el7.centos.plus.x86_64",
            "LANG": "en_US.UTF-8",
            "biosdevname": "0",
            "console": "ttyS0",
            "net.ifnames": "0",
            "nomodeset": true,
            "ro": true,
            "root": "/dev/vda1",
            "vconsole.font": "latarcyrheb-sun16",
            "vconsole.keymap": "us"
        },
        "ansible_date_time": {
            "date": "2019-12-11",
            "day": "11",
            "epoch": "1576106741",
            "hour": "23",
```



some of the examples of Ansible special variables

- ansible_architecture ansible_bios_date ansible_bios_version ansible_date_time ansible_machine ansible_memefree_mb ansible_os_family ansible_selinux

- ansible_architecture
- ansible_bios_date
- ansible_bios_version
- ansible_date_time
- ansible_machine
- ansible_memefree_mb
- ansible_os_family
- ansible_selinux

Inventory Variables

- Lastly, on the list, we have Ansible inventory variables. An inventory is a file in **INI** format that contains all the hosts to be managed by Ansible.
- In inventories, you can assign a variable to a host system and later use it in a playbook.

```
[web_servers]

web_server_1 ansible_user=centos
web_server_2 ansible_user=ubuntu
```

Inventory Variables

- If the host systems share the same variables, you can define another group in the inventory file to make it less cumbersome and avoid unnecessary repetition.
- For example:

```
[web_servers]
web_server_1 ansible_user=centos
web_server_2 ansible_user=centos
```

The above can be structured as:

```
[web_servers]
web_server_1
web_server_2

[web_servers:vars]
ansible_user=centos
```

Ansible Facts

- When running playbooks, the first task that Ansible does is the execution of setup task. I'm pretty sure that you must have come across the output:
- TASK: [Gathering facts] *********
- Ansible facts are nothing but system properties or pieces of information about remote nodes that you have connected to. This information includes the System architecture, the OS version, BIOS information, system time and date, system uptime, IP address, and hardware information to mention just a few.

Ansible Facts

- To get the facts about any system simply use the **setup** module as shown in the command below:

ansible -m setup hostname

- For example:

ansible -m setup database_server

- This prints out a large set of data in **JSON** format as shown:

```
[root@rhel-8 ~]# ansible -m setup database_servers
173.82.202.239 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "173.82.202.239"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::216:3eff:fef3:493e"
        ],
        "ansible_apparmor": {
            "status": "enabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "04/01/2014",
        "ansible_bios_version": "1.11.0-2.el7",
        "ansible_cmdline": {
            "BOOT_IMAGE": "/boot/vmlinuz-4.15.0-50-generic",
            "biosdevname": "0",
            "net.ifnames": "0",
            "quiet": true,
            "ro": true,
            "root": "/dev/vda1",
            "splash": true,
            "vga16fb.modeset": "0"
        },
        "ansible_date_time": {
            "date": "2019-12-12",
            "day": "12",
            "epoch": "1576145994",
            "hour": "10",
            "iso8601": "2019-12-12T10:19:54Z",
            "iso8601_basic": "20191212T1019549793341",
            "iso8601_basic_short": "20191212T101954",
            "iso8601_micro": "2019-12-12T10:19:54.979618Z",
            "minute": "19",
            "month": "12",
            "second": "54",
            "time": "10:19:54",
            "tz": "UTC",
            "tz_offset": "+0000",
            "weekday": "Thursday",
```

Ansible Variable example

```
#vim   /etc/ansible/chseperm.yml
-
 name: play1
 hosts: client
 remote_user: root
 vars:
   status: disabled
 tasks:
   - name: change selinux mode from file
   lineinfile:P
   path: /etc/selinux/config
   regexp: '^SELINUX='
   line: 'SELINUX={{ status }}'
:wq
#ansible-playbook  chseperm.yml   --syntax
#ansible-playbook   chseperm.yml   -e  "status=permissive" -i  hosts
```