First show the debug module to print a message:

1. Print any message using playbooks

We will use module -> debug

this module will print statements during execution and can be usefull for debugging variables and expression

Debug moduel excepts 3 parameters:

> msg

> var

> verbosity

**vim printmessage.yml**

**---**

**- name: Print a message**

**hosts: webserver**

**become: true**

**become\_user: root**

**tasks:**

**- name: print any message**

**debug: msg="Welcome to ansible playbooks"**

**:wq!**

Ansible Variables:

Variables are used to store the value

There are 2 types of variables:

Ansible variables🡺 special variables or magic variables

Special variables have value set by ansible and cannot be changed by user.

These variables cannot be set directly by the user; Ansible will always override them to reflect internal state.

Custom Variables🡪 defined by user

Created by user and values are given by user in playbook or at runtime

Lets see an example with special Variables using Debug Module

vim printmessage.yml

---

- name: Print a message

hosts: webserver

become: true

become\_user: root

tasks:

- name: print any message

debug: msg="The hostname is {{inventory\_hostname}}"

Now add the Var parameter also , as shown below

vim printmessage.yml

- name: Print a message

hosts: localhost:172.31.21.30

tasks:

- name: print any message and variable value

debug:

msg: "The hostname is {{inventory\_hostname}}"

- name: print variable value using var parameter

debug:

var: inventory\_hostname

with msg you have to give parameter in {{}}

with var , no need to give {{}}, only variable name

both will print value

Custom variables:

A variable name can only include letters, numbers, and underscores. [Python keywords](https://docs.python.org/3/reference/lexical_analysis.html#keywords) or [playbook keywords](https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#playbook-keywords) are not valid variable names. A variable name cannot begin with a number.

Variable names can begin with an underscore. In many programming languages, variables that begin with an underscore are private. This is not true in Ansible. Variables that begin with an underscore are treated exactly the same as any other variable.

**Example:**

Show example of tomcat playbook

Vars section and value

- hosts: 172.31.21.30

become: true

become\_user: root

vars:

src\_url: https://tomcat.apache.org/tomcat-7.0-doc/appdev/sample/sample.war

dest\_path: /usr/share/tomcat/webapps

tasks:

- name: Install tomcat

yum: name=tomcat state=present

- name: Start tomcat service

service: name=tomcat state=started

- name: deploy the war file on tomcat server

get\_url:

url={{src\_url}}

dest={{dest\_path}}

- name: restart tomcat server

service: name=tomcat state=restarted

REGISTER MODULE

You can also register the value of a variable, i.e. store the output of a variable

Using register module

Example:

In below playbook, command is executed but you will not see the output at runtime:

**---**

**- hosts: webserver**

**become: true**

**become\_user: root**

**gather\_facts: false**

**tasks:**

**- name: display bash version**

**shell: "bash --version"**

**:wq!**

So we need to use register module to store the output in a variable

**---**

**- hosts: webserver**

**become: true**

**become\_user: root**

**gather\_facts: false**

**tasks:**

**- name: display bash version**

**shell: "bash --version"**

**register: bash\_version**

**- name: print the variable value**

**debug:**

**var: bash\_version**

this will return value of variable as maps==> keys & values

this will return lots of data

lets add some python scripting to filter the data and fetch only stdout value

fetch the value of stdout key in the variable and fetch the value at 0th index

here bash\_version 🡺 variable

std.out 🡺 key

[0]🡺 value index

**---**

**- hosts: webserver**

**become: true**

**become\_user: root**

**gather\_facts: false**

**tasks:**

**- name: display bash version**

**shell: "bash --version"**

**register: bash\_version**

**- name: print the variable value**

**debug:**

**var: bash\_version.stdout.split("\n")[0]**

**:wq!**

Use of Fact variables instead of using some linux commads to find os release, distribution etc

Show this

---

- hosts: localhost:172.31.21.30

tasks:

- name: find os distribution

shell: "cat /etc/os-release"

register: os\_dist

- name: find the os name

shell: "uname"

register: os\_name

- name: display the values

debug:

msg:

- "the os distribution is {{os\_dist}}"

- "the os name is {{os\_name}}"

Execute this:

---

- hosts: localhost:172.31.21.30

tasks:

- debug:

msg:

- "The os distribution is: {{ansible\_distribution}}"

- "THe os name is: {{ansible\_system}}"

- "The os family is: {{ansible\_os\_family}}"

- "THe mount points are :{{ansible\_mounts}}"

Read the variable value at runtime by giving vars\_prompt option

to install packages

---

- hosts: localhost

become: true

become\_user: root

gather\_facts: false

vars\_prompt:

- name: pkg\_name1

prompt: Enter your package Name

private: false

- name: service\_name1

prompt: Enter the service to start

private: false

tasks:

- name: Install a package

yum: name={{pkg\_name1}} state=present

- name: Start a service

service: name={{service\_name1}} state=started

[**List variables**](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#id22)[**ℑ**](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#list-variables)

A list variable combines a variable name with multiple values. The multiple values can be stored as an itemized list or in square brackets [], separated with commas.

[**Defining variables as lists**](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#id23)[**ℑ**](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#defining-variables-as-lists)

You can define variables with multiple values using YAML lists. For example:

**region:**

**-** northeast

**-** southeast

**-** Midwest

[**Referencing list variables**](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#id24)[**ℑ**](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#referencing-list-variables)

When you use variables defined as a list (also called an array), you can use individual, specific fields from that list. The first item in a list is item 0, the second item is item 1. For example:

**region:** "{{ **region[**1**]** }}"

The value of this expression would be “northeast”.

## [Dictionary variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#id25)[ℑ](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#dictionary-variables)

A dictionary stores the data in key-value pairs. Usually, dictionaries are used to store related data, such as the information contained in an ID or a user profile.

### [Defining variables as key:value dictionaries](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#id26)[ℑ](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#defining-variables-as-key-value-dictionaries)

You can define more complex variables using YAML dictionaries. A YAML dictionary maps keys to values. For example:

**foo:**

**field1:** one

**field2:** two

### [Referencing key:value dictionary variables](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#id27)[ℑ](https://docs.ansible.com/ansible/latest/user_guide/playbooks_variables.html#referencing-key-value-dictionary-variables)

When you use variables defined as a key:value dictionary (also called a hash), you can use individual, specific fields from that dictionary using either bracket notation or dot notation:

foo['field1'] -🡪one

foo.field1

Example:

Defining variables in a separate file and calling them in a playbook

Read variable values from a yaml files

create a file:

sudo vim myvariables.yml

pkgs:

- docker

- git

- wget

pkgs2:

linux: httpd

ubuntu: apache2

sudo vim variablefilesDemo.yml

---

- hosts: localhost

become: true

become\_user: root

gather\_facts: false

vars\_files:

- myvariables.yml

tasks:

- name: read package value from list

debug: msg="the package is {{pkgs[0]}}”

- name: install package

yum: name={{pkgs[0]}} state=present

- name: read package value as maps

debug: msg="the package is {{pkgs2.linux}}”

- name: Install package

yum: name={{pkgs2.linux}} state=present

Working with FACT\_VARIABLES

Working with fact variables in ansible

if you will execute this adhoc command, it will give all the facts releated to your ansible host

$ ansible all -m setup

$ ansible all -m setup | grep ansible

it will show all the variables that can used in ansible playbooks

look for variable : ansible\_system & ansible\_distribution and use it in a playbook

Use the variables in playbook

---

- hosts: all

tasks:

- name: print fact variable values

debug:

msg:

- " the os distribution is : {{ansible\_distribution}}"

- " the os name is : {{ansible\_system}}"

- " the os family is : {{ansible\_os\_family}}"

It is very simple to work if we use ansible fact variables

else we will have to write big playbook like

Conditional Statements:

when

failed\_when

changed\_when

when is conditional statement in ansible, it is like if condition

---

- name: Simple playbook to install httpd

hosts: all

gather\_facts: true

tasks:

- name: Installing httpd using yum

yum: name=httpd state=present

when: ansible\_distribution != "Ubuntu"

- name: install apache2 using apt

apt: name=apache2 state=present

when: ansible\_distribution == "Ubuntu"

Loops:

\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*

playbook with no loops:

---

- hosts: localhost

gather\_facts: false

become: yes

tasks:

- yum:

name: gettext-devel

state: present

- yum:

name: openssl-devel

state: present

- yum:

name: perl-CPAN

state: present

- yum:

name: perl-devel

state: present

- yum:

name: zlib-devel

state: present

playbook with loop:

---

- hosts: localhost

gather\_facts: false

become: yes

tasks:

- yum:

name: "{{item}}"

state: absent

loop:

- gettext-devel

- openssl-devel

- perl-CPAN

- perl-devel

- zlib-devel

Combination of variables & loops

---

- name: Creating users and creating files/dir

hosts: localhost:172.31.21.30

become: yes

tasks:

- name: User creation

user:

name: "{{item.a}}"

home: "{{item.b}}"

loop:

- {a: Bob,b: /tmp/bob}

- {a: Ravi,b: /tmp/Ravi}

- name: Creating files/dirs in user home dir

file:

name: "{{item.d}}"

state: "{{item.e}}"

loop:

- {d: /tmp/bob/file1,e: touch}

- {d: /tmp/Ravi/dir1,e: directory}

Error Handling : ignore the errored out tasks

When executing a playbook, if one task fails then playbook will stop execution and will not go further

Lets consider this example:

- hosts: localhost

tasks:

- name: execute command

command: ls /home

register: var\_myhome

- name: print the variable

debug: var=var\_myhome

- name: execute command

command: ls /tmp

register: var\_mytmp

- name: print the value

debug: var=var\_mytmp

Execute.. it will work fine

Now chnage task1 and give a wrong directory name

- hosts: localhost

tasks:

- name: execute command

command: ls /home12cvfg

register: var\_myhome

- name: print the variable

debug: var=var\_myhome

- name: execute command

command: ls /tmp

register: var\_mytmp

- name: print the value

debug: var=var\_mytmp

Execute playbook, task 1 will fail and playbook will stop, no task will be executed

here, there is no dependency between the tasks, so even if one task fails.. others should be executed

ansible can ignore the task that has been failed and go ahead and execute the rest of the tasks

- hosts: localhost

tasks:

- name: execute command

command: ls /home12cvfg

register: var\_myhome

ignore\_errors: yes

- name: print the variable

debug: var=var\_myhome

- name: execute command

command: ls /tmp

register: var\_mytmp

- name: print the value

debug: var=var\_mytmp

Blocks in playbooks:

We may not know which tasks may fail, so we may not know for which task should we write ignore\_errors=yes or not

So we can create a block in playbook.

Block can be used to apply a condition or apply become=yes or ignore\_errors=yes for multipel tasks

- hosts: localhost

tasks:

- block:

- name: execute command

command: ls /home1234e

register: var\_myhome

- name: execute command

command: ls /tmp

register: var\_mytmp

ignore\_errors: yes

- name: print the value

debug: var=var\_myhome

- name: print value2

debug: var=var\_mytmp

Block with when condition:

---

- hosts: web\_servers

gather\_facts: true

tasks:

- block:

- name: Installing htttpd for RedHat os family

yum:

name: httpd

state: present

- name: starting httpd for RedHat os family

service:

name: httpd

state: started

when: ansible\_os\_family=="RedHat"

become: yes

- block:

- name: Installing apache2 for Debian os family

apt:

name: apache2

state: present

- name: starting apache2 for Debian os family

service:

name: apache2

state: started

when: ansible\_os\_family=="Debian"

become: yes

- debug:

msg: "Succesfully completed all tasks"

Just like try & catch block in java.. we have block and rescue option & always options

here if task in block fails.. it will be handled by rescue block(it will execute)

and task under always block will always get executed

---

- hosts: localhost

gather\_facts: false

tasks:

- block:

- name: Finding files in /home/ansadmin/tomcat8

command: "ls /home/ansadmin/tomcat8"

register: tomcat8\_out

rescue:

- debug:

msg: "The given path: /home/ansadmin/tomcat8 is not a validpath"

always:

- debug:

msg: "THis will always executes"

|  |
| --- |
| Template Module: |
|  |  |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | While using template moduel |
|  |  |
|  | we will have a folder with name as templates in our current directory and jinja2 file in it : |
|  |  |
|  | ./templates/my\_app.conf.j2 |
|  |  |
|  | local\_ip = {{ ansible\_default\_ipv4["address"] }} |
|  | local\_user = {{ ansible\_user }} |
|  |  |
|  |  |
|  |  |
|  | Running the following task in playbook.yml: |
|  |  |
|  | - name: install my\_app configuration file from template |
|  | template: |
|  | src: my\_app.conf.j2 |
|  | dest: tmp/my\_app.conf |
|  |  |
|  | Produces the following on my Ubuntu 18.04 test host: |
|  |  |
|  | local\_ip = 10.1.11.72 |
|  | local\_user = ubuntu |
|  |  |
|  | And the following on Centos host: |
|  |  |
|  | local\_ip = 10.1.11.62 |
|  | local\_user = centos |
|  |  |
|  |  |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  |  |
|  | example 1: |
|  |  |
|  | cd /etc/ansible |
|  |  |
|  | sudo mkdir templates |
|  |  |
|  | cd templates  Welcome to {{ ansible\_facts.hostname }} |
|  | (IP Address: {{ ansible\_facts.default\_ipv4.address }}) |
|  |  |
|  | Access is restricted; if you are not authorized to use it |
|  | please logout from this system |
|  |  |
|  | If you have any issues, please contact {{ system\_admin\_email }}. |
|  | Phone: {{ system\_admin\_phone | default('1800 1111 2222') }} |
|  |  |
|  | ------------------------------------- |
|  | This message is configured by Ansible |
|  | ------------------------------------- |
|  |  |
|  |  |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  |  |
|  | playbook |
|  |  |
|  | sudo vim templatedemo.yml |
|  |  |
|  | - name: Using Jinja2 |
|  | hosts: 172.31.21.30 |
|  | become: yes |
|  | vars: |
|  | system\_admin\_email: admin@lab.local |
|  | #system\_admin\_phone: '1800 0000 0000' |
|  | tasks: |
|  | - name: Deploy motd |
|  | template: |
|  | dest: /etc/motd |
|  | src: templates/motd.j2 |
|  |  |
|  |  |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

Example2:

\*\*\*\*\*\*\*

template module:

- hosts: localhost

become: yes

become\_user: root

vars:

page\_title: "placeholder"

page\_description: "This is my placeholder page example"

tasks:

- name: install EPEL repo

yum: name=epel-release state=present

- name: install nginx

yum: name=httpd state=present

- name: start https

service: name=httpd state=started

- name: apply page template

ansible.builtin.template:

src: templates/placeholder.html.j2

dest: /var/www/html/index.html

- name: restart https

service: name=httpd state=restarted

Create a directory as templates in etc ansible

mkdir templates

cd templates

create jinja2 template

vim placeholder.html.j2

<html>

<head>

<title>{{ page\_title}} </title>

</head>

<body>

<h1> {{page\_title}} </h1>

<p> {{page\_description}} </p>

</body>

</html>

:wq!

Chnages will be applied to all the hosts

Child playbook and calling one playbook in other

ChildPlaybook

---

- name: Install firewall

apt:

name: firewalld

state: present

update\_cache: yes

Parentplaybook

---

- name: Call child playbook

hosts: all

tasks:

- name: Install firewlld using child playbook

include:

playbook16.yml

---

- name: Setup of docker for ansible

hosts: localhost

become: yes

tasks:

- name: Install python-pip

yum:

name: python-pip

state: absent

- name: Install docker-py,download docker,install it

shell: "{{item}}"

with\_items:

- pip install docker-py

- curl -fsSL https://get.docker.com -o get-docker.sh

- sh get-docker.sh

Docker containers:

---

- name: Working on tomcat

hosts: localhost

become: yes

tasks:

- name: Install docker

yum:

name: docker

state: present

- name: start docker

service:

name: docker

state: started

- name: Start tomcat as a container

docker\_container:

image: nginx

name: mynginx

- name: Stop the tomcat container

docker\_container:

name: mynginx

state: stopped

- name: Delete the container

docker\_container:

name: mynginx

state: absent