



## ❖ **MANAGING VARIABLES:**

- Ansible supports variables that can be used to store values that can be reused throughout files in an entire Ansible project.
- Variables provides a convenient way to manage dynamic values for a given environment in your ansible project.
- Some examples of values that variables might contain include.
  - Users to create
  - Package to install
  - Services to restart
  - files to remove
- Variables have names which consist of a string that must start with a letter and can only contain letters, numbers, and underscores.

## **DEFINING VARIABLES:**

- Variables can be defined in a variety of places in an Ansible project. However, this can be simplified to three basic scope levels:

**GLOBAL SCOPE:** Variables set from the command line or Ansible Configuration

**PLAY SCOPE:** Variables set in the play and related structures.

**HOST SCOPE:** Variables set on host groups and individual hosts by the inventory, fact gathering, or registered tasks

## ➤ **VARIABLES IN PLAYBOOKS:**

- When writing playbooks, administrators can use their own variables and call them in a task.
- For example, a variable web\_package can be defined with a value of httpd and called by the yum module in order to install the httpd package

## INSTALLING WEB SERVER USING VARIABLES:

- hosts: webservers

become: true

become\_user: root

**vars:**

web\_pkg: httpd

firewall\_pkg: firewalld

perl\_pkg: perl

rule: http

tasks:

**- name: Mount the OS media Drive**

command: mount /dev/sr0 /mnt

**- name: Copy the local repo file**

copy:

src: /home/raju/ansible/server.repo

dest: /etc/yum.repos.d

**- name: Install Package**

yum:

name:

- "{{ web\_pkg }}"

- "{{ firewall\_pkg }}"

- "{{ perl\_pkg }}"

state: latest

**- name: Start & Enable Service httpd**

service:

name: "{{ web\_pkg }}"

enabled: true

state: started

**- name: Create web content**

copy:

content: "Welcome To Ansible"

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

**- name: Open the port for {{ rule }}**

firewalld:

service: "{{ rule }}"

permanent: true

immediate: yes

state: enabled

...

\$mkdir ansible

\$cd ansible

\$touch server.repo

\$ansible-playbook --syntax-check var.yml

\$ansible-playbook var.yml -K

## ➤ HOST VARIABLES & GROUP VARIABLES:

- Inventory variables that apply directly to hosts fall into broad categories that apply to a specific host, and group variables that apply to all hosts in a host group or in a group of hosts.
- Host variables take precedence over group variables, but variables defined by a playbook take precedence over both.
- This is a host variable, `ansible_user`, being defined for the host `server.example.com`

```
[servers]
```

```
server.example.com ansible_user=jai
```

```
$vi /etc/ansible/hosts
```

```
[servers1]
```

```
agent1 ansible_user=jai
```

```
agent2 ansible_user=ramu
```

```
agent3
```

```
[servers2]
```

```
agent4
```

```
agent5
```

```
agent6
```

```
[servers:children]
```

```
servers1
```

```
servers2
```

```
[servers:vars]
```

```
ansible_user=raju
```

```
ansible_hosts=xyz
```

## EXAMPLE OF PLAYBOOK:

```
$vi var2.yml
```

```
- hosts: webservers
```

```
  become: true
```

```
  become_user: root
```

```
  vars:
```

```
    remote_dir: /etc/devops/ansible
```

```
    ans_file: sample
```

```
  tasks:
```

### - name: Ceate a Remote Directory

```
    file:
```

```
      state: directory
```

```
      recurse: yes
```

```
      path: "{{ remote_dir }}"
```

### - name: Copy a file

```
    copy:
```

```
      src: "{{ ans_file }}"
```

```
      dest: "{{ remote_dir }}"
```

```
$touch sample
```

```
$ansible-playbook --syntax-check var2.yml
```

```
$ansible-playbook var2.yml -K --step
```

**NOTE:** \$ansible-playbook var2.yml -K -e "ans\_file=java"