Ansible Variables

What is Variable?

- A **variable** is a datatype whose value cannot be fixed. It can be change based on other parameters.
- **Variables** can represent numeric values, characters, character strings, or memory addresses.
- **Variables** play an important role in computer programming because they enable programmers to write flexible programs. Rather than entering data directly into a program, a programmer can use **variables** to represent the data.

What is Ansible Variable?

- Ansible supports variables that can be used to store values that can then be reused throughout files in an Ansible project.
- This can simplify the creation and maintenance of a project and reduce the number of errors.
- Variables provide a convenient way to manage dynamic values for a given environment in your Ansible project.
- Variables can store the return value of executed commands.
- They are typically used for configuration values and various parameters.
- Ansible provides a number of predefined variables**.**
- Examples of values that variables might contain include: User Creation, Installing Packages, starting or restarting services, etc.

Naming Structure:

- Variables name should only contain letters, number, and underscores.
- Variables should always start with a letter.

Examples:

Invalid Variable Name	Valid Variables Name
web server	web_server
1stServer 1.Server	server_1 server1
remote.file remote$file.1	remote_file remote_file_1

Ansible Variables

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.
        - This is like an environment variable.
    - *Play scope*:
        - Variables set in the play and related structures.
        - It is associate with play within a playbook.
    - *Host scope*:
        - Variables set on host groups and individual hosts by the inventory, fact gathering, or registered tasks.
        - It going to include variable that are directly associate with your hosts or group of hosts.
- If the same variable name is defined at more than one level, the level with the highest precedence wins.
- A narrow scope takes precedence over a wider scope: such as variables defined by the inventory are overridden by variables defined by the playbook, which are overridden by variables defined on the command line.

Declare Variable in Playbook

- Variables play an important role in Ansible Playbooks because they ease the management of variable data in a playbook.
- When writing playbooks, you can define your own variables and then invoke those values in a task.
- Once your variables have been declared, you can use the variables in tasks. Variables are referenced by placing the variable name in double curly braces ({{ }}).
- Ansible substitutes the variable with its value when the task is executed.
- Playbook variables can be defined in multiple ways.
    - One common method is to place a variable in a **vars** block at the beginning of a playbook.

```
---
 - name:
   hosts: all
   vars:
     x: 45                    # this is your varibale
     y: 50                    # this is your varibale
   tasks:
    - debug:
       msg:
           - " The value of x is {{ x }}
           - " The value of y is {{ y }}
```

Ansible Variables

- If you want to read your variable value while running the playbook.

```
---
 - name:
   hosts: all
   vars:
     x: 45                        # this is your variable
     y: 50                        # this is your variable
   vars_prompt:
     name: z                      # this is your variable
     prompt: Enter z value:       # provide value at run time
     private: false               # to display the value while providing z value
   tasks:
    - debug:
       msg:
          - "The value of x is {{ x }} "
          - "The value of y is {{ y }} "
          - "The value of y is {{ y }} "
```

- Another ways to define playbook variables in external files.

```
# vim variable.yml
---
  x: 45             # this is your variable
  y: 50             # this is your variable
  friends:
     - anaya
     - priya
     - mahi

wq:
```

```
---
 - name:
   hosts: all
   vars_files:
      -variable.yml     # variable file name
   - debug:
      msg:
         -"The value of x: {{x}}
         -"The value of y: {{y}}
         -"My Friends list: {{friends}}
```

Hosts Variable and Groups Variable

- Inventory variables that apply directly to hosts fall into two broad categories:
- *host variables* apply to a specific host,
- *group variables* apply to all hosts in a host group or in a group of host groups.
- Host variables take precedence over group variables, but variables defined by a playbook take precedence over both.
- The preferred approach to defining variables for hosts and host groups is to create two directories, in the same working directory as the inventory file or directory. These directories contain files defining group variables and host variables, respectively.
- The recommended practice is to define inventory variables using **host_vars** and **group_vars** directories, and not to define them directly in the inventory files.

Ansible Variables

Command Line Arguments

- Passing the variable while executing the playbook.
- Command line arguments are useful to pass variable from command instead for declared in playbook.
- Variables set on the command line are called *extra variables*.
- Inventory variables are overridden by variables set in a playbook, but both kinds of variables may be overridden through arguments passed to the **ansible** or **ansible-playbook** commands on the command line.
- $ ansible-playbook playboo_name - -extra-var “variable_value”
- $ ansible-playbook playbook_name -e “variable_value”

```
---
 - name: variable will pass in command
   hosts: all
   tasks:
    - debug:
       msg:
           - " The value of x is {{ x }}
           - " The value of y is {{ y }}
```

- $ ansible-playbook variable_cmd.yml -e “x=22 y=44”
- $ ansible-playbook variable_cmd.yml - -extra-var “x=22 y=44”
- $ ansible-playbook variable_cmd.yml -e “@variable_file_name”

Lets understand the advantage of Command Line Arguments with playbook

```
---
 - name: variable will pass in command
   hosts: all
   tasks:
   become: yes
   tasks:
     - name: Installing and Uninstalling {{ pkg }}
       yum:
           name: “{{ pkg_name }}”
           state: “{{ pkg_state }}”
```

- $ ansible-playbook variable_cmd.yml -e “pkg_name=httpd pkg_state=present”

Register Variable:

- Use the register keyword to store the results of running a command as a variable.
- Administrators can use the register statement to capture the output of a command.
- The output is saved into a temporary variable that can be used later in the playbook for either debugging purposes or to achieve something else, such as a particular configuration based on a command's output.

Ansible Variables

```
---
 - name: Registerd output
   hosts: 172.16.79.134
   gather_facts: false
   tasks:
    - command: "ls /home/ansadmin/playbook"
      register: ls_out
    - debug: var=ls_out
```

- Deploying Web server through playbook using variable

```
---
 - name: Deploy and start  HTTPD service
   hosts: active
   gather_facts: false
   become: true
   vars:                                        # Declared variable
    web_pkg: httpd
    firewall_pkg: firewalld
    web_service: httpd
    firewall_service: firewalld
    rule: http
   tasks:                                                                    # Running tasks
    - name: Required packages are installed and up to date                   # Tasks-1
      yum:
       name:
        - "{{ web_pkg }}"
        - "{{ firewall_pkg }}"
       state: latest
    - name: The {{ firewall_service }} service is started and enabled        # Task-2
      service:
       name: "{{ firewall_service }}"
       enabled: true
       state: started
    - name: The {{ web_service }} service is started and enabled             # Task-3
      service:
        name: "{{ web_service }}"
        enabled: true
        state: started
    - name: Web content is in place                                          # Task-3
      copy:
       content: "Example web content\n"
       dest: /var/www/html/index.html
    - name: The firewall port for {{ rule }} is open                         # Task-4
      firewalld:
      service: "{{ rule }}"
        permanent: true
        immediate: true
        state: enabled
```