Ansible Variables

Variables are used to store values in programs and as the name suggests the values can be changed throughout the program. These values are essential to deciding the code flow. Ansible variables help to determine how the tasks execute on different systems based on the values assigned to these variables.

Ansible variable names should be letters, numbers, underscores and they should always start with a letter.

In the playbook, we can give a variable in ‘variable_name: variable_value‘ format. And we can use the variable_name inside double braces anywhere in the playbook. In the following task, I am declaring a variable hellowith value world and is referenced in one of the tasks. The task will output the value world.

```
- hosts: all
  vars:
    hello: world
  tasks:
  - name: Ansible  Variable Example
    debug:
      msg: "{{ hello }}"
```

Ansible List (Array) variables

We can also have an array or list variables. The following task shows how to declare an array variable in Ansible and how to use the values. The hello contains 9 values, and each one can be accessed using the index numbers(starting from zer0). The following task will output ‘South America‘

```
- hosts: all
  vars:
    hello:
```

```
   - World
   - Asia
   - South America
   - North America
   - Artic
   - Antartic
   - Oceania
   - Europe
   - Africa
 tasks:
 - name: Ansible List variable Example
  debug:
   msg: "{{ hello[2] }}"
```

You can also give the variable values in the following format.

```
vars:
 hello: [Asia, Americas, Artic, Antartic ,Oceania,Europe,Africa]
```

If you need to access all the variables, you can use the with_items structure to loop through all the values.

```
- hosts: all
 vars:
  hello: [Asia, Americas, Artic, Antartic ,Oceania,Europe,Africa]
 tasks:
 - name: Ansible array variables example
  debug:
   msg: "{{ item }}"
  with_items:
   - "{{ hello }}"
```

Ansible Multiline Variables

It is possible to give variable values in multiple lines using YAML block operators. There are two options for this, depending on whether you need the newline characters to be retained or not.

You can use the ‘|’ operator if you want the new line characters to be included. If you do not want the new line character, and you just want to use it for easy understanding during editing then use ‘>’ operator.

```
- hosts: all
  vars:
    include_newlines_example: |
            The new line charaters
            will appear

    ignore_newlines_example2: >
            The new line character will
            be removed. Useful when editing
            lines
  tasks:
  - name: Ansible varible multiple line Example
    debug:
      msg: "{{ include_newlines_example }}"

  - name: Ansible variables multiline Example
    debug:
      msg: "{{ ignore_newlines_example2 }}"
```