Lab 12: Writing Loops and Conditional Tasks

Introduction:

Task Iteration with Loops

Using loops saves administrators from the need to write multiple tasks that use the same module. For example, instead of writing five tasks to ensure five users exist, you can write one task that iterates over a list of five users to ensure they all exist. Ansible supports iterating a task over a set of items using the loop keyword. You can configure loops to repeat a task using each item in a list, the contents of each of the files in a list, a generated sequence of numbers, or using more complicated structures. This section covers simple loops that iterate over a list of items. Consult the documentation for more advanced looping scenarios.

Objectives:

- **Adding new user to Target with loops**

1. Adding new user to Target with loops

1.1 Let's create Playbook **users.yml** that adds a new user on the target system using the user module as shown.

```
1  ---
2  - hosts: eoc-node1
3    become: yes
4    tasks:
5      - name: Create new user john
6        user:
7          name: john
8          state: present
```

```
 9      - name: craete new user mike
10       user:
11         name: mike
12         state: present
```

```
13      - name: create new user andrew
14        user:
15          name: andrew
16          state: present
```

1.2 Let's view the manifest.

```
# cat -n users.yml
```

Output:

```
[admin@eoc-controller ~]$ cat -n users.yml
     1  ---
     2  - hosts: eoc-node1
     3    become: yes
     4    tasks:
     5      - name: Create new user john
     6        user:
     7          name: john
     8          state: present
     9      - name: craete new user mike
    10        user:
    11          name: mike
    12          state: present
    13      - name: create new user andrew
    14        user:
    15          name: andrew
    16          state: present
```

1.3 Let's verify the syntax is correct of **users.yml** by running ansible-playbook --syntax-check command.

```
# ansible-playbook --syntax-check users.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook --syntax-check users.yml

playbook: users.yml
```

1.4 Let's run the play book **users.yml** by executing below command

```
# ansible-playbook users.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook  users.yml

PLAY [eoc-node1] ***************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [eoc-node1]

TASK [Create new user john] ****************************************************************************************
changed: [eoc-node1]

TASK [craete new user mike] ****************************************************************************************
changed: [eoc-node1]

TASK [create new user andrew] **************************************************************************************
changed: [eoc-node1]

PLAY RECAP *********************************************************************************************************
eoc-node1                  : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

1.5 Let's verify the loop task on the **eoc-node1** host.

```
# ssh eoc-node1 getent passwd | cut -d: -f1 | tail -5
```

Output:

```
[admin@eoc-controller ~]$ ssh eoc-node1 getent passwd | cut -d: -f1 | tail -5
admin
apache
john
mike
andrew
```

2. Ansible Approach

2.1 To make things easier, the same playbook can be written using the loop directive. The loop directive executes the same task multiple times. It stores the value of each item in a variable called item. So, instead of specifying the names of the users to be added, simply specify a variable called item enclosed between double curly braces as shown.

2.2 Let's create a Playbook manifest with the name **loop_users.yml** that adds a new user on the target system using the user module.

```
---
- hosts: eoc-node1
  become: yes
```

2.3 Let's create a task which install multiple users by using loops

```
 5      - name: create new users
 6        user:
 7          name: '{{ item }}'
 8          state: present
 9        loop:
10          - james
11          - mary
12          - robert
13          - irwin
14          - joe
```

2.4 Let's view the yaml manifest.

```
# cat -n loop_users.yml
```

Output:

```
[admin@eoc-controller ~]$cat -n loop_users.yml
     1  ---
     2  - hosts: eoc-node1
     3    become: yes
     4    tasks:
     5      - name: create new users
     6        user:
     7          name: '{{ item }}'
     8          state: present
     9        loop:
    10          - james
    11          - mary
    12          - robert
    13          - irwin
    14          - joe
```

2.5 Let's verify the syntax of **loops_users.yml** playbook.

```
# ansible-playbook --syntax-check loop_users.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook --syntax-check loop_users.yml

playbook: loop_users.yml
```

2.6 Let's run the play book **loop_users.yml** .

```
# ansible-playbook -v loop_users.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook -v loop_users.yml
Using /home/admin/.ansible.cfg as config file

PLAY [eoc-node1] ***************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [eoc-node1]

TASK [create new users] ********************************************************************************************
changed: [eoc-node1] => (item=james) => {"ansible_loop_var": "item", "changed": true, "comment": "", "create_hom
s", "shell": "/bin/bash", "state": "present", "system": false, "uid": 1005}
changed: [eoc-node1] => (item=mary) => {"ansible_loop_var": "item", "changed": true, "comment": "", "create_home
"shell": "/bin/bash", "state": "present", "system": false, "uid": 1006}
changed: [eoc-node1] => (item=robert) => {"ansible_loop_var": "item", "changed": true, "comment": "", "create_ho
obert", "shell": "/bin/bash", "state": "present", "system": false, "uid": 1007}
changed: [eoc-node1] => (item=irwin) => {"ansible_loop_var": "item", "changed": true, "comment": "", "create_hom
n", "shell": "/bin/bash", "state": "present", "system": false, "uid": 1008}
changed: [eoc-node1] => (item=joe) => {"ansible_loop_var": "item", "changed": true, "comment": "", "create_home"
ll": "/bin/bash", "state": "present", "system": false, "uid": 1009}

PLAY RECAP *********************************************************************************************************
eoc-node1                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored
```

2.7 Let's verify the loop task on the **eoc-node1** host.

```
# ssh eoc-node1 getent passwd | cut -d: -f1 | tail -5
```

Output:

```
[admin@eoc-controller ~]$ssh eoc-node1 getent passwd | cut -d: -f1 | tail -5
james
mary
robert
irwin
joe
```