Ansible Loops

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Session - 14 Agenda:

1. Ansible Loops.

In the course of executing tasks in Ansible playbook, you might come across some tasks that are repetitive in nature. These are tasks that require you to create multiple plays, something which can be quite tedious. As with any programming language, loops in Ansible provide an easier way of executing repetitive tasks using fewer lines of code in a playbook. When creating loops, Ansible provides these two directives: loop and with_* keyword.

The loop keyword was recently added to Ansible 2.5. The loop keyword is usually used to create simple and standard loops that iterate through several items.

Ansible loop allows the repetition of tasks as many times as possible. Tasks such as creating multiple users, changing the ownership on files, installing multiple packages, etc.

Problems in Ansible Without Loops:

Why do we require loops in Ansible? Let's understand this with an example.

Let's us suppose we have n number of user accounts which we need to add in the managed nodes through ansible. There are many of performing such task in ansible.

1. Create separate tasks for each user account in the ansible playbook.

$ vim users.yml

```
---
- name: User account creation playbook
  hosts: all
  become: true
  tasks:
     - name: Adding user vikas
       user:
          name: vikas
          state: present
     - name: Adding user amit
       user:
          name: amit
          state: present
     - name: Adding user greta
       user:
          name: greta
          state: present
...
```

Let's check the playbook for errors.

$ ansible-playbook users.yml --syntax-check

Ansible Loops

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Let's check the playbook execution with dry run.

$ ansible-playbook users.yml -C

Problem here is we have to create separate tasks for each user account which we want to create on the managed nodes. In case if there are 100s or 1000s of such user accounts to be created, we have to write separate task for each such user account to be created on the managed nodes. In will be very hectic, lengthy and time-consuming process which would require huge human effort. And there are chances of errors due to complexities in the playbook having large number of tasks.

2. Using Lists in the user module under the same task:

Some people here may suggest using lists option in the user module but that doesn't work. We can easily use lists option with some modules like yum but not with user module because user module has been developed in such a way that it doesn't supports multiple values in key value pair by default. This limitation here is due to useradd command in Linux, which doesn't support multiple usernames. Name type in the user module will be string or list is decided by the developer who writes its code in python for ansible on the basis of it's working in Linux. If he will mention the type as list here for the user module then also it will not work because of the limitation of the useradd command which is actually responsible at backend for adding the user accounts in the Linux machine (OS will not support it). Let's test this method as well.

$ vim users_lists.yml

```
---
- name: User account creation playbook
  hosts: all
  become: true
  tasks:
     - name: Adding user accounts
       user:
          name:
               - vikas
               - amit
               - greta
               - tarun
          state: present
...
```

Let's check the playbook for errors.

$ ansible-playbook users_lists.yml --syntax-check

Let's check the playbook execution.

$ ansible-playbook users_lists.yml

Ansible Loops

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

So, this method is not supported in Ansible.

3. Using Ansible Loops:

This task can easily be performed using Ansible Loops. Ansible loops can be categorized into following categories:

a) Simple/Standard Loop:

It is similar to using for loop in Linux. If we want to perform any simple operation or task n number of times. It is suitable for simple tasks having multiple values of the same key ansible.

e.g. Suppose if we want to create n number of user accounts having the same password in the managed nodes using standard loop. Such tasks where only one list of items is used in the module, there we prefer simple/standard loop.

Usernames:

Vikas
Amit
Greta
Tarun

b) Hash/Mapping Loops:

If we want to perform any complex operation or task n number of times which further has multiple lists. It is suitable for complex tasks having multiple values of the same key ansible.

e.g. Suppose if we want to create multiple user accounts having different passwords in the managed nodes. Such tasks where multiple lists of items are needed in the same module, there we prefer mapping loops.

Usernames:	**Passwords:**
Vikas	p@$$word
Amit	redhat
Greta	greta

c) Nested Loops:

If we want to perform any complex operation or task n number of times which further requires using loops for this task.

e.g. Let's suppose if we want to add n number of user accounts in the managed nodes with different passwords and each user should be the members of the few groups.

Usernames:	**Passwords:**	**Groups:**
Vikas	p@$$word	Admin Nehra Classes
Amit	redhat	Admin Nehra Classes

Ansible Loops

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

Let's create a playbook with simple loop for the above task to create n number of user accounts in the managed nodes.

$ vim standard_loop.yml

```
---
- name: Ansible simple loop playbook
  hosts: all
  become: true
  tasks:
        - name: Adding user accounts
          user:
                name: "{{ item }}"
                state: present
          with_items:
                - vikas
                - amit
                - greta
                - tarun
                - deepika
...
```

Let's check the playbook for errors.

$ ansible-playbook standard_loop.yml --syntax-check

Let's check the playbook execution with dry run.

$ ansible-playbook standard_loop.yml -C

Execute the playbook to check it's working.

$ ansible-playbook standard_loop.yml

Verify the task using ansible ad-hoc command.

$ ansible all -m command -a 'cat /etc/passwd'

Loops are sometimes not required for some modules which supports list options like yum module but if we want, we can use loops with such modules in the latest versions of Ansible (this feature was not supported earlier for such modules).

Let's verify the same with the help of a playbook to install multiple packages on the managed nodes.

$ vim yum_loop.yml

Ansible Loops

By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years

```
---
- name: Ansible loop playbook for yum module
  hosts: node1
  become: true
  tasks:
        - name: Using yum with loop
          yum:
             name: "{{ item }}"
             state: present
          with_items:
             - zsh
             - httpd
             - vsftpd
             - ftp
             - wget
...
```

Let's check the playbook for errors.

$ ansible-playbook yum_loop.yml --syntax-check

Let's check the playbook execution with dry run.

$ ansible-playbook yum_loop.yml -C

Let's verify the packages first with the ansible ad-hoc commands.

$ ansible node1 -m command -a 'rpm -qi zsh'

$ ansible node1 -m command -a 'rpm -qi httpd'

$ ansible node1 -m command -a 'yum list installed ftp'

Now, execute the playbook.

$ ansible-playbook yum_loop.yml

Let's verify the task execution with the ansible ad-hoc commands.

$ ansible node1 -m command -a 'rpm -qi zsh'

$ ansible node1 -m command -a 'rpm -qi httpd'

$ ansible node1 -m command -a 'yum list installed ftp'

Thanks