Ansible Loops

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

Session - 15 Agenda:
1. Ansible Loops:

Loops are sets of commands or instructions that are set to repeat a certain number of times as per user requirements. Loops allow for better control flow in your scripts and remove overall redundancy from them.

Keywords for Loops in Ansible:
Ansible uses the following keywords for its loop structures:
“loop”
“with_<lookup term>”
“until”
As the name suggests, “loop” is the go-to command for implementing loops in blocks of code.

Similar to “loop”, you have “with_<lookup>”. This command requires you to provide it with a lookup plugin. It is recommended to use with_* rather than loop when lookup plugins are involved.

“Until” allows you to keep on executing a task till the required condition is met. It is the closest to the “while” condition in the control flow.

(a) Simple/Standard Loops:
Let’s create a playbook with simple/standard loop 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 'tail -5 /etc/passwd'
```

Let’s suppose we need to create multiple user accounts which should be members of a secondary group (wheel) in the managed nodes. This time we will use loop keyword instead of with_* in the playbook.

```
$ vim standard_loop2.yml
---
- name: Ansible standard loop playbook to add user accounts with admin access
  hosts: all
  become: true
  tasks:
        - name: Adding user accounts with admin access
          user:
                  name: "{{ item }}"
                  state: present
                  group: wheel
          loop:
                   - ashish
                   - buddy
                   - harish
                   - sawan
                   - ankit
...
```

Let’s check the playbook for errors.

```
$ ansible-playbook standard_loop2.yml --syntax-check
```

Let’s check the playbook execution with dry run.

```
$ ansible-playbook standard_loop2.yml -C
```

Execute the playbook to check it’s working.

```
$ ansible-playbook standard_loop2.yml
```

Verify the task using ansible ad-hoc command.

```
$ ansible all -m command -a 'tail -5 /etc/passwd'
```

To see the list of available plugins.

```
$ ansible-doc -t lookup -l
```

To see specific documents and examples.

```
$ ansible-doc -t lookup <plugin name>
```

Calling the variable list for the loop from another file in ansible playbook:

Let’s create a file first having the name list for the loop which will be called in the ansible playbook.

```
$ vim users.yml
users:
        - ashu
        - codebuddy
        - jack
        - john
        - sameer
```

Now, let’s create the playbook and call the above file in it.

```
$ vim standard_loop3.yml
---
- name: Ansible standard loop playbook to add user accounts with admin access
  hosts: all
  vars_files:
            - /home/vikasnehra/users.yml
  become: true
  tasks:
        - name: Adding user accounts with admin access
          user:
                  name: "{{ item }}"
                  state: present
                  group: wheel
          with_items:
                  - "{{ users }}"
...
```

Let’s check the playbook for errors.

```
$ ansible-playbook standard_loop3.yml --syntax-check
```

Let’s check the playbook execution with dry run.

```
$ ansible-playbook standard_loop3.yml -C
```

Execute the playbook to check it’s working.

```
$ ansible-playbook standard_loop3.yml
```

Verify the task using ansible ad-hoc command.

```
$ ansible all -m command -a 'tail -5 /etc/passwd'
```

We can also directly call the lines from another file without using vars_files option (without importing the complete file) in the playbook. We can directly call the lines instead; this can be done using lines lookup plugin in the ansible.

```
$ ansible-doc -t lookup lines
```

Let’s create a file first which will have the list of the user accounts to be created.

```
$ vim myusers
admin1
admin2
admin3
admin4
admin5
```

Now, let’s create the playbook and use the lines lookup plugin there.

```
$ vim standard_loop4.yml
---
- name: Ansible standard loop playbook to add user accounts with admin access
  hosts: all
  become: true
  tasks:
        - name: Adding user accounts with admin access
          user:
                  name: "{{ item }}"
                  state: present
                  group: wheel
          with_lines: cat myusers
...
```

Let’s check the playbook for errors.

```
$ ansible-playbook standard_loop4.yml --syntax-check
```

Let’s check the playbook execution with dry run.

```
$ ansible-playbook standard_loop4.yml -C
```

Execute the playbook to check it's working.

```
$ ansible-playbook standard_loop4.yml
```

Verify the task using ansible ad-hoc command.

```
$ ansible all -m command -a 'tail -5 /etc/passwd'
```

(b) Looping over List of Hashes:

Hashes are a type of data structure that stores key-value pairs.

Let's suppose we need to create multiple user accounts and each user should be the member of a different secondary group in the managed nodes as per the following table.

Username	Group
rahul	wheel
rohan	wheel
anuj	root
harsha	root

Let's create an ansible playbook for the above task where we will run the loop on list of hashes.

```
$ vim hash_loop.yml
---
- name: Playbook to add multiple users using hash loops
  hosts: all
  become: true
  tasks:
    - name: Adding local user accounts with their groups
      user:
        name: "{{ item.name }}"
        state: present
        groups: "{{ item.groups }}"
      loop:
        - { name: 'rahul', groups: wheel }
        - { name: 'rohan', groups: wheel }
        - { name: 'anuj', groups: root }
        - { name: 'harsha', groups: root }
…
```

Let's check the playbook for errors.

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

Let’s check the playbook execution with dry run.

```
$ ansible-playbook hash_loop.yml -C
```

Execute the playbook to check it’s working.

```
$ ansible-playbook hash_loop.yml
```

Verify the task using ansible ad-hoc command.

```
$ ansible all -m command -a 'tail -5 /etc/passwd'
```
