Цикл по спискам

Ansible использует цикл ключевых слов для перебора элементов списка. Для демонстрации давайте создадим очень простую книгу с именем print-list.yml, которая покажет вам, как распечатать элементы в списке:

```
 [destroyer@andreyex]$ cat print-list.yml
---
- name: print list
  hosts: node1
  vars:
    prime: [2,3,5,7,11]
  tasks:
    - name: Show first five prime numbers
      debug:
        msg: "{{ item }}"
      loop: "{{ prime }}"
```

Обратите внимание, что мы используем переменную item с циклами Ansible. Задача будет выполняться пять раз, что равно количеству элементов в простом списке.

При первом запуске переменная элемента будет установлена на первый элемент в массиве простых чисел (2). При втором запуске переменная item будет установлена на второй элемент в массиве простых чисел (3) и так далее.

Следуйте дальше и запустите playbook, чтобы увидеть все отображаемые элементы основного списка:

```
 [destroyer@andreyex]$ ansible-playbook print-list.yml

PLAY [print list] ************************************************************

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

TASK [Show first five prime numbers] *****************************************
ok: [node1] => (item=2) => {
    "msg": 2
}
ok: [node1] => (item=3) => {
    "msg": 3
}
ok: [node1] => (item=5) => {
    "msg": 5
}
ok: [node1] => (item=7) => {
    "msg": 7
}
ok: [node1] => (item=11) => {
    "msg": 11
}

PLAY RECAP *******************************************************************
node1                      : ok=2    changed=0    unreachable=0    failed=0
```

Теперь вы применяете циклы к реальному приложению. Например, вы можете создать playbook add-users.yml, который добавит нескольких пользователей на все хосты в группе dbservers:

```
 [destroyer@andreyex]$ cat add-users.yml
---
- name: Add multiple users
  hosts: dbservers
  vars:
    dbusers:
      - username: brad
        pass: pass1
      - username: david
        pass: pass2
      - username: jason
        pass: pass3
  tasks:
    - name: Add users
      user:
        name: "{{ item.username }}"
        password: "{{ item.pass | password_hash('sha512') }}"
      loop: "{{ dbusers }}"
```

Сначала мы создали список dbusers, который в основном представляет собой список хешей/словарей. Затем мы использовали пользовательский модуль вместе с циклом, чтобы добавить пользователей и установить пароли для всех пользователей в списке dbusers.

Обратите внимание, что мы также использовали точечную нотацию item.username и item.pass для доступа к значениям ключей внутри хешей/словарей списка dbusers.

Также стоит отметить, что мы использовали фильтр password_hash ('sha512') для шифрования паролей пользователей с помощью алгоритма хеширования sha512, поскольку пользовательский модуль не позволял устанавливать незашифрованные пароли пользователей.

Теперь запустим playbook add-users.yml :

```
 [destroyer@andreyex]$ ansible-playbook add-users.yml

PLAY [Add multiple users] ******************************************************

TASK [Gathering Facts] *********************************************************
ok: [node4]

TASK [Add users] ***************************************************************
changed: [node4] => (item={'username': 'brad', 'pass': 'pass1'})
changed: [node4] => (item={'username': 'david', 'pass': 'pass2'})
changed: [node4] => (item={'username': 'jason', 'pass': 'pass3'})

PLAY RECAP *********************************************************************
node4                      : ok=2    changed=1    unreachable=0    failed=0    skipped=0
```

Читать Важность SEO аудита в продвижении сайтов

Вы можете убедиться, что три пользователя добавлены, выполнив специальную команду Ansible:

```
 [destroyer@andreyex]$ ansible dbservers -m command -a "tail -3 /etc/passwd"
node4 | CHANGED | rc=0 >>
brad:x:1001:1004::/home/brad:/bin/bash
david:x:1002:1005::/home/david:/bin/bash
jason:x:1003:1006::/home/jason:/bin/bash
```

Перебирать словари

Вы можете использовать цикл только со списками. Вы получите сообщение об ошибке, если попытаетесь перебрать словарь.

Например, если вы запустите следующую книгу воспроизведения print-dict.yml:

```
 [destroyer@andreyex]$ cat print-dict.yml
---
- name: Print Dictionary
  hosts: node1
  vars:
    employee:
      name: "destroyer Alderson"
      title: "Penetration Tester"
      company: "Linux AndreyEx"
  tasks:
    - name: Print employee dictionary
      debug:
        msg: "{{ item }}"
      loop: "{{ employee }}"
```

Вы получите следующую ошибку:

```
 [destroyer@andreyex]$ ansible-playbook print-dict.yml

PLAY [Print Dictionary] ********************************************************

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

TASK [Print employee dictionary] ***********************************************
fatal: [node1]: FAILED! => {"msg": "Invalid data passed to 'loop', it requires a list, got this instead: {'name': 'destro
```

Как видите, ошибка явно говорит о том, что требуется список.

Чтобы исправить эту ошибку; вы можете использовать фильтр dict2items для преобразования словаря в список. Итак, в playbook print-dict.yml, отредактируйте строку:

```
 loop: "{{ employee }}"
```

и примените фильтр dict2items следующим образом:

```
 loop: "{{ employee | dict2items }}"
```

Затем снова запустите playbook:

```
 [destroyer@andreyex]$ ansible-playbook print-dict.yml

PLAY [Print Dictionary] ***************************************************

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

TASK [Print employee dictionary] ******************************************
ok: [node1] => (item={'key': 'name', 'value': 'destroyer Alderson'}) => {
    "msg": {
        "key": "name",
        "value": "destroyer Alderson"
    }
}
ok: [node1] => (item={'key': 'title', 'value': 'Penetration Tester'}) => {
    "msg": {
        "key": "title",
        "value": "Penetration Tester"
    }
}
ok: [node1] => (item={'key': 'company', 'value': 'Linux AndreyEx'}) => {
    "msg": {
        "key": "company",
        "value": "Linux AndreyEx"
    }
}

PLAY RECAP ****************************************************************
node1                      : ok=2    changed=0    unreachable=0    failed=0    skipped=0
```

Успех! Были отображены пары ключ/значение словаря сотрудников.

Цикл по диапазону чисел

Вы можете использовать функцию range() вместе с фильтром списка для циклического перебора диапазона чисел.

Например, следующая задача распечатает все числа от 0 до 9:

```
- name: Range Loop
 debug:
   msg: "{{ item }}"
 loop: "{{ range(10) | list }}"
```

Вы также можете начать свой диапазон с числа, отличного от нуля. Например, следующая задача напечатает все числа от 5 до 14:

```
- name: Range Loop
 debug:
   msg: "{{ item }}"
 loop: "{{ range(5,15) | list }}"
```

По умолчанию шаг установлен на 1. Однако вы можете установить другой шаг.

Например, следующая задача распечатает все четные IP-адреса в подсети 192.168.1.x:

```
- name: Range Loop
 debug:
   msg: 192.168.1.{{ item }}
 loop: "{{ range(0,256,2) | list }}"
```

Читать Избавьтесь от проблем с сетевым подключением в SSH с помощью Mosh

Где начало = 0, конец <256 и шаг = 2.

Зацикливание на запасах

Вы можете использовать встроенную переменную групп Ansible, чтобы перебрать все ваши хосты инвентаризации или только их подмножество. Например, чтобы перебрать все ваши инвентарные хосты; ты можешь использовать:

```
loop: "{{ groups['all'] }}"
```

Если вы хотите перебрать все хосты в группе веб-серверов, вы можете использовать:

```
loop: "{{ groups['webservers'] }}"
```

Чтобы увидеть, как это работает в playbook; взгляните на следующую книгу loop-inventory.yml:

```
 [destroyer@andreyex]$ cat loop-inventory.yml
---
- name: Loop over Inventory
  hosts: node1
  tasks:
    - name: Ping all hosts
      command: ping -c 1 "{{ item }}"
      loop: "{{ groups['all'] }}"
```

Этот сценарий проверяет, может ли node1 пинговать все остальные хосты в вашем инвентаре. Идите вперед и запустите playbook:

```
 [destroyer@andreyex]$ ansible-playbook loop-inventory.yml

PLAY [Loop over Inventory] ********************************************************

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

TASK [Ping all hosts] *************************************************************
changed: [node1] => (item=node1)
changed: [node1] => (item=node2)
changed: [node1] => (item=node3)
changed: [node1] => (item=node4)

PLAY RECAP ************************************************************************
node1                      : ok=2    changed=1    unreachable=0    failed=0    skipped=0
```

Если вы получите какие-либо ошибки; это будет означать, что ваши управляемые хосты не могут пинговать (достигать) друг друга.

Пауза в циклах

Вы можете сделать паузу на определенное время между каждой итерацией цикла. Для этого вы можете использовать директиву pause вместе с ключевым словом loop_control.

Чтобы продемонстрировать это, давайте напишем playbook countdown.yml, который будет просто делать десятисекундный обратный отсчет перед отображением сообщения «С днем рождения!» на экране:

```
 [destroyer@andreyex]$ cat countdown.yml
---
- name: Happy Birthday Playbook
  hosts: node1
  tasks:
  - name: Ten seconds countdown
    debug:
      msg: "{{ 10 - item }} осталось несколько секунд ..."
    loop: "{{ range(10) | list }}"
    loop_control:
      pause: 1

  - name: Display Happy Birthday
    debug:
      msg: "С днем рождения!"
```

Идите вперед и запустите playbook: