Oftentimes, we will want to perform a single task, but use that single task to iterate over a set of data. For example, you might not want to create one user account but 10. Or you might want to install 15 packages to a system. The possibilities are endless, but the point remains the same—you would not want to write 10 individual Ansible tasks to create 10 user accounts. Fortunately, Ansible supports looping over datasets to ensure that you can perform large scale operations using tightly defined code. In this section, we will explore how to make practical use of loops in your Ansible playbooks.

**Iterating over a simple loop**

This is a very simple playbook where we just iterate over a loop of items and echo the individual item. To access individual data in the loop we use "**item**"

---

- name: Working with loop module

hosts: localhost

gather\_facts: false

tasks:

- name: Echo the value

command: echo "{{ item }}"

loop:

- 1

- 2

- 3

- 4

- 5

Let us execute this playbook:

[ansible@controller ~]$ ansible-playbook loop-iterate.yml

PLAY [Working with loop module] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Echo the value] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost] => (item=1)

changed: [localhost] => (item=2)

changed: [localhost] => (item=3)

changed: [localhost] => (item=4)

changed: [localhost] => (item=5)

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=1 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So the playbook has neatly printed all the 5 items without spending much effort in writing the playbook or else imaging writing the same thing 5 times to iterate over each value.

**Register output to a variable**

As we capture the output from any command using register, similar can be done for loop output. The only difference is that the results will now be stored in a dictionary, with one dictionary entry for each iteration of the loop rather than just one set of results.

For example in the existing playbook I have reduced the number of item to shorten the output and then I will store the output into "loopresult" variable. Then we will print the content of loopresult variable:

---

- name: Working with loop module

hosts: localhost

gather\_facts: false

tasks:

- name: Echo the value

command: echo "{{ item }}"

loop:

- 1

- 2

- 3

register: loopresult

- name: print the results from loop

debug:

var: loopresult

Let us execute the playbook and observe the result:

[ansible@controller ~]$ ansible-playbook loop-iterate.yml

PLAY [Working with loop module] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Echo the value] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [localhost] => (item=1)

changed: [localhost] => (item=2)

changed: [localhost] => (item=3)

TASK [print the results from loop] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => {

"loopresult": {

"changed": true,

"msg": "All items completed",

"results": [

{

"ansible\_loop\_var": "item",

"changed": true,

"cmd": [

"echo",

"1"

],

"delta": "0:00:00.003420",

"end": "2020-09-25 12:42:46.060420",

"failed": false,

"invocation": {

"module\_args": {

"\_raw\_params": "echo \"1\"",

"\_uses\_shell": false,

"argv": null,

"chdir": null,

"creates": null,

"executable": null,

"removes": null,

"stdin": null,

"stdin\_add\_newline": true,

"strip\_empty\_ends": true,

"warn": true

}

},

"item": 1,

"rc": 0,

"start": "2020-09-25 12:42:46.057000",

"stderr": "",

"stderr\_lines": [],

"stdout": "1",

"stdout\_lines": [

"1"

]

},

{

"ansible\_loop\_var": "item",

"changed": true,

"cmd": [

"echo",

"2"

],

"delta": "0:00:00.003135",

"end": "2020-09-25 12:42:46.288099",

"failed": false,

"invocation": {

"module\_args": {

"\_raw\_params": "echo \"2\"",

"\_uses\_shell": false,

"argv": null,

"chdir": null,

"creates": null,

"executable": null,

"removes": null,

"stdin": null,

"stdin\_add\_newline": true,

"strip\_empty\_ends": true,

"warn": true

}

},

"item": 2,

"rc": 0,

"start": "2020-09-25 12:42:46.284964",

"stderr": "",

"stderr\_lines": [],

"stdout": "2",

"stdout\_lines": [

"2"

]

},

{

"ansible\_loop\_var": "item",

"changed": true,

"cmd": [

"echo",

"3"

],

"delta": "0:00:00.002974",

"end": "2020-09-25 12:42:46.526784",

"failed": false,

"invocation": {

"module\_args": {

"\_raw\_params": "echo \"3\"",

"\_uses\_shell": false,

"argv": null,

"chdir": null,

"creates": null,

"executable": null,

"removes": null,

"stdin": null,

"stdin\_add\_newline": true,

"strip\_empty\_ends": true,

"warn": true

}

},

"item": 3,

"rc": 0,

"start": "2020-09-25 12:42:46.523810",

"stderr": "",

"stderr\_lines": [],

"stdout": "3",

"stdout\_lines": [

"3"

]

}

]

}

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

**Iterating over list of hashes**

For those of you who are familiar with programming languages, the idea of a hash is nothing new. For those of you who are not familiar, a hash is simply a set of data points identified by a key. Within a hash can be multiple keys, and each key has an associated value.

Advertisement

Let's take a look at a basic example of a hash to get a better idea of how this unique but popular data structure works:

Key: Value

FName: Deepak

LName: Prasad

Location: India

From this, table we can see that a key is simply an identifier, and the value that key represents could be any string or data piece stored in the value table that is associated with that specific key.

Here I have created a playbook with some hash value which contains 3 key elements i.e. fname, lname and location which can be used with item to iterate over the loop.

---

- name: Working with loop module

hosts: localhost

gather\_facts: false

tasks:

- name: Iterate over hashes

debug:

msg:

- "Hello '{{ item.fname }}', nice to meet you"

- "Your last name as per our record is '{{ item.lname }}'"

- "Your country of residence is '{{ item.location }}'"

loop:

- { fname: 'deepak', lname: 'prasad', location: 'India' }

- { fname: 'amit', lname: 'kumar', location: 'Argentina' }

- { fname: 'rahul', lname: 'sharma', location: 'Canada' }

- { fname: 'vivek', lname: 'mittal', location: 'Brazil' }

- { fname: 'anurag', lname: 'keshav', location: 'Japan' }

Let us execute this playbook:

[ansible@controller ~]$ ansible-playbook loop-iterate.yml

PLAY [Working with loop module] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Iterate over hashes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => (item={'fname': 'deepak', 'lname': 'prasad', 'location': 'India'}) => {

"msg": [

"Hello 'deepak', nice to meet you",

"Your last name as per our record is 'prasad'",

"Your country of residence is 'India'"

]

}

ok: [localhost] => (item={'fname': 'amit', 'lname': 'kumar', 'location': 'Argentina'}) => {

"msg": [

"Hello 'amit', nice to meet you",

"Your last name as per our record is 'kumar'",

"Your country of residence is 'Argentina'"

]

}

ok: [localhost] => (item={'fname': 'rahul', 'lname': 'sharma', 'location': 'Canada'}) => {

"msg": [

"Hello 'rahul', nice to meet you",

"Your last name as per our record is 'sharma'",

"Your country of residence is 'Canada'"

]

}

ok: [localhost] => (item={'fname': 'vivek', 'lname': 'mittal', 'location': 'Brazil'}) => {

"msg": [

"Hello 'vivek', nice to meet you",

"Your last name as per our record is 'mittal'",

"Your country of residence is 'Brazil'"

]

}

ok: [localhost] => (item={'fname': 'anurag', 'lname': 'keshav', 'location': 'Japan'}) => {

"msg": [

"Hello 'anurag', nice to meet you",

"Your last name as per our record is 'keshav'",

"Your country of residence is 'Japan'"

]

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So we have an organised output with all the values from the hash.

**Iterate over dictionary**

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.  
Below we take data in the simplest form such as key value pair in the YAML format where Key and value are separated by a colon. It is important that you give a whitespace followed y colon differentiating the Key and Value

Fruit: Apple

Vegetable: Carrot

Liquid: Water

Meat: Chicken

Here Keys are Fruit, Vegetable, Liquid, Meat while the respective Values are Apple, Carrot, Water and Chicken

To loop over a dict in Ansible we use dict2items. Here I have written a simple playbook which iterates over a dictionary using item.key to access the content from KEY and item.value to access the content from the value.

Advertisement

---

- name: Working with loop module

hosts: localhost

gather\_facts: false

tasks:

- name: Iterate over dictionary

debug:

msg:

- "The {{ item.key }} of your car is {{ item.value }}"

loop: "{{ my\_car | dict2items }}"

vars:

my\_car:

Color: Blue

Model: Corvette

Transition: Manual

Price: $20,000

Let us check the output from this play:

[ansible@controller ~]$ ansible-playbook loop-iterate.yml

PLAY [Working with loop module] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Iterate over hashes] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [localhost] => (item={'key': 'Color', 'value': 'Blue'}) => {

"msg": [

"The Color of your car is Blue"

]

}

ok: [localhost] => (item={'key': 'Model', 'value': 'Corvette'}) => {

"msg": [

"The Model of your car is Corvette"

]

}

ok: [localhost] => (item={'key': 'Transition', 'value': 'Manual'}) => {

"msg": [

"The Transition of your car is Manual"

]

}

ok: [localhost] => (item={'key': 'Price', 'value': '$20,000'}) => {

"msg": [

"The Price of your car is $20,000"

]

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

localhost : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So our playbook has executed successfully by iterating over the key value pair of dictionary.