Mahammad Azeem <amazeeeeem6@gmail.com>

About ansible

6 messages

Mahammad Azeem <amazeeeeem6@gmail.com> Sun, Aug 12, 2018 at 3:32 AM
To: "amazeeeeem6@gmail.com" <Mahammadazeem14@gmail.com>

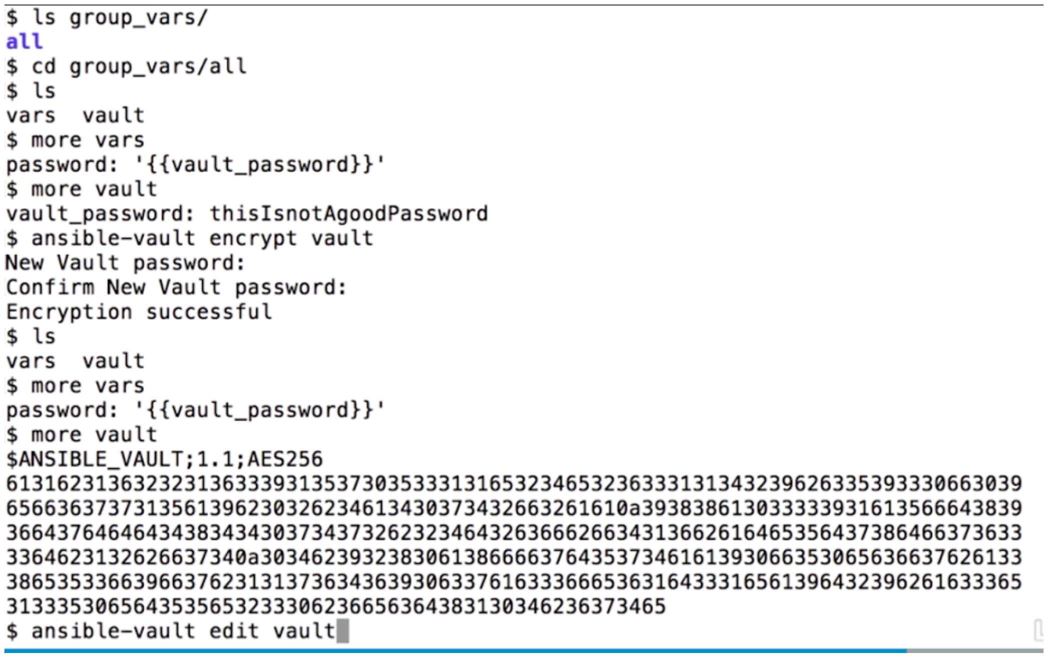
Ansible dynamic inventory:

Since, it is difficult to define variables for dynamic inventory , what we do is we use group_vars and host_vars folder to define variables that needs to be run on specific groups defined under the folder group_vars and specific host/s defined under host_vars.

Here in the picture, under group_vars , the group/s to which the variable needs to be defined is created as a file (file name is same as the name of the group/s , just like how we use in inventory file for defining hosts and groups in static inventory), which consists of the variable. Similarly, for host/s to which variable needs to defined is created as a file (file name should be the ip address of the host)

So , we run the tasks by using below syntax / format
here , two inventories are defined , one for getting the hosts names and the other for our group_vars and host_vars (since, we are dealing with the dynamic inventories)

Mahammad Azeem <amazeeeeem6@gmail.com> Sat, Aug 25, 2018 at 11:18 PM
To: Mahammadazeem14@gmail.com

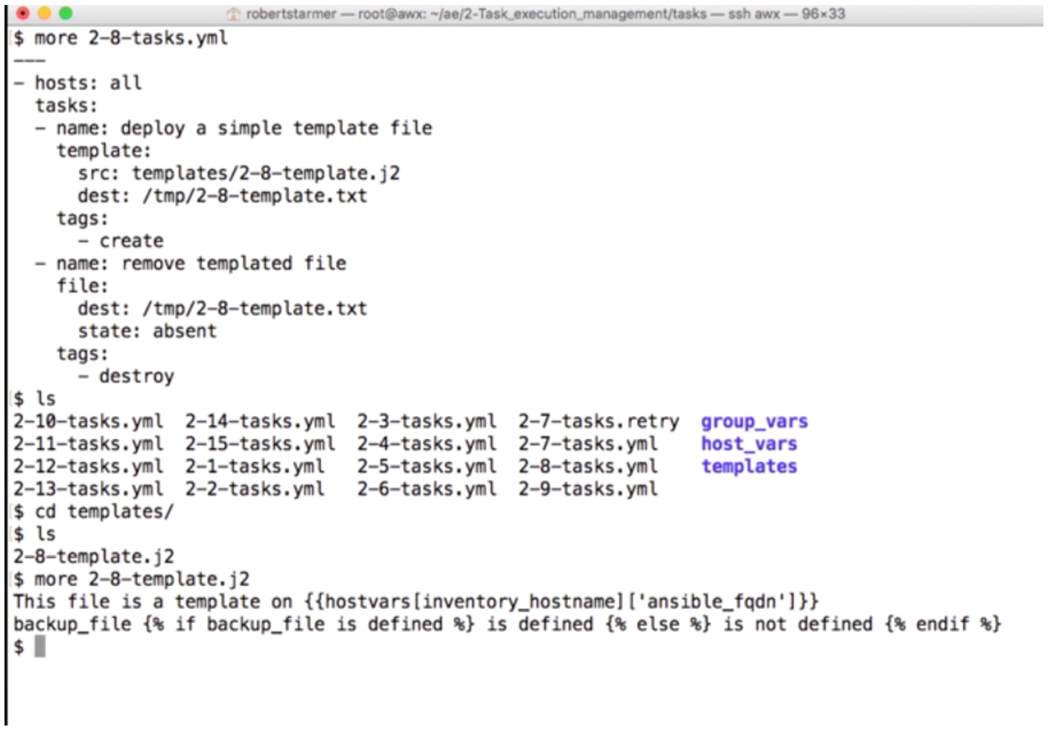


```
$ ls group_vars/
all
$ cd group_vars/all
$ ls
vars  vault
$ more vars
password: '{{vault_password}}'
$ more vault
vault_password: thisIsnotAgoodPassword
$ ansible-vault encrypt vault
New Vault password:
Confirm New Vault password:
Encryption successful
$ ls
vars  vault
$ more vars
password: '{{vault_password}}'
$ more vault
$ANSIBLE_VAULT;1.1;AES256
61316231363232313633393135373035333131653234653236333131343239626335393330663039
6566363737313561396230326234613430373432663261610a393838613033333931613566643839
36643764646434383434303734373262323464326366626634313662616465356437386466373633
3364623132626637340a303462393238306138666637643537346161393066353065636637626133
38653533663966376231313736343639306337616333666536316433316561396432396261633365
3133353065643535653233306236656364383130346236373465
$ ansible-vault edit vault
```

Ansible templates :

Instead of creating or deleting files on the remote machine via varible file_state=touch or absent , we can also make use of template module (Jinja 2 language) to use tags to create and destroy (delete) files on the remote machine.

Playbook using Ansible template module:

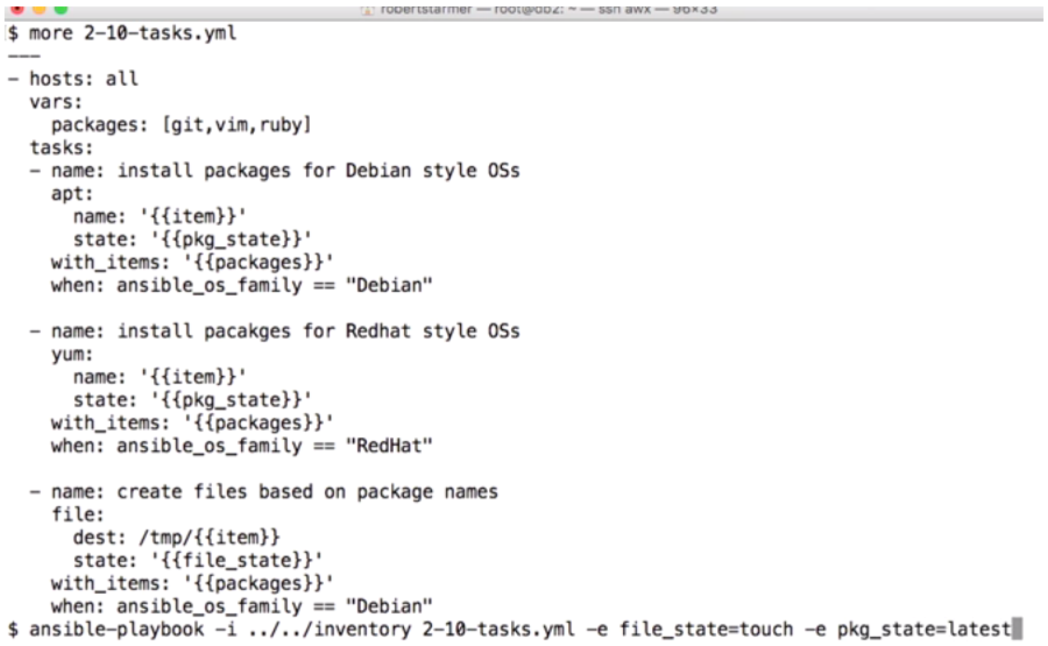


robertstarmer — root@awx: ~/ae/2-Task_execution_management/tasks — ssh awx — 96×33

```
$ more 2-8-tasks.yml
---
- hosts: all
  tasks:
  - name: deploy a simple template file
    template:
      src: templates/2-8-template.j2
      dest: /tmp/2-8-template.txt
    tags:
      - create
  - name: remove templated file
    file:
      dest: /tmp/2-8-template.txt
      state: absent
    tags:
      - destroy
$ ls
2-10-tasks.yml  2-14-tasks.yml  2-3-tasks.yml  2-7-tasks.retry  group_vars
2-11-tasks.yml  2-15-tasks.yml  2-4-tasks.yml  2-7-tasks.yml    host_vars
2-12-tasks.yml  2-1-tasks.yml   2-5-tasks.yml  2-8-tasks.yml    templates
2-13-tasks.yml  2-2-tasks.yml   2-6-tasks.yml  2-9-tasks.yml
$ cd templates/
$ ls
2-8-template.j2
$ more 2-8-template.j2
This file is a template on {{hostvars[inventory_hostname]['ansible_fqdn']}}
backup_file {% if backup_file is defined %} is defined {% else %} is not defined {% endif %}
$
```

Looping task/s (just like for loop in shell scripting)

In the below example, we are installing git , vim and ruby on Debian and red hat operating systems (debian uses apt to installing git and vim where as red hat uses yum)



```
$ more 2-10-tasks.yml
---
- hosts: all
  vars:
    packages: [git,vim,ruby]
  tasks:
  - name: install packages for Debian style OSs
    apt:
      name: '{{item}}'
      state: '{{pkg_state}}'
    with_items: '{{packages}}'
    when: ansible_os_family == "Debian"

  - name: install pacakges for Redhat style OSs
    yum:
      name: '{{item}}'
      state: '{{pkg_state}}'
    with_items: '{{packages}}'
    when: ansible_os_family == "RedHat"

  - name: create files based on package names
    file:
      dest: /tmp/{{item}}
      state: '{{file_state}}'
    with_items: '{{packages}}'
    when: ansible_os_family == "Debian"
$ ansible-playbook -i ../../inventory 2-10-tasks.yml -e file_state=touch -e pkg_state=latest
```

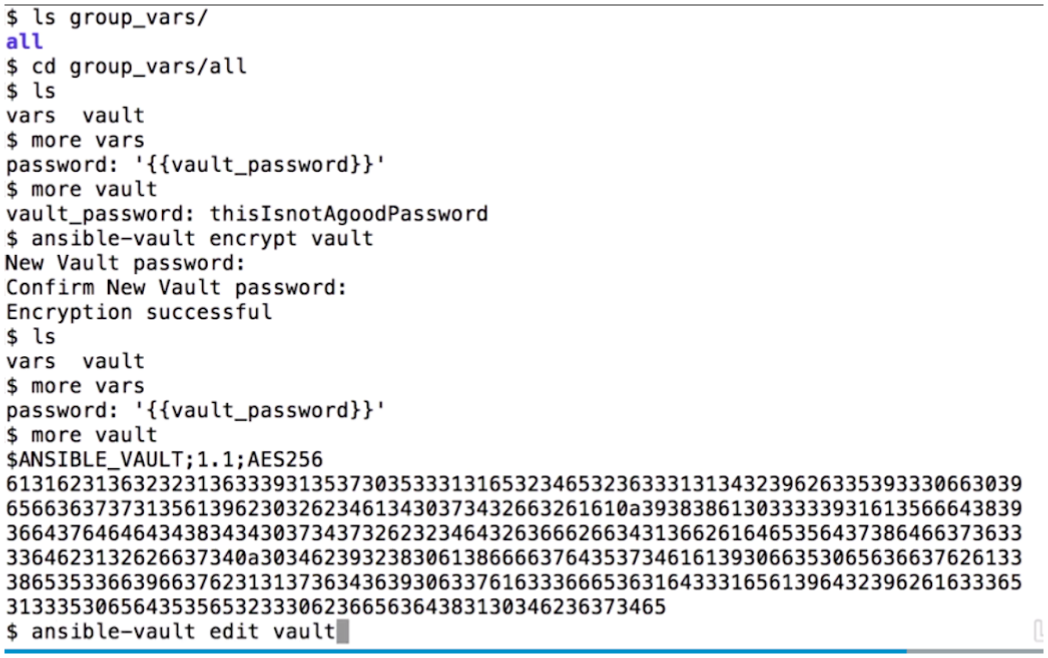
name here is the name of the package to be installed and incase of creating file named ruby (file uby in this case) we need to specify the destination (dest) indicating the location to create the file . When here is similar to if statement. Here, when ansible_os_family (O.S on the remote machine) is "xyz" then, the package/file is installed/created.

We need to pass two variables while running the playbook, one defining the state of the package(package_state=latest) and other for the state of the file (file_state=touch)

Ansible Vault:

Is mechanism to store secrets (passwords, secret text/s, authentication token etc). We create a folder named group_vars and inside it we create the folder name, which is the name of the group defined in the inventory files (vault secret/s acts or used only on these group/s). Then, this folder needs to have two files named vars and vault.

vars will have the variable name/s and vault file will have its value assigned to the variable defined in the vars file , see example below:
vars file has the value vault_password assgined to the password variable,
whereas, the actual password is present in the vault file assigned to the variable vault_password: M$zeem123

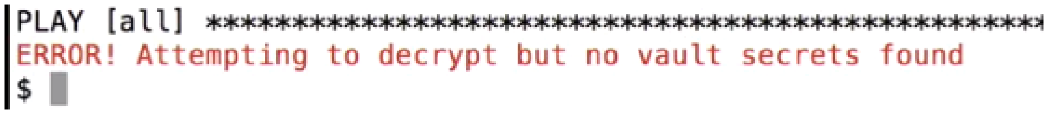


```
$ ls group_vars/
all
$ cd group_vars/all
$ ls
vars  vault
$ more vars
password: '{{vault_password}}'
$ more vault
vault_password: thisIsnotAgoodPassword
$ ansible-vault encrypt vault
New Vault password:
Confirm New Vault password:
Encryption successful
$ ls
vars  vault
$ more vars
password: '{{vault_password}}'
$ more vault
$ANSIBLE_VAULT;1.1;AES256
61316231363232313633393135373035333131653234653236333131343239626335393330663039
6566363737313561396230326234613430373432663261610a393838613033333931613566643839
36643764646434383434303734373262323464326366626634313662616465356437386466373633
3364623132626637340a303462393238306138666637643537346161393066353065636637626133
38653533663966376231313736343639306337616333666536316433316561396432396261633365
3133353065643535653233306236656364383130346236373465
$ ansible-vault edit vault
```

now, once edited the vault file , u can ecrypt it using the command ansible-vault encrypt vault (name of the file to be encrypted) and enter a password .So, when u open the encrypted file (vault in this case) it will have the encrypted value. If u want to edit your password in the vault file sometime later then u need to do the following:

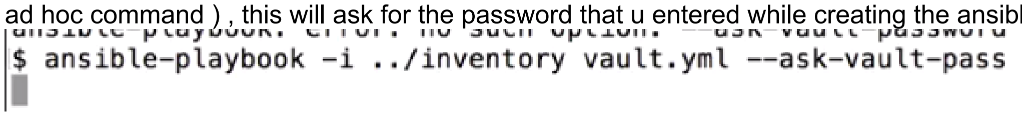
ansible-vault edit vault (name of the file to be encrypted) , it asks for the password (it is the same password that u created while securing vault file).

Suppose, if we want to transfer a file containing password [defined by a {{ password }} variable in the vault file (name of the file)and is ecrypted], but to a remote machine , then the password has to be decrypted , if not then it will through below error



```
PLAY [all] ***************************************************
ERROR! Attempting to decrypt but no vault secrets found
$
```

If, we don't want to decrypt the password present in the vault file (name of the file in this case), then we need to pass --ask-vault-pass or --vault-password-file argument to the ansible command (ansible playbook running command or ansible ad hoc command) , this will ask for the password that u entered while creating the ansible-vault. for this file ...



```
ansible-playbook: error: no such option: --ask-vault-password
$ ansible-playbook -i ../inventory vault.yml --ask-vault-pass
```

If you want to change the vault password used for encrypting the any file then use below command:

ansible-vault rekey vault (name of the vault file in this case)

[Quoted text hidden]

Mahammad Azeem <amazeeeeem6@gmail.com> Sat, Aug 25, 2018 at 11:33 PM
To: "amazeeeeem6@gmail.com" <Mahammadazeem14@gmail.com>

Ansible hosts:

we can read the list of hosts from a file using @ symbol

ansible -i @file_name_containingthehost/s_name.txt -m shell -a ' echo "hello , reading the host/s name/s from the specified .txt file " '

We can use ! to exclude and & intersect group name/s from the inventory / hosts lists:

Assume the following hosts file present under the inventory :

cat /inventory/hosts

[all]

```
xx.com
yy.com
zz.com
11.com
22.com
33.com
44.com
aa.com

[web]
xx.com
11.com

[php]
yy.com
22.com

[mysql]
zz.com
33.com

[other]
44.com
aa.com

[common]
xx.com
aa.com

[lamp]
web:php:mysql:!others   # == hosts that are part of web, php and mysql group but not others

[var]
all:&common # hosts of all group which are part of common group -- xx.com and aa.com in this case

[imp]
web[0]   # only the xx.com of the web group

[imp1]
all[1:3]  # only the hosts from yy.com to zz.com

[imp2]
all[3:] # from 11.com to aa.com
```

[Quoted text hidden]

Mahammad Azeem <amazeeeeem6@gmail.com> Wed, Aug 29, 2018 at 6:18 AM
To: "amazeeeeem6@gmail.com" <Mahammadazeem14@gmail.com>

```
---
- hosts: webservers
  vars:
  ansible_connection: ssh
  ansible_user: ma3020
  ansible_become_user: apache
  become: sudo
  ansible_private_key_file: /opt/app/keyfile
  - name: restarting apache
   tasks:
    service:
        name: httpd
        status: restarted
```

notify and handlers in Ansible playbook:

Ansible modules are idempotent means when a task (using modules) is executed on the remote machine , the playbook is smart enough to recoginze its completion. so whenever, we try to run the same task again it will affect or run since it already ran.

So, assume we have made a configuration change in apache config file and want to restart apache once the change is made on the remote machine, So notify will contain handlers that will include steps or just regular tasks to restart apache when the task is finished like below examples

```
---
- hosts: webservers
  ansible_user: ma3020
  ansible_become_user: apache
  ansible_method: sudo
  ansible_connection: ssh
  - name: copy cofnig update into apcahe config file/s
  tasks:
   copy:
    src: /opt/app/httpd.conf
        dest: /opt/appa/apache/conf/httpd.conf
   notify:
    - restart apache
        - restart memcache
```

Example of a handlers section:

```
handlers:
    - name: restart memcached
      service:
        name: memcached
        state: restarted
    - name: restart apache
      service:
        name: apache
        state: restarted
```

Using listen to call particular handler to be executed once the task/s is finished

```
handlers:
    - name: restart memcached
      service:
        name: memcached
        state: restarted
      listen: "restart web services"
    - name: restart apache
      service:
        name: apache
        state:restarted
      listen: "restart web services"

tasks:
    - name: restart everything
      command: echo "this task will restart the web services"
      notify: "restart web services"
```

NOTE:

If two handler tasks have the same name, only one will run.

To check the syntax of a playbook, use ansible-playbook with the --syntax-check flag.

If you ever want to see detailed output from successful modules as well as unsuccessful ones, use the --verbose flag.

If you see what hosts would be affected by a playbook before you run it, you can do this:

ansible-playbook playbook.yml --list-hosts

[Quoted text hidden]

Mahammad Azeem <amazeeeeem6@gmail.com> Tue, Sep 11, 2018 at 2:58 AM
To: "amazeeeeem6@gmail.com" <Mahammadazeem14@gmail.com>

Ansible template module:

In ansible, template is a file that contains all the configuration related parameters where the dynamic parameters are store as variables. (example: {{ dynamic_variable }})
In order to use template module in ansible we need source and destination parameters (mondatory).

```
src: the source of the template file. This can be relative or absolute path.
dest: the destination path on the remote server
```

Now, we can use jinja2 as the template langauge to get more of the ansible templates functionality (like we can have conditional statements, loops, write macros, filters for transforming the data, do arithmetic calculations, etc.)

Example :

```
- hosts: all
  vars:
    variable_to_be_replaced: 'Hello world'
    inline_variable: 'hello again'
  tasks:
    - name: Ansible Template Example
      template:
        src: hello_world.j2
        dest: /Users/mdtutorials2/Documents/Ansible/hello_world.txt

hello_world.j2
--------------
{{ variable_to_be_replaced }}
This line won't be changed
Variable given as inline - {{ inline_variable }} - :)

output - hello_world.txt
------
Hello world
This line won't be changed
Variable given as inline - hello again - :)

mdtutorials2$ ls -lrt hello_world.txt
-rw-r--r--  1 root  wheel  81 Oct 16 07:23 hello_world.txt
```

```
Here, the variables defined in the playbook are used within {{ }} braces inside the hellow_world.j2 file
and the contents of this file

have been updated as hello_world.txt in the destination path after replacing the variables with their
defined values.If, in the destination

path , only the folder path is given, then the file name will be deployed as hello_world.j2 ( source file
name ) in the destination path.
```

There are other modules of template module which we can use to change some default behaviours

- force
  – If the destination file already exists, then this parameter decides whether it should be replaced or not. By default, the value is 'yes'. So if there is any difference between the rendered source file and the destination file, destination file would be replaced. If you do not want this behaviour, set the value to 'no'.
- Mode – If you want to set the permissions for the destination file explicitly, then you can use this parameter.
- backup
  – If you want a backup file to be created in the destination directory, you should set the value of the backup parameter to 'yes'. By default, the value is 'no'. The backup file will be created every time there is a change in the destination directory. A timestamp would be appended to the filename. So if I changed the 'template.j2' file in the last example two time and also changed the backup parameter to yes, then I get the following two files after two runs.

Using 'for' loop structure inside Ansible template

One of the main program expression we usually use is the 'for' loop. It can be used to iteratively go through the values of a list, dictionary etc.

It is possible to use this in ansible templates also using the jinja2 format.

In the following example, I am looping through the value 0 to 2 using the python range function. On each iteration, a line with the variable is printed.

```
Jinja_loop.j2
-------------
Ansible template for loop example
{% for i in range(3)%}
  This is the {{ i }}th variable
{% endfor %}

output
------
mdtutorials2$$ cat hello_world.txt
Ansible template for loop example
  This is the 0th variable
  This is the 1th variable
  This is the 2th variable
```

But, in the above example, each iteration is printed on new lines. It is because it is retaining the whitespaces.

But in some scenarios, we may want to remove the white spaces. So, how can we do that?

For such scenarios, we can use the minus sign(-) to manually strip the whitespaces including newlines. For example, in the following task, I am adding the '-' sign to the end of the 'for' expression. This will remove the white spaces at the end of the block. The resulting output shows all the variables on the same line.

```
Removing the whitespaces in ansible template
{% for i in range(3) %}
  variable {{ i }}
{%- endfor %}
```

```
output
------
mdtutorials2 $ cat hello_world.txt
Removing the whitespaces in ansible template
variable 0 variable 1 variable 2
```

Using list variables in Ansible templates

In the below task, I am looping over the *list1* variable in the template, using the for loop structure. Note that, after each iteration, a *new line* is also added. So the three list items will be in three lines.

```
- hosts: loc
  vars:
    list1: ['Template iterate','Template loop','Template item']
  tasks:
    - name: Ansible template loop example.
    - template:
        src: templates_example2.j2
        dest: /home/mdtutorials2/output.txt
        mode: 0777

templates_example2.j2
This is an example of template module loop with a list.
{% for item in list1 %}
  {{ item }}
{% endfor %}

output.txt
This is an example of using with_items in template module loop with a list.
Template iterate
Template loop
Template item
```

Ansible template with_items for multiple files

You can use the with_items parameter on a dictionary to render
multiple files. We are using the dictionary since the source and
destination will be different for each template.

In the following example, I am rendering three templates, each with different source and destination.

```
- hosts: loc
  tasks:
    - name: Ansible template with_items example.
      template:
        src: "{{ item.src }}"
        dest: "{{ item.dest }}"
        mode: 0777
      with_items:
        - {src: 'ex.j2',dest: '/home/dnpjose/ex_rew1.txt'}
        - {src: 'ex2.j2',dest: '/home/dnpjose/ex_rew2.txt'}
        - {src: 'ex3.j2',dest: '/home/dnpjose/ex_rew3.txt'}
```

Ansible template comment example

You might give a lot of
comments in your code for clarity. You can also give the comments in the

template file. But sometimes you won't want it to appear in the rendered file. You can do this by giving Jinja2 style comments by enclosing the comments within **{# … #}**.

e.g.: {# This is an Ansible template comment. This won't be shown in the output file #}

```
Loops :

with_items loops over items defined as variable/s


- name: add several users
  user:
    name: "{{ item }}"
    state: present
    groups: "wheel"
  with_items:
     - testuser1
     - testuser2
```

Looping over Files

`with_file` iterates over the content of a list of files, *item* will be set to the content of each file in sequence. It can be used like this:

```
---
- hosts: all

  tasks:

    # emit a debug message containing the content of each file.
    - debug:
        msg: "{{ item }}"
      with_file:
        - first_example_file
        - second_example_file
```

Assuming that `first_example_file` contained the text "hello" and `second_example_file` contained the text "world", this would result in:

```
TASK [debug msg={{ item }}] ******************************************************
ok: [localhost] => (item=hello) => {
    "item": "hello",
    "msg": "hello"
}
ok: [localhost] => (item=world) => {
    "item": "world",
    "msg": "world"
}
```

Looping over Fileglobs

`with_fileglob` matches all files in a single directory, non-recursively, that match a pattern. It calls Python’s glob library, and can be used like this:

```
---
- hosts: all

  tasks:

    # first ensure our target directory exists
    - name: Ensure target directory exists
      file:
        dest: "/etc/fooapp"
        state: directory

    # copy each file over that matches the given pattern
    - name: Copy each file over that matches the given pattern
      copy:
        src: "{{ item }}"
        dest: "/etc/fooapp/"
        owner: "root"
        mode: 0600
      with_fileglob:
        - "/playbooks/files/fooapp/*"
```

[Quoted text hidden]

Mahammad Azeem <amazeeeeem6@gmail.com> Wed, Aug 18, 2021 at 6:08 PM
To: md asif Mm <mdasif.mm@gmail.com>

[Quoted text hidden]
--
Regards,
Azeem
+6590507284