ANSIBLE

Ansible Advanced, Automating With Ansible

Introduction

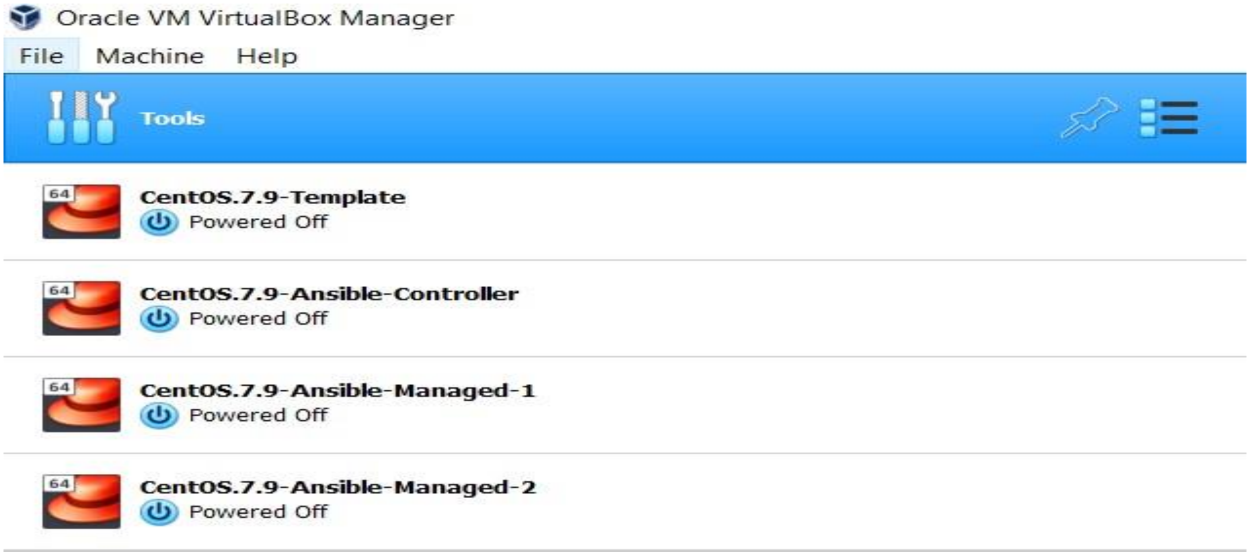
This course was created as a sequel to Ansible Fundamentals. In Ansible Fundamentals, we discussed the basics of working with Ansible and this course continues where Ansible Fundamentals stops. So in case you are approaching Ansible with a foundation outside of the Ansible Fundamentals course, you will need to know how to set up an environment where Ansible is used for configuration management. You should also be familiar with the basics of working with Ansible modules, ad-hook commands, and with playbooks. And you have already learned how to work with Jinja2 templates.

we'll have a look at some of the more advanced Ansible features. Such as working with roles, task control, and using Ansible Vault. Then we will discuss Ansible optimization and troubleshooting.

Lab Requirements

1. MobaXterm Home Edition [Free]
2. Oracle VirtualBox-6.1.20-143896-Win [Free]
3. Oracle_VM_VirtualBox_Extension_Pack-6.1.20 [Free]
4. CentOS Linux release 7.9.2009 (Core) minimal [Free]
5. Ubuntu Linux 20.04.2.0 LTS

Lab Setup

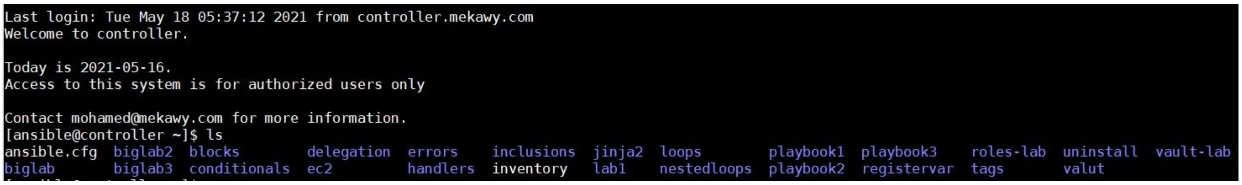


CentOS.7.9-Template
Powered Off

CentOS.7.9-Ansible-Controller
Powered Off

CentOS.7.9-Ansible-Managed-1
Powered Off

CentOS.7.9-Ansible-Managed-2
Powered Off



```
Last login: Tue May 18 05:37:12 2021 from controller.mekawy.com
Welcome to controller.

Today is 2021-05-16.
Access to this system is for authorized users only

Contact mohamed@mekawy.com for more information.
[ansible@controller ~]$ ls
ansible.cfg  biglab2  blocks        delegation  errors    inclusions  jinja2  loops        playbook1  playbook3    roles-lab  uninstall  vault-lab
biglab       biglab3  conditionals  ec2         handlers  inventory   lab1    nestedloops  playbook2  registervar  tags       valut
```

1- Managing Task Control

1.1 Loops

1.1.1 Using with_items

with_items is a very efficient way to have ansible loops over different items. it is the simplest looping mechanism

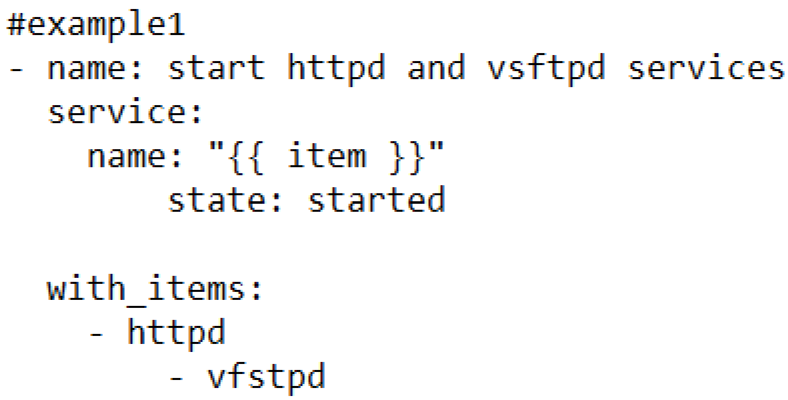
Understand loops:

loops can be used to repeat tasks based on different items. So for each item in the list, or the contents of files in a list, or a sequence of numbers and more can be used in a loop.

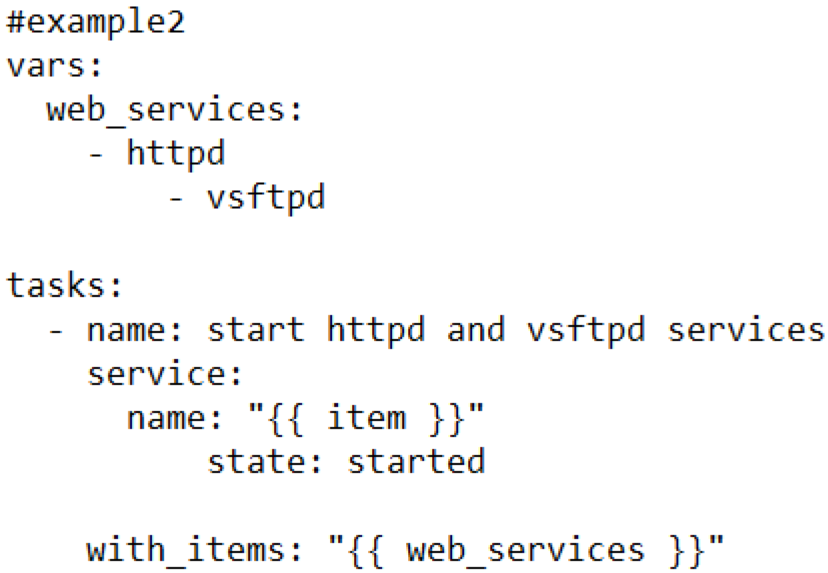
You can use loops to avoid writing multiple tasks, if that's not necessary, so using loops makes your playbook more efficient.

with_items it defines a list of items that needs to be processed.

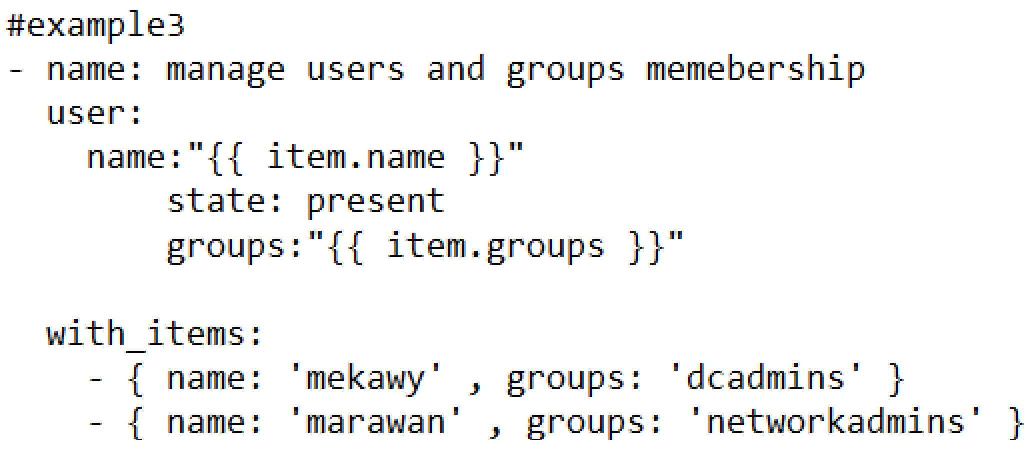
with_items is used as a label that lists the different items, and to refer to an item you can use {{item}} and refer to it just the way you would refer to a variable.



```
#example1
- name: start httpd and vsftpd services
  service:
    name: "{{ item }}"
        state: started

  with_items:
    - httpd
        - vfstpd
```



```
#example2
vars:
  web_services:
    - httpd
        - vsftpd

tasks:
  - name: start httpd and vsftpd services
    service:
      name: "{{ item }}"
          state: started

    with_items: "{{ web_services }}"
```



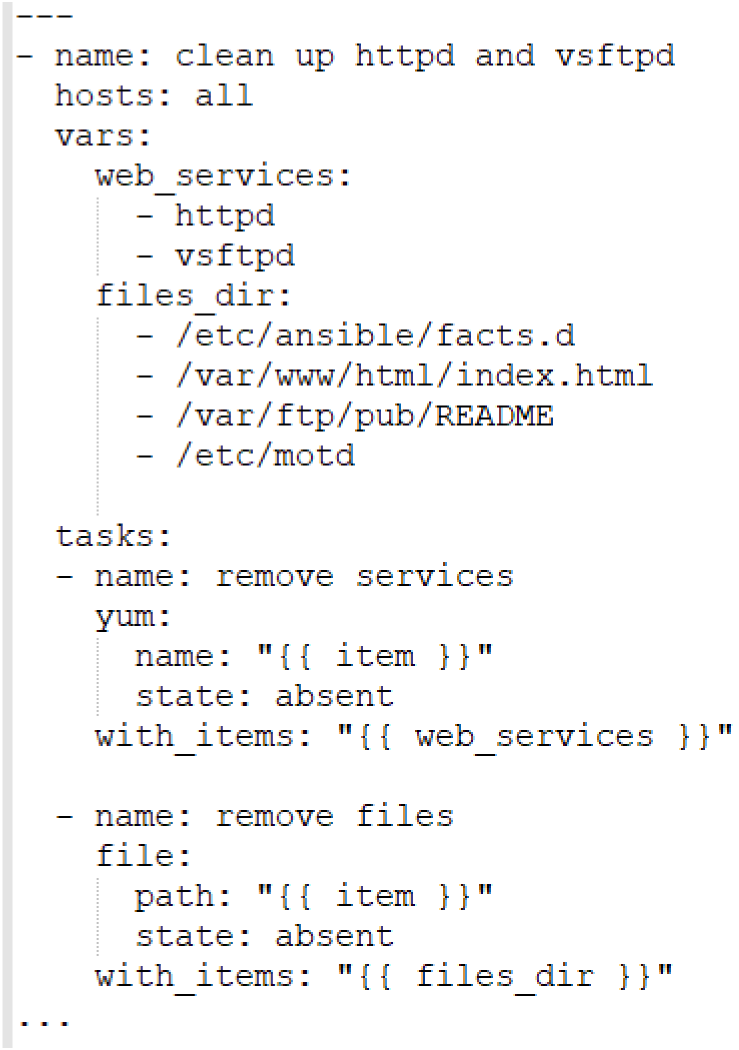
```
#example3
- name: manage users and groups memebership
  user:
    name:"{{ item.name }}"
        state: present
        groups:"{{ item.groups }}"

  with_items:
    - { name: 'mekawy' , groups: 'dcadmins' }
    - { name: 'marawan' , groups: 'networkadmins' }
```

#example 4

```
[ansible@controller ~]$ mkdir uninstall

[ansible@controller ~]$ cd uninstall/

[ansible@controller uninstall]$ vim inventory [all]

controller.mekawy.com

managed1.mekawy.com

managed2.mekawy.com [lamp]

managed2.mekawy.com [file]

managed1.mekawy.com

[ansible@controller uninstall]$ vim ansible.cfg
```

```
[defaults] remote_user=ansible
host_key_checking=false
inventory=inventory

[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False
```

```
[ansible@controller uninstall]$ vim uninstall.yml
```

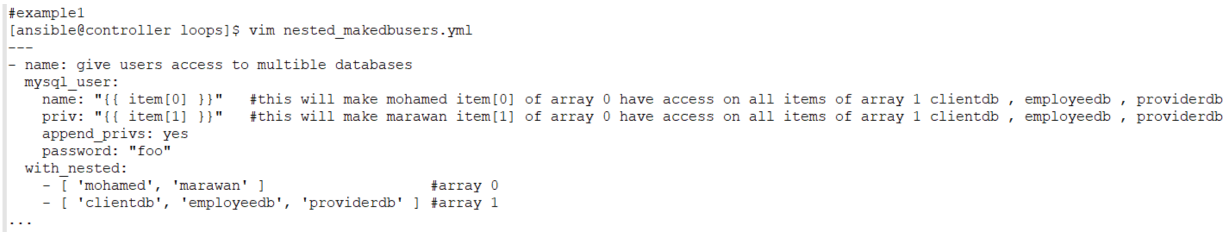


```
---
- name: clean up httpd and vsftpd
  hosts: all
  vars:
    web_services:
      - httpd
      - vsftpd
    files_dir:
      - /etc/ansible/facts.d
      - /var/www/html/index.html
      - /var/ftp/pub/README
      - /etc/motd

  tasks:
  - name: remove services
    yum:
      name: "{{ item }}"
      state: absent
    with_items: "{{ web_services }}"

  - name: remove files
    file:
      path: "{{ item }}"
      state: absent
    with_items: "{{ files_dir }}"
...
```

```
[ansible@controller uninstall]$ ls ansible.cfg
inventory uninstall.yml

[ansible@controller uninstall]$ ansible-playbook uninstall.yml
```

1.1.2 Using Nested Loops

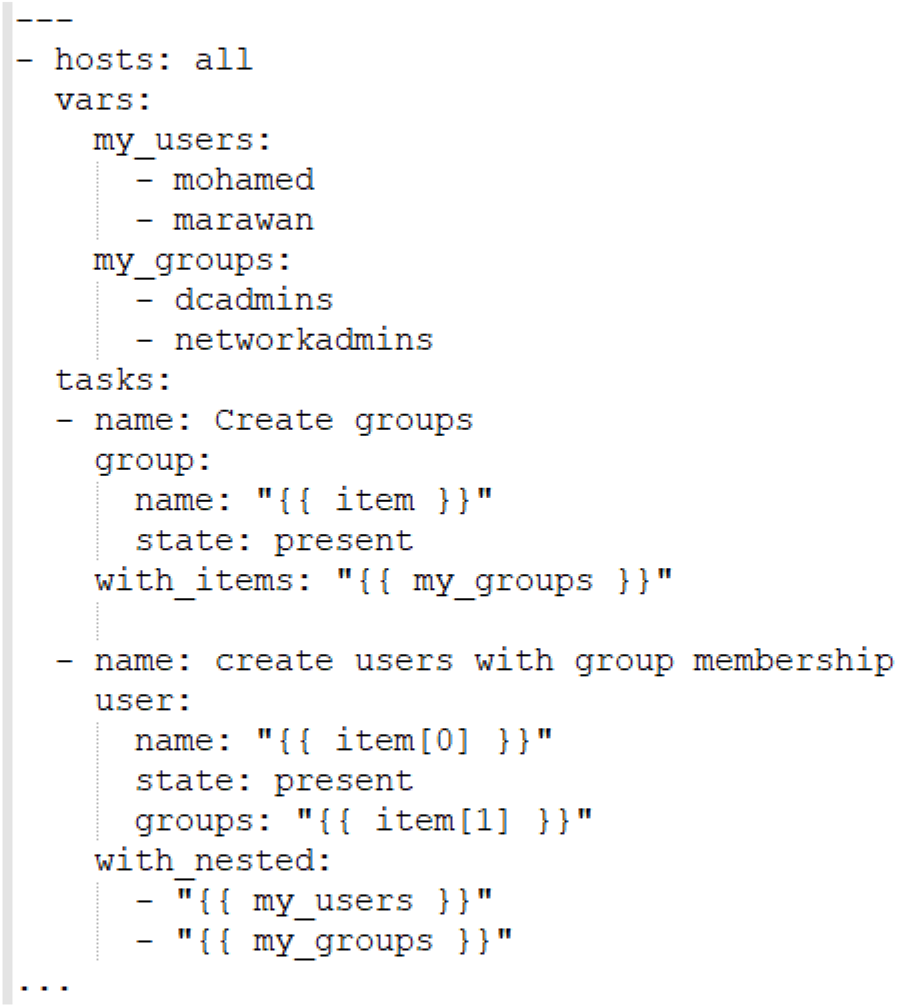
A nested loop in a loop within a loop. Two lists are used in a nested loop and tasks will run on one item in the first list, combined with items in the second list.



```
#example1
[ansible@controller loops]$ vim nested_makedbusers.yml
---
- name: give users access to multible databases
  mysql_user:
    name: "{{ item[0] }}"   #this will make mohamed item[0] of array 0 have access on all items of array 1 clientdb , employeedb , providerdb
    priv: "{{ item[1] }}"   #this will make marawan item[1] of array 0 have access on all items of array 1 clientdb , employeedb , providerdb
    append_privs: yes
    password: "foo"
  with_nested:
    - [ 'mohamed', 'marawan' ]                   #array 0
    - [ 'clientdb', 'employeedb', 'providerdb' ] #array 1
...
```

#example2

```
[ansible@controller ~]$ mkdir nestedloops

[ansible@controller nestedloops]$ cp /home/ansible/uninstall/* .
[ansible@controller nestedloops]$ ls

ansible.cfg  inventory

[ansible@controller nestedloops]$ vim group_user_membership.yml
```



```
---
- hosts: all
  vars:
    my_users:
      - mohamed
      - marawan
    my_groups:
      - dcadmins
      - networkadmins
  tasks:
  - name: Create groups
    group:
      name: "{{ item }}"
      state: present
    with_items: "{{ my_groups }}"

  - name: create users with group membership
    user:
      name: "{{ item[0] }}"
      state: present
      groups: "{{ item[1] }}"
    with_nested:
      - "{{ my_users }}"
      - "{{ my_groups }}"
...
```

```
[ansible@controller nestedloops]$ ansible-playbook group_user_membership.yml

PLAY [all]
*****************************************************************************
**************************************************************

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

[managed2.mekawy.com]

ok: [managed1.mekawy.com]
```

```
ok: [controller.mekawy.com]

TASK [Create groups]
********************************************************************************
***************************************************** changed:

[managed1.mekawy.com] => (item=dcadmins) changed:

[managed2.mekawy.com] => (item=dcadmins) changed:

[controller.mekawy.com] => (item=dcadmins) changed:

[managed2.mekawy.com] => (item=networkadmins) changed:

[managed1.mekawy.com] => (item=networkadmins) changed:

[controller.mekawy.com] => (item=networkadmins)

TASK [create users with group membership]
********************************************************************************

*********************************

changed: [managed1.mekawy.com] => (item=[u'mohamed', u'dcadmins']) changed:

[managed2.mekawy.com] => (item=[u'mohamed', u'dcadmins']) changed:

[controller.mekawy.com] => (item=[u'mohamed', u'dcadmins']) changed:

[managed2.mekawy.com]=>(item=[u'mohamed',u'networkadmins']) changed:

[managed1.mekawy.com]=>(item=[u'mohamed',u'networkadmins']) changed:

[controller.mekawy.com]=>(item=[u'mohamed',u'networkadmins']) changed:

[managed1.mekawy.com] => (item=[u'marawan', u'dcadmins']) changed:

[managed2.mekawy.com] => (item=[u'marawan', u'dcadmins']) changed:

[controller.mekawy.com] => (item=[u'marawan', u'dcadmins']) changed:

[managed1.mekawy.com]=>(item=[u'marawan',u'networkadmins'])
```

```
changed: [managed2.mekawy.com] => (item=[u'marawan', u'networkadmins']) changed:

[controller.mekawy.com] => (item=[u'marawan', u'networkadmins'])

PLAY RECAP

*****************************************************************************

*************************************************************

controller.mekawy.com        : ok=3    changed=2     unreachable=0     failed=0    skipped=0
rescued=0     ignored=0

managed1.mekawy.com          : ok=3    changed=2     unreachable=0     failed=0    skipped=0
rescued=0     ignored=0

managed2.mekawy.com          : ok=3    changed=2     unreachable=0     failed=0    skipped=0
rescued=0     ignored=0

#Verification

[ansible@controller nestedloops]$ ansible all -m command -a "id mohamed"

managed2.mekawy.com | CHANGED | rc=0 >>

uid=1002(mohamed) gid=1004(mohamed) groups=1004(mohamed),1003(networkadmins)

managed1.mekawy.com | CHANGED | rc=0 >>

uid=1002(mohamed) gid=1004(mohamed) groups=1004(mohamed),1003(networkadmins)

controller.mekawy.com | CHANGED | rc=0 >>

uid=1002(mohamed) gid=1004(mohamed) groups=1004(mohamed),1003(networkadmins)

[ansible@controller nestedloops]$ ansible all -m command -a "id marawan" managed1.mekawy.com

| CHANGED | rc=0 >>

uid=1003(marawan) gid=1005(marawan) groups=1005(marawan),1003(networkadmins)

managed2.mekawy.com | CHANGED | rc=0 >>
```

```
uid=1003(marawan) gid=1005(marawan) groups=1005(marawan),1003(networkadmins)
controller.mekawy.com | CHANGED | rc=0 >>

uid=1003(marawan) gid=1005(marawan) groups=1005(marawan),1003(networkadmins)
```

1.1.3.Other loop types

There are some other loop types that can be used as well. For a full list you can consult the documentation.like

1. with_file: which evaluates a list of files

2. with_fileglob: which evaluated a list of files based on a globing pattern. Globing pattern means a*, as in, everything starting with an a.

3. With_sequence: which generates a sequence of items in increasing numerical order. It can work with "start" and "end" to define a range, and it works with decimal, octal, or even hexadecimal integer values.

4. with_random_choice: which takes a list, and "item" is set to one of the list items at random.

1.2 Conditionals

Conditionals make sure that tasks only run if a host meets specific conditions, and that's also something that you can use on Ansible facts.

So in conditionals, operators can be used such as string comparison, mathematical operators or booleans.

And the conditionals can look at different items like a value of a registered variable, an Ansible fact, or even the output of a command, and different operators can be used.

To test conditionals, you can use string comparison, mathematical operators, and or booleans.

1.2.1 Operators examples

Equal on string: {{ ansible_machine }} == "x86_64" Equal on

numeric: {{ max_memory }} == 1024 less than: {{

min_memory }} < 128

grater than: {{ min_memory }} > 256

less than or equal: {{min_memory}}<=512 not

equal: {{ min_memory }} != 512

variable exists: {{ min_memory }}is defined

variable does not exists: {{ min_memory }}is not defined variable is

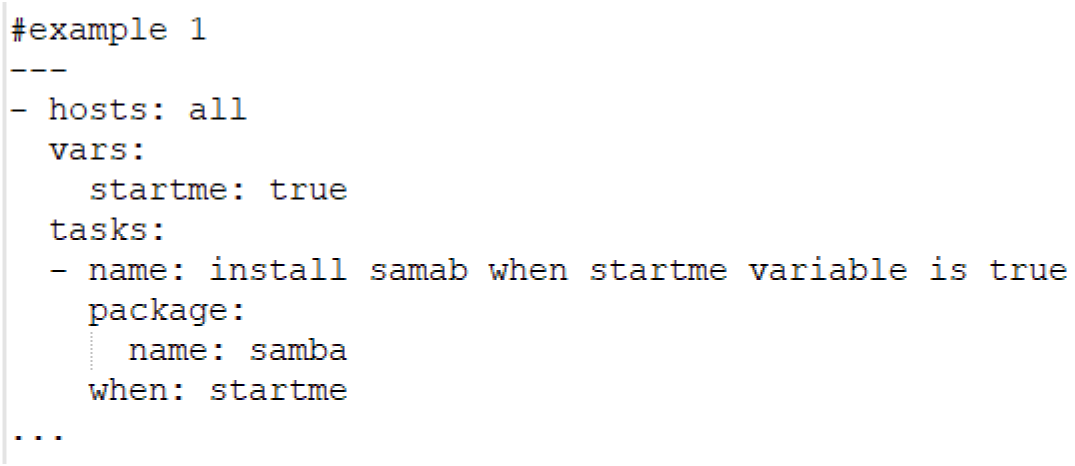
set to yes, true or 1: {{ available_memory }} variable is set to no,

false or 0: not{{available_memory}}

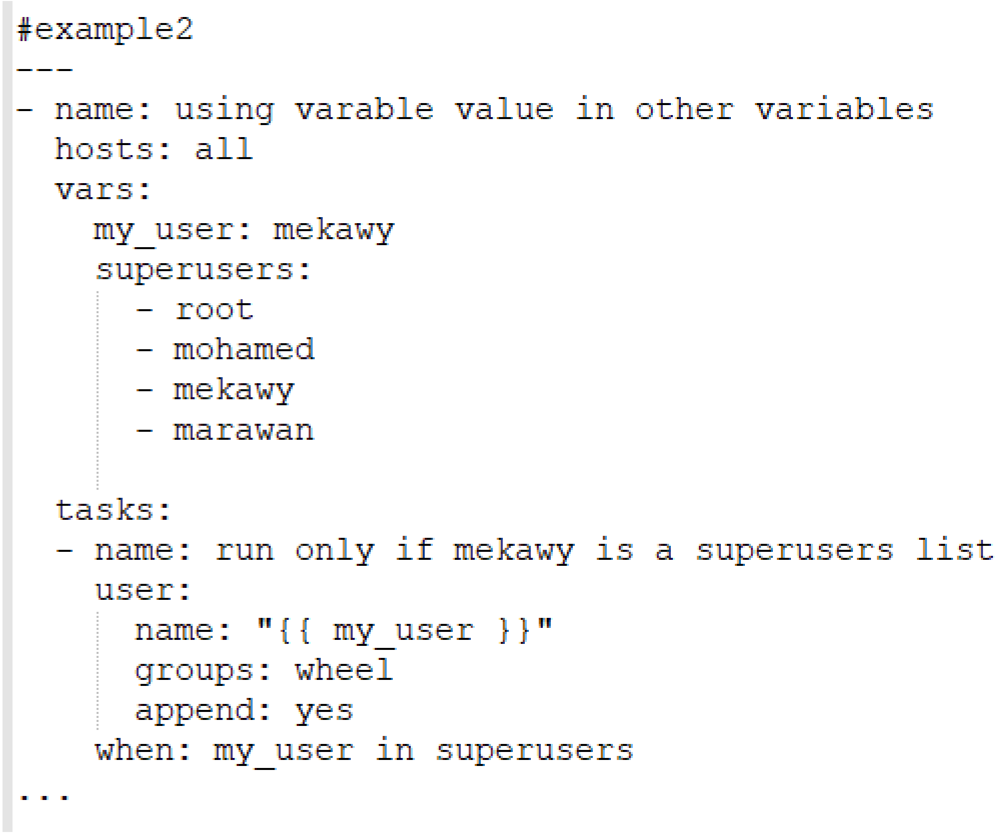
value is present in a variable or array: {{ users }} in users["db_admins"]

1.2.2 When Statement

The when statement is used to implement a condition.Notice that the when statement must be indented outside of the module at the top level of the task.So don't include it in the module, that won't work.



```
#example 1
---
- hosts: all
  vars:
    startme: true
  tasks:
  - name: install samab when startme variable is true
    package:
      name: samba
    when: startme
...
```



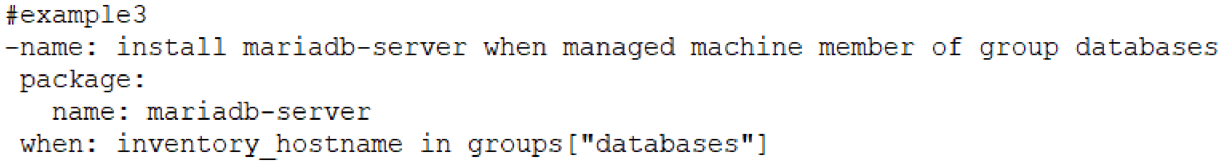
```
#example2
---
- name: using varable value in other variables
  hosts: all
  vars:
    my_user: mekawy
    superusers:
      - root
      - mohamed
      - mekawy
      - marawan

  tasks:
  - name: run only if mekawy is a superusers list
    user:
      name: "{{ my_user }}"
      groups: wheel
      append: yes
    when: my_user in superusers
...
```

1.2.3 Magic Variables

Magic variables are variables that are provided automatically by ansible and cannot be used by users like,

hostvars: which allows to request variables set on other hosts, including facts group_names: an array of all groups the host currently using groups: which

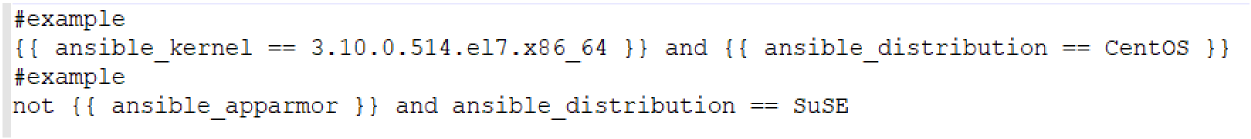
is a list of all hosts and groups in the inventory



```
#example3
-name: install mariadb-server when managed machine member of group databases
 package:
   name: mariadb-server
 when: inventory_hostname in groups["databases"]
```

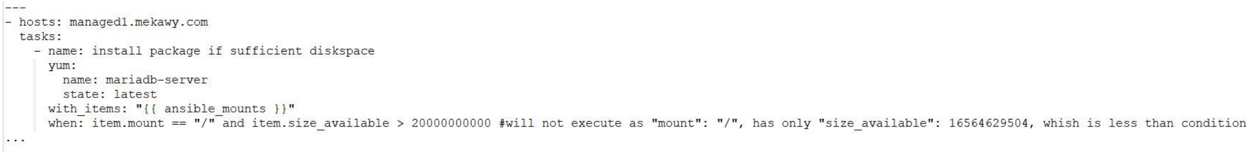
1.2.4 Multiple Conditions

Multiple conditions can be combined with and and or keywords, or grouped in parentheses



```
#example
{{ ansible_kernel == 3.10.0.514.el7.x86_64 }} and {{ ansible_distribution == CentOS }}
#example
not {{ ansible_apparmor }} and ansible_distribution == SuSE
```

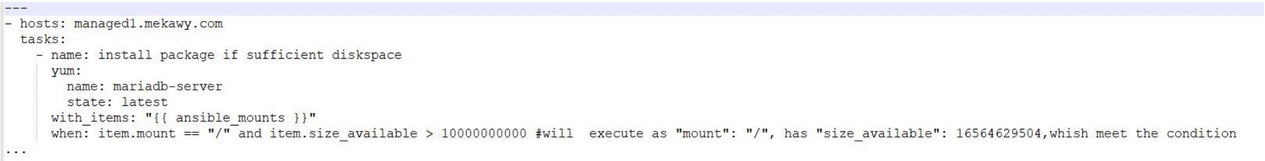
```
#Example conditional 1 [ansible@controller

biglab]$ cd .. [ansible@controller ~]$ mkdir

conditionals [ansible@controller ~]$ cd

conditionals/

[ansible@controller conditionals]$ cp /home/ansible/lab1/inventory .

[ansible@controller conditionals]$ ansible managed1.mekawy.com -i inventory -m setup -a
'filter=ansible_mounts'

managed1.mekawy.com | SUCCESS => {

  "ansible_facts": {

     "ansible_mounts": [

        {

          "block_available": 224587,
          "block_size": 4096,
          "block_total": 259584,
          "block_used": 34997,
          "device": "/dev/sda1",
          "fstype": "xfs",
          "inode_available": 523962,
          "inode_total": 524288,
          "inode_used": 326,
          "mount": "/boot",
          "options": "rw,seclabel,relatime,attr2,inode64,noquota",
          "size_available": 919908352,
          "size_total": 1063256064,
          "uuid": "176f0be6-cc4c-4ecf-a972-5bce29eb70ee"
        },
        {
          "block_available": 4044099,
          "block_size": 4096,
          "block_total": 4452864,
          "block_used": 408765,
          "device": "/dev/mapper/centos-root", "fstype":
          "xfs",
                "inode_available": 8873131,
                "inode_total": 8910848,
                "inode_used": 37717,
                "mount": "/",
                "options": "rw,seclabel,relatime,attr2,inode64,noquota",
                "size_available": 16564629504,
                "size_total": 18238930944,
                "uuid": "442cc7b6-4a72-4b4b-aa36-df4935896b13"
            }
        ],
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
[ansible@controller conditionals]$ vim installif.yml
```



```
---
- hosts: managed1.mekawy.com
  tasks:
    - name: install package if sufficient diskspace
      yum:
        name: mariadb-server
        state: latest
      with_items: "{{ ansible_mounts }}"
      when: item.mount == "/" and item.size_available > 20000000000 #will not execute as "mount": "/", has only "size_available": 16564629504, whish is less than condition
...
```

```
[ansible@controller conditionals]$ ansible-playbook installif.yml -i inventory PLAY

[managed1.mekawy.com]
********************************************************************************
*************************************************
```

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

[managed1.mekawy.com]

TASK [install package if sufficient diskspace]
****************************************************************************

**************************

skipping: [managed1.mekawy.com] => (item={u'block_used': 34997, u'uuid': u'176f0be6-cc4c-
4ecf-a972-5bce29eb70ee', u'size_total': 1063256064, u'block_total': 259584, u'mount': u'/boot',
u'block_available': 224587, u'size_available': 919908352,
u'fstype': u'xfs', u'inode_total': 524288, u'inode_available': 523962, u'device': u'/dev/sda1', u'inode_used': 326,
u'block_size': 4096, u'options': u'rw,seclabel,relatime,attr2,inode64,noquota'})

skipping: [managed1.mekawy.com] => (item={u'block_used': 408813, u'uuid': u'442cc7b6-4a72-
4b4b-aa36-df4935896b13', u'size_total': 18238930944, u'block_total': 4452864, u'mount': u'/',
u'block_available': 4044051, u'size_available': 16564432896, u'fstype': u'xfs', u'inode_total':
8910848, u'inode_available': 8873131, u'device': u'/dev/mapper/centos-root', u'inode_used':
37717, u'block_size': 4096, u'options': u'rw,seclabel,relatime,attr2,inode64,noquota'})

PLAY RECAP
****************************************************************************

***************************************************************

managed1.mekawy.com         : ok=1    changed=0    unreachable=0    failed=0    skipped=1
rescued=0    ignored=0

[ansible@controller conditionals]$ vim installif.yml
```



```
---
- hosts: managed1.mekawy.com
  tasks:
    - name: install package if sufficient diskspace
      yum:
        name: mariadb-server
        state: latest
      with_items: "{{ ansible_mounts }}"
      when: item.mount == "/" and item.size_available > 10000000000 #will  execute as "mount": "/", has "size_available": 16564629504,whish meet the condition
...
```

```
[ansible@controller conditionals]$ ansible-playbook installif.yml -i inventory

PLAY [managed1.mekawy.com]
********************************************************************************

*************************************************

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

[managed1.mekawy.com]

TASK [install package if sufficient diskspace]
********************************************************************************

***************************

skipping: [managed1.mekawy.com] => (item={u'block_used': 34997, u'uuid': u'176f0be6-cc4c-
4ecf-a972-5bce29eb70ee', u'size_total': 1063256064, u'block_total': 259584, u'mount': u'/boot',
u'block_available': 224587, u'size_available': 919908352,
u'fstype': u'xfs', u'inode_total': 524288, u'inode_available': 523962, u'device': u'/dev/sda1', u'inode_used': 326,
u'block_size': 4096, u'options': u'rw,seclabel,relatime,attr2,inode64,noquota'})

ok: [managed1.mekawy.com] => (item={u'block_used': 445277, u'uuid':
u'442cc7b6-4a72-4b4b-aa36-df4935896b13', u'size_total': 18238930944, u'block_total': 4452864,
u'mount': u'/', u'block_available': 4007587, u'size_available': 16415076352, u'fstype': u'xfs',
u'inode_total': 8910848, u'inode_available': 8872674, u'device': u'/dev/mapper/centos-root',
u'inode_used': 38174, u'block_size': 4096, u'options':
u'rw,seclabel,relatime,attr2,inode64,noquota'})
```

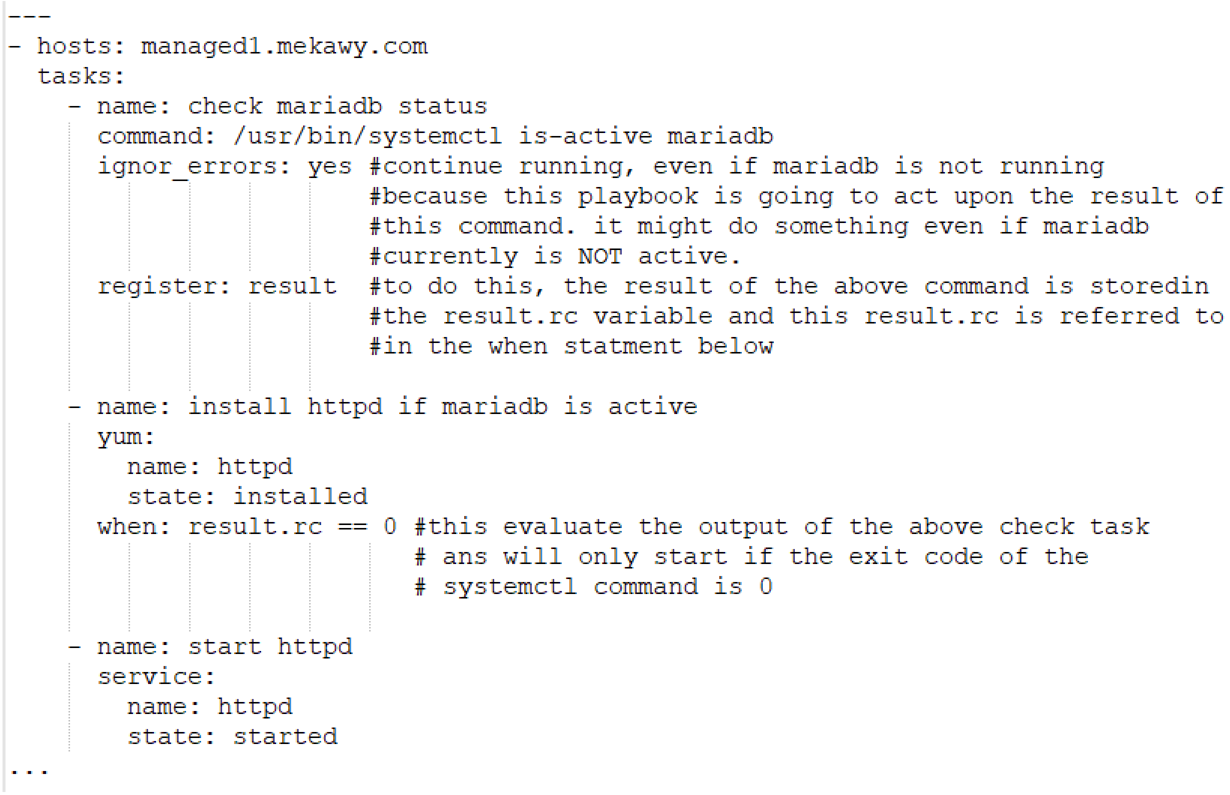
PLAY RECAP

****

****

managed1.mekawy.com   : ok=2   changed=0   unreachable=0   failed=0   skipped=0
rescued=0   ignored=0

#Example conditional 2



```
---
- hosts: managed1.mekawy.com
  tasks:
    - name: check mariadb status
      command: /usr/bin/systemctl is-active mariadb
      ignor_errors: yes #continue running, even if mariadb is not running
                        #because this playbook is going to act upon the result of
                        #this command. it might do something even if mariadb
                        #currently is NOT active.
      register: result  #to do this, the result of the above command is storedin
                        #the result.rc variable and this result.rc is referred to
                        #in the when statment below

    - name: install httpd if mariadb is active
      yum:
        name: httpd
        state: installed
      when: result.rc == 0 #this evaluate the output of the above check task
                           # ans will only start if the exit code of the
                           # systemctl command is 0

    - name: start httpd
      service:
        name: httpd
        state: started
...
```

#Example Conditional 3

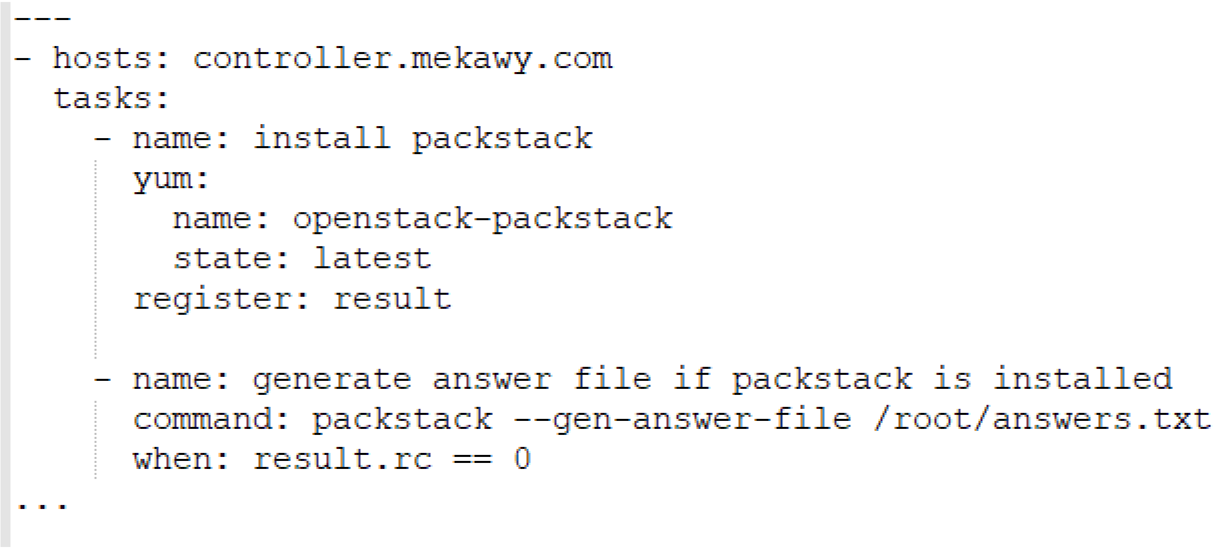
Requirements:

write an Ansible playbook that runs some tasks to setup an OpenStack controller.

It should install the openstack-packstack package and if that is successful, it should run the packstack --gen-answer-file/root/answers.txt command.

Solution

```
[ansible@controller conditionals]$ yum search openstack

centos-release-openstack-queens.noarch : OpenStack from the CentOS Cloud SIG repo configs

[ansible@controller conditionals]$ yum -y install centos-release-openstack-queens.noarch [ansible@controller
conditionals]$ vim setup_controller.yml
```



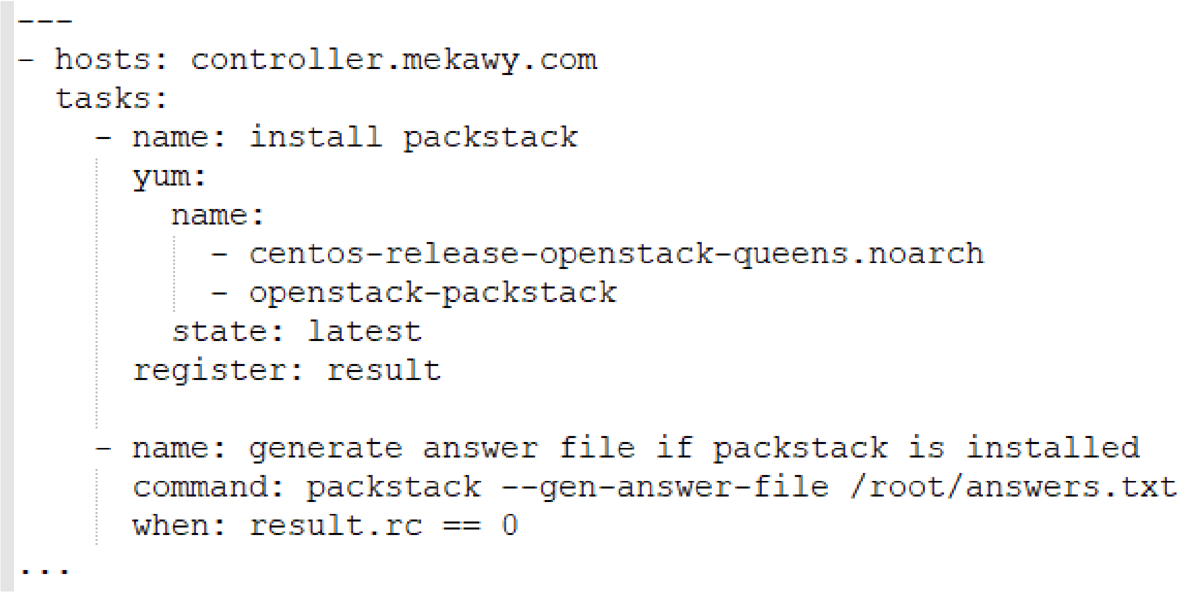
```
---
- hosts: controller.mekawy.com
  tasks:
    - name: install packstack
      yum:
        name: openstack-packstack
        state: latest
      register: result

    - name: generate answer file if packstack is installed
      command: packstack --gen-answer-file /root/answers.txt
      when: result.rc == 0
...
```

```
[ansible@controller conditionals]$ ansible-playbook setup_controller.yml -i inventory PLAY

[controller.mekawy.com]
*****************************************************************************
*********************************************

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

[controller.mekawy.com]
```

```
TASK [install packstack]
*****************************************************************************
***********************************************

changed: [controller.mekawy.com]

TASK [generate answer file if packstack is installed]
*****************************************************************************

*******************

changed: [controller.mekawy.com]

PLAY RECAP
*****************************************************************************

****************************************************************

controller.mekawy.com         : ok=3    changed=2    unreachable=0     failed=0    skipped=0
rescued=0     ignored=0
```

#OR you can replace it with

```
[ansible@controller conditionals]$ vim setup_controller_v2.yml
```

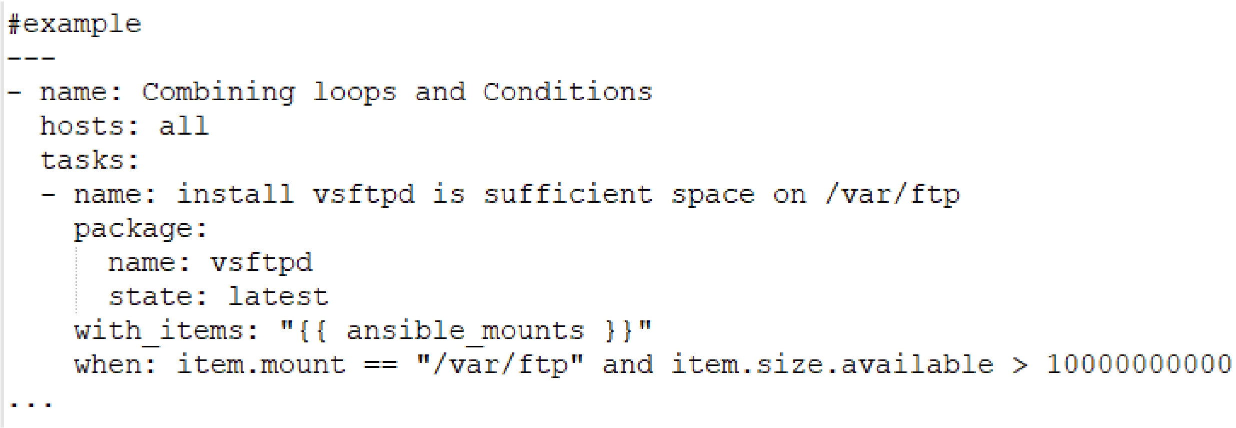


```
---
- hosts: controller.mekawy.com
  tasks:
    - name: install packstack
      yum:
        name:
          - centos-release-openstack-queens.noarch
          - openstack-packstack
        state: latest
      register: result

    - name: generate answer file if packstack is installed
      command: packstack --gen-answer-file /root/answers.txt
      when: result.rc == 0
...
```

```
[ansible@controller conditionals]$ ansible-playbook setup_controller_v2.yml -i inventory

PLAY [controller.mekawy.com]
*************************************************************************

*******************************************

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

[controller.mekawy.com]

TASK [install packstack]
*************************************************************************
********************************************* ok:

[controller.mekawy.com]

TASK [generate answer file if packstack is installed]
*************************************************************************

******************

changed: [controller.mekawy.com]

PLAY RECAP
*************************************************************************

*************************************************************

controller.mekawy.com      : ok=3    changed=1     unreachable=0     failed=0     skipped=0
rescued=0      ignored=0

[root@controller ~]# vim /root/answers.txt
```

1.2.5 Combining loops and Conditions

Ansible facts may present a dictionary with multiple values and in that case, you can iterate through each value until a specific condition is met.



```
#example
---
- name: Combining loops and Conditions
  hosts: all
  tasks:
  - name: install vsftpd is sufficient space on /var/ftp
    package:
      name: vsftpd
      state: latest
    with_items: "{{ ansible_mounts }}"
    when: item.mount == "/var/ftp" and item.size.available > 10000000000
...
```

1.3 Registering Variables

The idea of registering a variable is to store the output of a task or a command in a variable.

And if the result is multi-value, each value is stored in a key and you can iterate through the different keys, and to do something with it, you need to refer to a specific value in the varname.value format.

#example

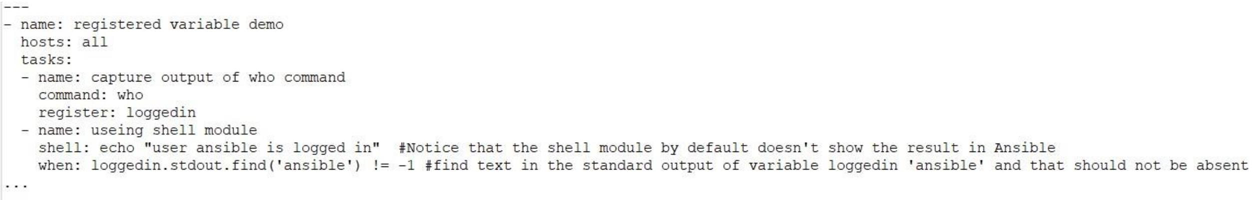
[ansible@controller ~]$ mkdir registervar

[ansible@controller ~]$ cd registervar/

[ansible@controller registervar]$ cp /home/ansible/conditionals/inventory . [ansible@controller registervar]$

vim register.yml

[ansible@controller registervar]$ ansible-playbook register.yml -i inventory



```
---
- name: registered variable demo
  hosts: all
  tasks:
  - name: capture output of who command
    command: who
    register: loggedin
  - name: useing shell module
    shell: echo "user ansible is logged in"  #Notice that the shell module by default doesn't show the result in Ansible
    when: loggedin.stdout.find('ansible') != -1 #find text in the standard output of variable loggedin 'ansible' and that should not be absent
...
```

```
PLAY [registered variable demo]
*****************************************************************************

******************************************

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

[managed2.mekawy.com]

ok: [managed1.mekawy.com] ok:

[controller.mekawy.com]

TASK [capture output of who command]
*****************************************************************************
***********************************

changed: [managed1.mekawy.com] changed:

[managed2.mekawy.com] changed:

[controller.mekawy.com]
TASK [using shell module]
*****************************************************************************
********************************************* changed:

[managed1.mekawy.com]

changed: [managed2.mekawy.com] changed:

[controller.mekawy.com]

PLAY RECAP
*****************************************************************************

****************************************************************
```

```
controller.mekawy.com          : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed1.mekawy.com            : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com            : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

#Basically it didn't change anything apart from setting the registered variable which we do not really see a result. If you wanna see a result, in this case, you want to make it more verbose.

#The reason why you don't see it if you run it without the ansible -v is that the commands that you are running through the shell module do not connect to the same standard output as ansible.

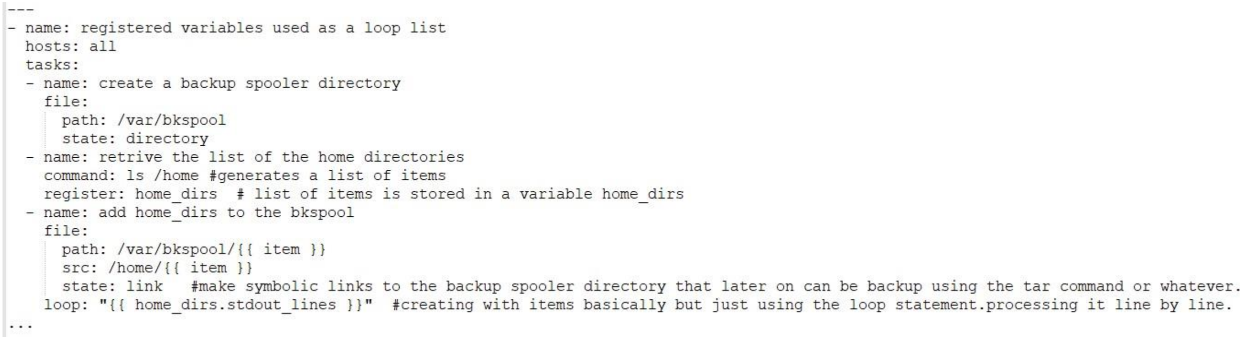
```
[ansible@controller registervar]$ ansible-playbook -v register.yml -i inventory Using

/etc/ansible/ansible.cfg as config file

PLAY [registered variable demo]
********************************************************************************

********************************************

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

[managed2.mekawy.com]

ok: [managed1.mekawy.com] ok:

[controller.mekawy.com]

TASK [capture output of who command]
********************************************************************************

**************************************
```

```
changed: [managed2.mekawy.com] => {"changed": true, "cmd": ["who"], "delta":
"0:00:00.005380", "end": "2021-05-12 02:15:32.887462", "rc": 0, "start": "2021-05-12
02:15:32.882082", "stderr": "", "stderr_lines": [], "stdout": "root              pts/0          2021-05-12 01:38
(192.168.1.4)\nansible pts/1            2021-05-12 02:15 (controller.mekawy.com)",
"stdout_lines": ["root        pts/0        2021-05-12 01:38 (192.168.1.4)","ansible pts/1
2021-05-12 02:15 (controller.mekawy.com)"]}

changed: [managed1.mekawy.com] => {"changed": true, "cmd": ["who"], "delta":
"0:00:00.006088", "end": "2021-05-12 02:15:34.169537", "rc": 0, "start": "2021-05-12
02:15:34.163449", "stderr": "", "stderr_lines": [], "stdout": "root              pts/0        2021-05-12 01:37
(192.168.1.4)\nansible pts/1            2021-05-12 02:15 (controller.mekawy.com)",
"stdout_lines": ["root        pts/0        2021-05-12 01:37 (192.168.1.4)","ansible pts/1
2021-05-12 02:15 (controller.mekawy.com)"]}

changed: [controller.mekawy.com] => {"changed": true, "cmd": ["who"], "delta": "0:00:00.005859", "end":
"2021-05-12 02:15:33.148740", "rc": 0, "start": "2021-05-12
02:15:33.142881", "stderr": "", "stderr_lines": [], "stdout": "ansible pts/0                2021-05-12
01:37 (192.168.1.4)\nansible pts/4             2021-05-12 02:15 (controller.mekawy.com)",
"stdout_lines": ["ansible pts/0            2021-05-12 01:37 (192.168.1.4)", "ansible pts/4
2021-05-12 02:15 (controller.mekawy.com)"]}

TASK [using shell module]
*******************************************************************************
**************************************************

changed: [managed1.mekawy.com] => {"changed": true, "cmd": "echo \"user ansible is logged in\"",
"delta": "0:00:00.004602", "end": "2021-05-12 02:15:35.233090", "rc": 0, "start":
"2021-05-12 02:15:35.228488", "stderr": "", "stderr_lines": [], "stdout": "user ansible is logged in", "stdout_lines":
["user ansible is logged in"]}

changed: [managed2.mekawy.com] => {"changed": true, "cmd": "echo \"user ansible is logged in\"",
"delta": "0:00:00.003582", "end": "2021-05-12 02:15:33.953719", "rc": 0, "start":
"2021-05-12 02:15:33.950137", "stderr": "", "stderr_lines": [], "stdout": "user ansible is logged in", "stdout_lines":
["user ansible is logged in"]}
```

```
changed: [controller.mekawy.com] => {"changed": true, "cmd": "echo \"user ansible is logged in\"", "delta":
"0:00:00.004169", "end": "2021-05-12 02:15:34.314421", "rc": 0, "start":
"2021-05-12 02:15:34.310252", "stderr": "", "stderr_lines": [], "stdout": "user ansible is logged in", "stdout_lines":
["user ansible is logged in"]}

PLAY RECAP
*************************************************************************************
***********************************************************************

controller.mekawy.com        : ok=3    changed=2     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0

managed1.mekawy.com          : ok=3    changed=2     unreachable=0    failed=0      skipped=0
rescued=0     ignored=0

managed2.mekawy.com          : ok=3    changed=2     unreachable=0    failed=0      skipped=0
rescued=0     ignored=0
```

#example2

```
[ansible@controller registervar]$ vim register2.yml
```



```
---
- name: registered variables used as a loop list
  hosts: all
  tasks:
  - name: create a backup spooler directory
    file:
      path: /var/bkspool
      state: directory
  - name: retrive the list of the home directories
    command: ls /home #generates a list of items
    register: home_dirs  # list of items is stored in a variable home_dirs
  - name: add home_dirs to the bkspool
    file:
      path: /var/bkspool/{{ item }}
      src: /home/{{ item }}
      state: link   #make symbolic links to the backup spooler directory that later on can be backup using the tar command or whatever.
    loop: "{{ home_dirs.stdout_lines }}"  #creating with items basically but just using the loop statement.processing it line by line.
...
```

```
[ansible@controller registervar]$ ansible-playbook register2.yml -i inventory PLAY [registered

variables used as a loop list]
*************************************************************************************
************************
```

```
TASK [Gathering Facts]
*****************************************************************************
************************************************* ok:
[managed2.mekawy.com]
ok: [managed1.mekawy.com] ok:
[controller.mekawy.com]

TASK [create a backup spooler directory]
*****************************************************************************
******************************* changed:
[managed2.mekawy.com] changed:
[managed1.mekawy.com] changed:
[controller.mekawy.com]
TASK [retrieve the list of the home directories]
*****************************************************************************
************************ changed:
[managed1.mekawy.com] changed:
[managed2.mekawy.com] changed:
[controller.mekawy.com]
TASK [add home_dirs to the bkspool]
*****************************************************************************
**************************************

changed: [managed2.mekawy.com] => (item=ansible) changed:
[managed1.mekawy.com]      =>      (item=ansible)      changed:
[controller.mekawy.com] => (item=ansible)
```

```
changed:     [managed1.mekawy.com]     =>     (item=marawan)
changed:     [managed2.mekawy.com]     =>     (item=marawan)
changed:     [controller.mekawy.com]     =>     (item=marawan)
changed:    [managed1.mekawy.com]    =>    (item=mmekawy)
changed:    [managed2.mekawy.com]    =>    (item=mmekawy)
changed:[controller.mekawy.com]=>(item=mmekawy) changed:
[managed1.mekawy.com]    =>    (item=mohamed)    changed:
[managed2.mekawy.com]    =>    (item=mohamed)    changed:
[controller.mekawy.com]=>(item=mohamed)

PLAY RECAP
*****************************************************************************
****************************************************************

controller.mekawy.com          : ok=4     changed=3     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0

managed1.mekawy.com            : ok=4     changed=3     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0

managed2.mekawy.com            : ok=4     changed=3     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0
```

1.4 Ansible Handlers

A handler is a task that is going to be executed if another task is successful.

Handlers are conditional tasks that only runs after being notified by another task. Handlers have globally unique name and are triggered after all tasks in the playbook. And that's important to realize that there is an ordering issue with handlers.

Apart from that, a handler is triggered by one or more of the tasks in the playbook, all of its properties are task properties. So a handler basically is defined in the same way as a task is defined.

If you want to trigger a handler, a playbook must have a notify item, which calls the name of the handler. Also, you can have more than one handler called from a task.

Handlers always run in the order in which the handlers section is written and not in the order of how they are called in the plays.

Handlers run after all the other tasks and that definitely is something that you should be realizing. Because when calling them from a task, you might think that they run immediately, but be aware, it's not the case. All of the other tasks in the play will run first.

Handler names must be globally unique. They are defined in the global area which is like configuration variables.

Handlers cannot be included in a playbook. They must be a part of the playbook itself. #example1

Requirements:

create an index.html in the /home/ansible/handlers/ directory on the control hosts. Create a

playbook that copies this file to the web server DocumentRoot

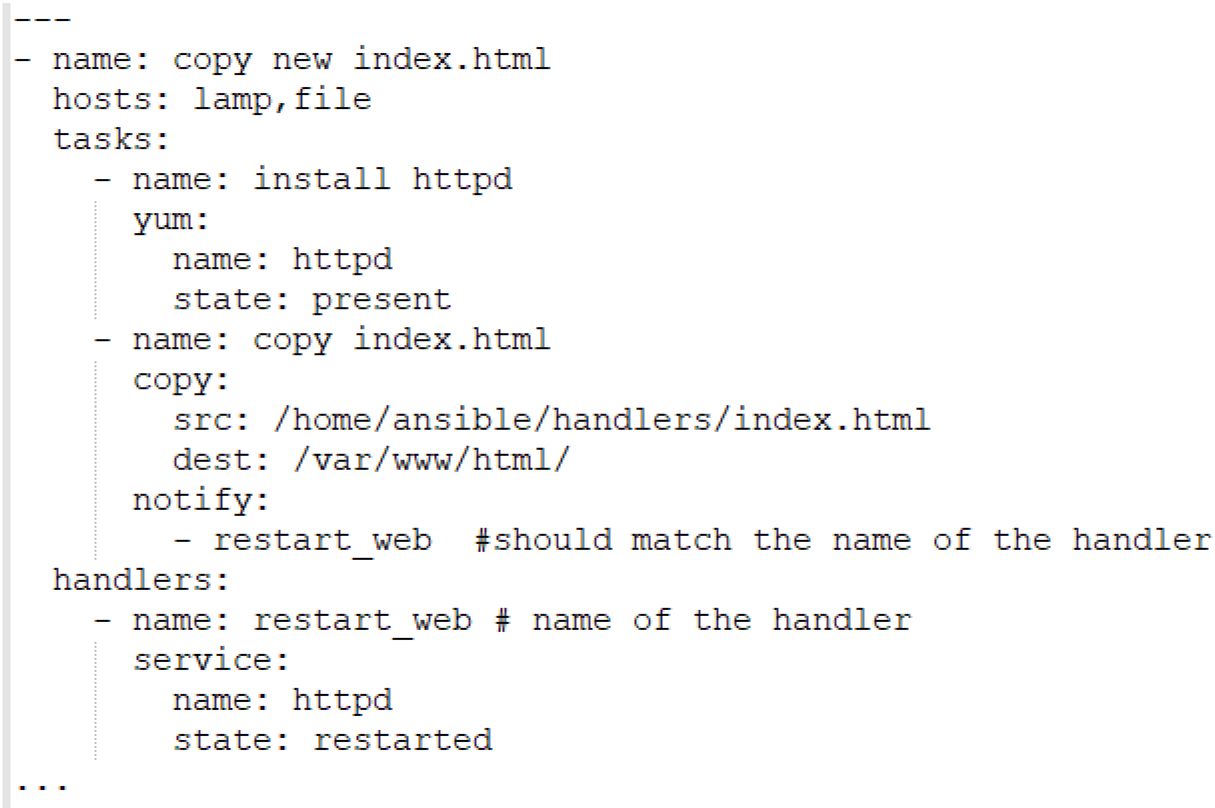
And only if this is successful, restart the Apache process. Solution:

```
[ansible@controller ~]$ mkdir handlers
```

```
[ansible@controller handlers]$ cp /home/ansible/uninstall/inventory . [ansible@controller
handlers]$ cp /home/ansible/uninstall/ansible.cfg . [ansible@controller handlers]$ vim
index.html
```

Test Handlers on Web Server

[ansible@controller ~]$ vim handelers.yml



```
---
- name: copy new index.html
  hosts: lamp,file
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: present
    - name: copy index.html
      copy:
        src: /home/ansible/handlers/index.html
        dest: /var/www/html/
      notify:
        - restart_web  #should match the name of the handler
  handlers:
    - name: restart_web # name of the handler
      service:
        name: httpd
        state: restarted
...
```

```
[ansible@controller handlers]$ ansible-playbook handelers.yml

[WARNING]: While constructing a mapping from /home/ansible/handlers/handelers.yml, line 2, column
3, found a duplicate dict key (tasks). Using last

defined value only.


PLAY [copy new index.html]
*******************************************************************************

*************************************************
```

```
TASK [Gathering Facts]
********************************************************************************
************************************************** ok:
[managed2.mekawy.com]
ok: [managed1.mekawy.com]

TASK [copy index.html]
********************************************************************************
************************************************** ok:
[managed1.mekawy.com]
ok: [managed2.mekawy.com]

PLAY RECAP
********************************************************************************
******************************************************************

managed1.mekawy.com          : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com          : ok=2    changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

[ansible@controller handlers]$ vim handelers [ansible@controller
handlers]$ vim handelers.yml [ansible@controller handlers]$ ansible-
playbook handelers.yml
```

```
PLAY [copy new index.html]
***************************************************************************************
***************************************************

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

[managed2.mekawy.com]

ok: [managed1.mekawy.com]

TASK [install httpd]
***************************************************************************************
********************************************************* changed:

[managed2.mekawy.com]

changed: [managed1.mekawy.com]

TASK [copy index.html]
***************************************************************************************
*******************************************************

changed: [managed2.mekawy.com]

changed: [managed1.mekawy.com]

RUNNING HANDLER [restart_web]
***************************************************************************************
***********************************************

changed: [managed1.mekawy.com]

changed: [managed2.mekawy.com]
```

```
PLAY RECAP
**************************************************************************************

********************************************************************

managed1.mekawy.com          : ok=4    changed=3    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com          : ok=4    changed=3    unreachable=0    failed=0    skipped=0
rescued=0    ignored=
```

1.5 Ansible Tags

If you are looking for a solution to create playbooks that can be used to do many, many things, but in some cases you only want to use specific parts of the playbook, you can use tags.

Tags are used at a resource level to give a name to a specific resource.

while you run the ansible-playbook command, you can use the --tags option to run only resources with a specific tag.

When a task file is included in a playbook, it can be tagged in the include statement. When tagging a

role or inclusion, the tag applies to everything in the roll or the inclusion.

When using tags you can use ansible-playbook--tags 'tagname'.If you use it that way only resources that are marked with those tags will run. So that means that if a resource at the same level does not have a tag, it won't run.

Using --skip-tags 'tagname' that will exclude resources with a specfic tag.

Special tag "always" can be used to make sure a resource is always executed no matter what. Unless it's specifically excluded with --skip-tags.

Tags can be included in yml file

The --tags option when you used with the ansible-playbook command can take three specific arguments.

1."tagged" will run any tagged resources 2."untagged"

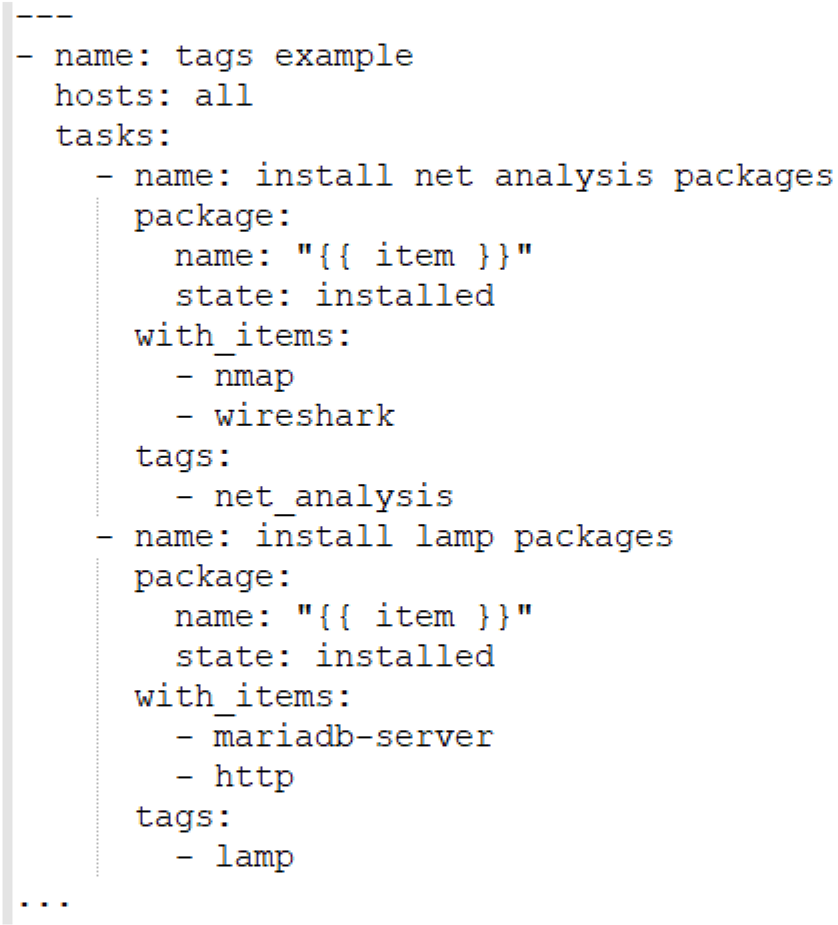
will exclude all tagged resources.

3."all" as well which will run all tasks (the default behavior and it also happens if no tags have been specified)

You should notice that it is possible in your design of the ansible environment to use tags to give labels to a specific place, but instead of using tags you could also choose to use a different solution and that is just to write different playbooks. Depends on what you want. If you want to put it all in one big playbook use stacks. If you want to use separate playbooks don't use tags.

#example

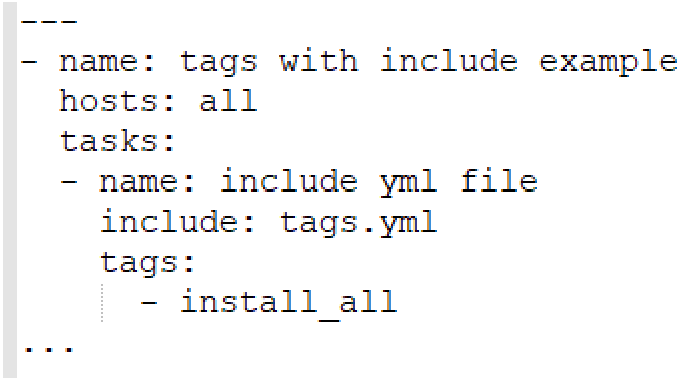
```
[ansible@controller ~]$ mkdir tags

[ansible@controller ~]$ cp /home/ansible/uninstall/inventory .

[ansible@controller ~]$ cp /home/ansible/uninstall/ansible.cfg .

[ansible@controller tags]$cp /home/ansible/inventory .

[ansible@controller tags]$cp /home/ansible/ansible.cfg .

[ansible@controller tags]$vim tags.yml
```



```
---
- name: tags example
  hosts: all
  tasks:
    - name: install net analysis packages
      package:
        name: "{{ item }}"
        state: installed
      with_items:
        - nmap
        - wireshark
      tags:
        - net_analysis
    - name: install lamp packages
      package:
        name: "{{ item }}"
        state: installed
      with_items:
        - mariadb-server
        - http
      tags:
        - lamp
...
```

```
[ansible@controller tags]$ ansible-playbook tags.yml --tags 'net_analysis'

PLAY [tags example]
***************************************************************************

******************************************************

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

[managed1.mekawy.com]

ok: [managed2.mekawy.com] ok:

[controller.mekawy.com]
```

```
TASK [install net analysis packages]
*************************************************************************************
*****************************************

changed: [managed2.mekawy.com] => (item=nmap) changed:
[managed1.mekawy.com] => (item=nmap) changed:
[managed2.mekawy.com] => (item=wireshark) changed:
[managed1.mekawy.com] => (item=wireshark) changed:
[controller.mekawy.com] => (item=nmap) changed:
[controller.mekawy.com] => (item=wireshark)

PLAY RECAP
*************************************************************************************
***********************************************************************

controller.mekawy.com      : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed1.mekawy.com        : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com        : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

[ansible@controller tags]$ vim tags_include.yml
```



```
---
- name: tags with include example
  hosts: all
  tasks:
  - name: include yml file
    include: tags.yml
    tags:
      - install_all
...
```

1.6 Dealing with Errors

When working with conditionals, you need to be able to deal with error situations.

The default behavior, if that's a failure, then the play execution stops but exceptions are possible.One way to define an exception is to use "ignore_errors: yes" to continue after failure.

""force_handlers: yes" If a task in a play fails, then no handlers will be executed.and that can be very confusing because the task that fails within the play may completely not be related to the handle that you want to run.to avoid that, now you can use force_handlers: yes which will execute handlers that were triggered by a previous action anyway.force_handlers: yes needs to be in the header of the task.

"failed_when: " can be used if it's not completely clear when a command is considered failed.And that is useful for commands that produce a specific output.

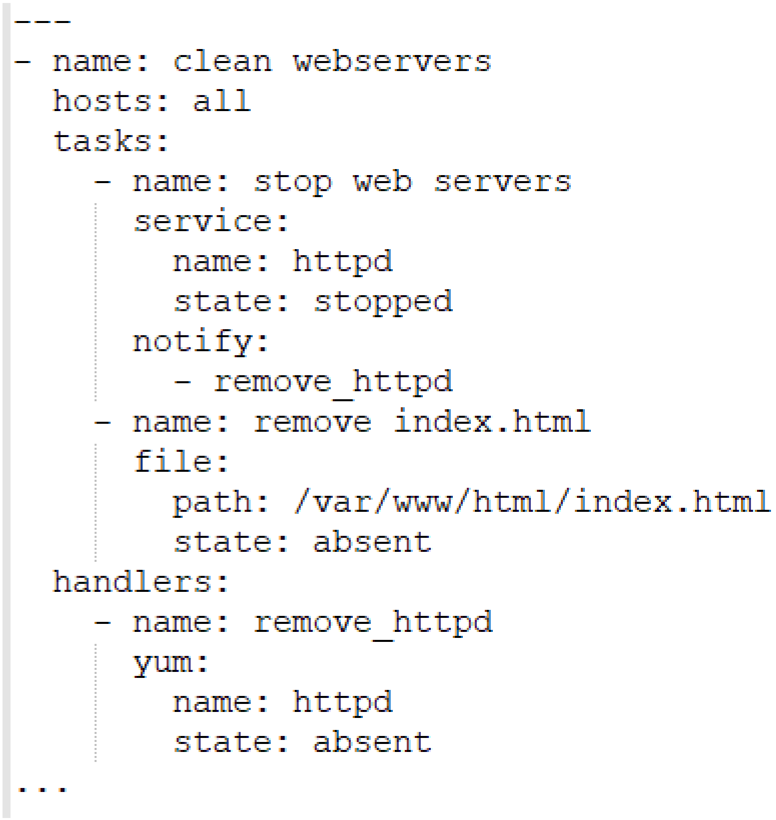
"changed_when: false" If a module thinks it changed the state of the affected machine, it will report the changed status. And that's not always desired and for that reason may be overwritten or it may be further specified. And as a result, the module will not generate a changed state.

#example

[ansible@controller ~]$ cd errors/

[ansible@controller errors]$ cp /home/ansible/inventory .

```
[ansible@controller errors]$ cp /home/ansible/ansible.cfg . [ansible@controller errors]$ cp
/home/ansible/handlers/index.html . [ansible@controller errors]$ ls
ansible.cfg index.html inventory [ansible@controller
errors]$vimcleanweb.yml
```



```
---
- name: clean webservers
  hosts: all
  tasks:
    - name: stop web servers
      service:
        name: httpd
        state: stopped
      notify:
        - remove_httpd
    - name: remove index.html
      file:
        path: /var/www/html/index.html
        state: absent
  handlers:
    - name: remove_httpd
      yum:
        name: httpd
        state: absent
...
```

```
[ansible@controller errors]$ ansible-playbook cleanweb.yml PLAY

[clean webservers]
***************************************************************************
*************************************************

TASK [Gathering Facts]
***************************************************************************
**************************************************
```

```
ok: [managed2.mekawy.com] ok:
[managed1.mekawy.com]       ok:
[controller.mekawy.com]

TASK [stop web servers]
***************************************************************************************
****************************************************** ok:
[controller.mekawy.com]
changed: [managed2.mekawy.com] changed:
[managed1.mekawy.com]

TASK [remove index.html]
***************************************************************************************
****************************************************
changed: [managed1.mekawy.com]
changed: [controller.mekawy.com] changed:
[managed2.mekawy.com]

RUNNING HANDLER [remove_httpd]
***************************************************************************************
**********************************************
changed: [managed2.mekawy.com]
changed: [managed1.mekawy.com]

PLAY RECAP
***************************************************************************************
************************************************************************
```

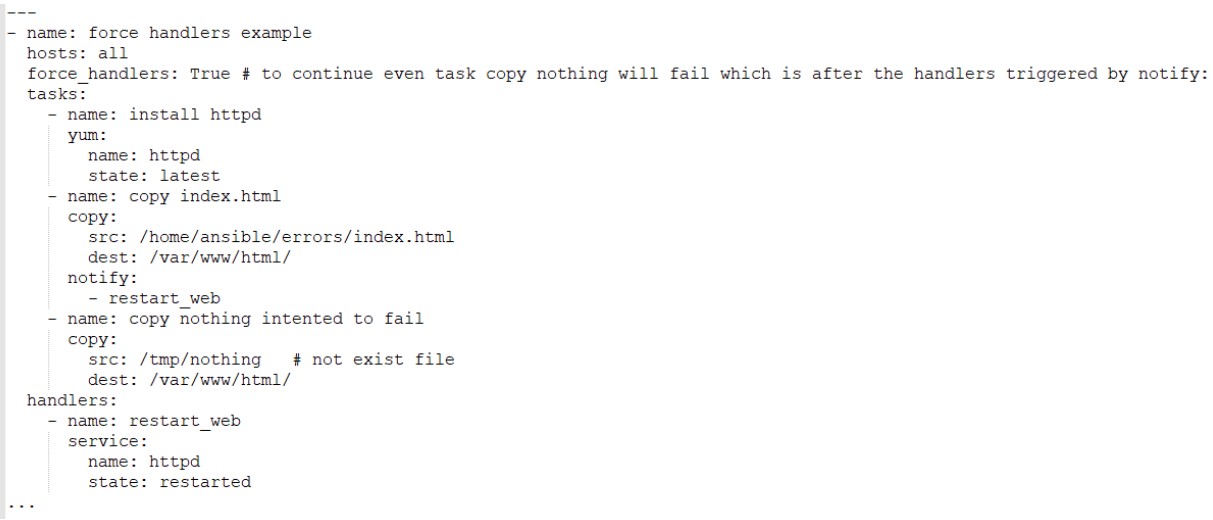
controller.mekawy.com : ok=3 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

managed1.mekawy.com : ok=4 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

managed2.mekawy.com : ok=4 changed=3 unreachable=0 failed=0 skipped=0

rescued=0 ignored=0

[ansible@controller errors]$ vim force_handlers.yml



```
---
- name: force handlers example
  hosts: all
  force_handlers: True # to continue even task copy nothing will fail which is after the handlers triggered by notify:
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest
    - name: copy index.html
      copy:
        src: /home/ansible/errors/index.html
        dest: /var/www/html/
      notify:
        - restart_web
    - name: copy nothing intented to fail
      copy:
        src: /tmp/nothing   # not exist file
        dest: /var/www/html/
  handlers:
    - name: restart_web
      service:
        name: httpd
        state: restarted
...
```

[ansible@controller errors]$ ansible-playbook force_handlers.yml PLAY

[force handlers example]

****

****

TASK [Gathering Facts]

****

** ok:

[managed1.mekawy.com]

ok: [managed2.mekawy.com]

```
ok: [controller.mekawy.com]

TASK [install httpd]
****************************************************************************
*************************************************** ok:

[controller.mekawy.com]

changed: [managed2.mekawy.com]

changed: [managed1.mekawy.com] TASK

[copy index.html]
****************************************************************************
************************************************* changed:

[managed1.mekawy.com]

changed: [managed2.mekawy.com] changed:

[controller.mekawy.com]

TASK [copy nothing intended to fail]
****************************************************************************

*************************************

An exception occurred during task execution. To see the full traceback, use -vvv. The error was: If you are
using a module and expect the file to exist on the remote, see the remote_src option

fatal: [managed2.mekawy.com]: FAILED! => {"changed": false, "msg": "Could not find or access
'/tmp/nothing' on the Ansible Controller.\nIf you are using a module and expect the file to exist on the
remote, see the remote_src option"}

An exception occurred during task execution. To see the full traceback, use -vvv. The error was: If you are
using a module and expect the file to exist on the remote, see the remote_src option
```

```
fatal: [managed1.mekawy.com]: FAILED! => {"changed": false, "msg": "Could not find or access
'/tmp/nothing' on the Ansible Controller.\nIf you are using a module and expect the file to exist on the
remote, see the remote_src option"}

An exception occurred during task execution. To see the full traceback, use -vvv. The error was: If you are
using a module and expect the file to exist on the remote, see the remote_src option

fatal: [controller.mekawy.com]: FAILED! => {"changed": false, "msg": "Could not find or access
'/tmp/nothing' on the Ansible Controller.\nIf you are using a module and expect the file to exist on the
remote, see the remote_src option"}

RUNNING HANDLER [restart_web]
***************************************************************************
*****************************************

changed: [managed2.mekawy.com]

changed: [managed1.mekawy.com] changed:

[controller.mekawy.com]

PLAY RECAP
***************************************************************************

**************************************************************

controller.mekawy.com        : ok=4    changed=2    unreachable=0     failed=1    skipped=0
rescued=0     ignored=0

managed1.mekawy.com          : ok=4    changed=3    unreachable=0     failed=1    skipped=0
rescued=0     ignored=0

managed2.mekawy.com          : ok=4    changed=3    unreachable=0     failed=1    skipped=0
rescued=0     ignored=0
```

1.7 Ansible Blocks

A block is used to logically group tasks

Blocks are useful for error handling and also for "when" statements. One statement can be applied to the entire block so that it affects all the tasks within the block.

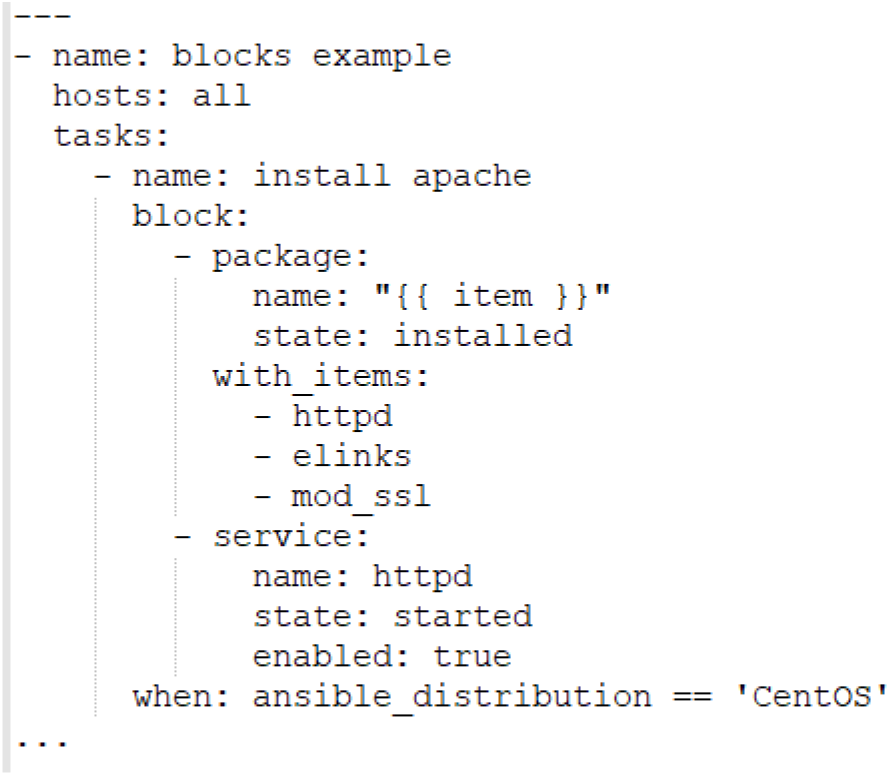
Blocks also allow for error handling. It's combined with "rescue" and "always" statements. If a task fails,

the tasks in the "rescue" task are executed for recovery.

Tasks in "always" will run, regardless of the success or failure of tasks that are defined in block and in rescue.

#example 1

```
[ansible@controller ~]$ mkdir blocks

[ansible@controller ~]$ cd blocks/

[ansible@controller blocks]$ cp /home/ansible/inventory .

[ansible@controller blocks]$ cp /home/ansible/ansible.cfg .

[ansible@controller blocks]$ vim blocks.yml
```



```
---
- name: blocks example
  hosts: all
  tasks:
    - name: install apache
      block:
        - package:
            name: "{{ item }}"
            state: installed
          with_items:
            - httpd
            - elinks
            - mod_ssl
        - service:
            name: httpd
            state: started
            enabled: true
      when: ansible_distribution == 'CentOS'
...
```

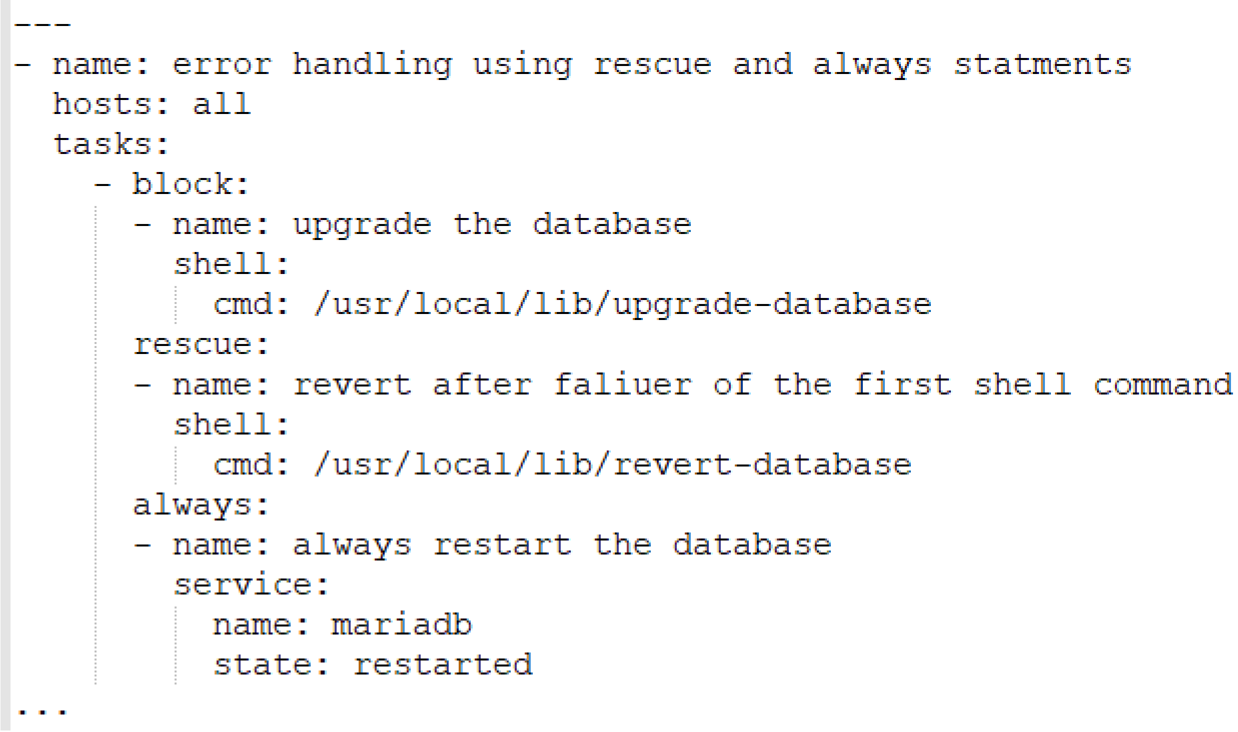
```
[ansible@controller blocks]$ ansible-playbook blocks.yml


PLAY [blocks example]
*****************************************************************************
*****************************************************


TASK [Gathering Facts]
*****************************************************************************
**************************************************** ok:
[managed1.mekawy.com]
ok: [managed2.mekawy.com] ok:
[controller.mekawy.com]
```

```
TASK [package]
**************************************************************************************
*************************************************************** ok:
[managed2.mekawy.com] => (item=httpd)
ok: [managed1.mekawy.com] => (item=httpd) ok:
[controller.mekawy.com] => (item=httpd)
changed: [managed2.mekawy.com] => (item=elinks) changed:
[controller.mekawy.com] => (item=elinks) changed:
[managed1.mekawy.com] => (item=elinks) changed:
[managed2.mekawy.com] => (item=mod_ssl) changed:
[controller.mekawy.com] => (item=mod_ssl) changed:
[managed1.mekawy.com] => (item=mod_ssl)

TASK [service]
**************************************************************************************
****************************************************************
changed: [managed2.mekawy.com]
changed: [managed1.mekawy.com] changed:
[controller.mekawy.com]
```

```
PLAY RECAP
****************************************************************************

***************************************************************

controller.mekawy.com       : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed1.mekawy.com         : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com         : ok=3    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

#example2

```
[ansible@controller blocks]$ vim rescue.yml
```



```
---
- name: error handling using rescue and always statments
  hosts: all
  tasks:
    - block:
      - name: upgrade the database
        shell:
          cmd: /usr/local/lib/upgrade-database
      rescue:
      - name: revert after faliuer of the first shell command
        shell:
          cmd: /usr/local/lib/revert-database
      always:
      - name: always restart the database
        service:
          name: mariadb
          state: restarted
...
```

Big Lab

Requirements:

1- create a playbook that is installing web services in Ubuntu as well as CentOS. You need to

have the:

1.Apache web server 2.The

vsftpd FTP service

3.The mariaDB database services on both platforms.

Notice, by the way, that on Ubuntu, you'll have to use MySQL because it doesn't provide any mariaDB packages.

2- Use custom facts on the local machines to define the names of the packages, and use with_items to process these custom facts.

3- Manage firewalling rules on both platforms.

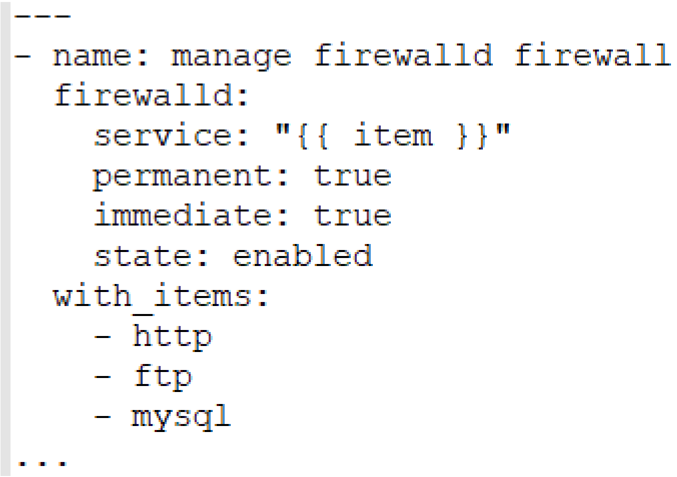
4- Use a handler to write the message, installation succeeded, if all packages installed successfully.

Solution:

#do not forget to add ubuntu.mekawy.com in /etc/hosts file and configure SSH key and create ansible user with sudoers delegation.

```
[ansible@controller ~]$ mkdir biglab3

[ansible@controller ~]$ cd biglab3/

[ansible@controllerbiglab3]$cp/home/ansible/inventory.

[ansible@controller biglab3]$ cp /home/ansible/ansible.cfg .

[ansible@controller biglab3]$ vim inventory
```

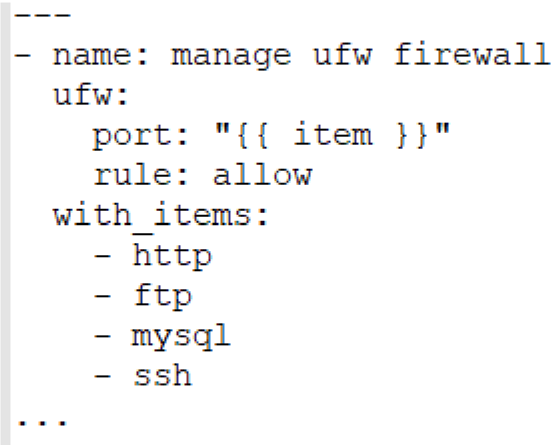
```
[all] controller.mekawy.com
managed1.mekawy.com
managed2.mekawy.com
ubuntu.mekawy.com [lamp]
managed2.mekawy.com [file]
managed1.mekawy.com
[ansible@controller biglab3]$ mkdir tasks
[ansible@controller biglab3]$ mkdir vars
[ansible@controller biglab3]$ cd tasks/
[ansible@controller tasks]$ sudo firewall-cmd --list-services dhcpv6-
client ssh
[ansible@controller tasks]$ sudo firewall-cmd --get-services | grep http [ansible@controller tasks]$ sudo
firewall-cmd --get-services | grep ftp [ansible@controller tasks]$ sudo firewall-cmd --get-services | grep
mysql [ansible@controller tasks]$ vim firewall_redhat.yml
```



```
---
- name: manage firewalld firewall
  firewalld:
    service: "{{ item }}"
    permanent: true
    immediate: true
    state: enabled
  with_items:
    - http
    - ftp
    - mysql
...
```

[ansible@controller tasks]$ cp firewall_redhat.yml firewall_ubuntu.yml [ansible@controller

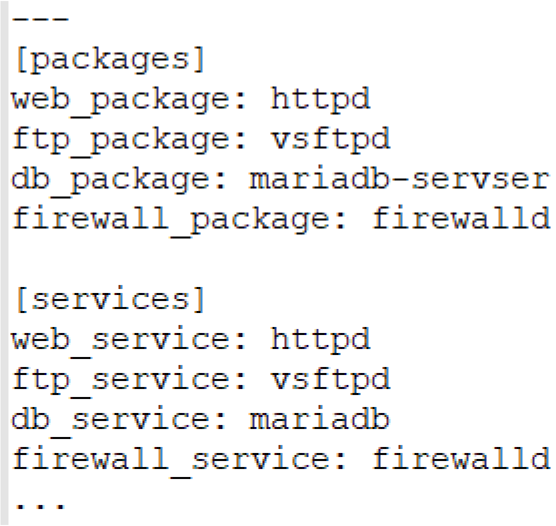
tasks]$ vim firewall_ubuntu.yml



```
---
- name: manage ufw firewall
  ufw:
    port: "{{ item }}"
    rule: allow
  with_items:
    - http
    - ftp
    - mysql
    - ssh
...
```

[ansible@controller tasks]$ cd ../vars/ [ansible@controller vars]$

vim vars_redhat.yml

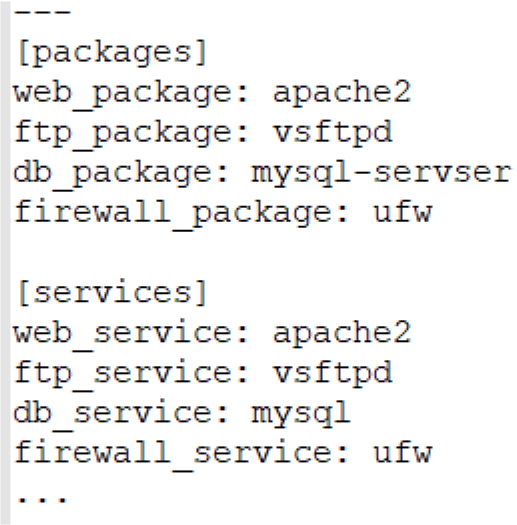


```
---
[packages]
web_package: httpd
ftp_package: vsftpd
db_package: mariadb-servser
firewall_package: firewalld

[services]
web_service: httpd
ftp_service: vsftpd
db_service: mariadb
firewall_service: firewalld
...
```

[ansible@controller vars]$ cp vars_redhat.yml vars_ubuntu.yml

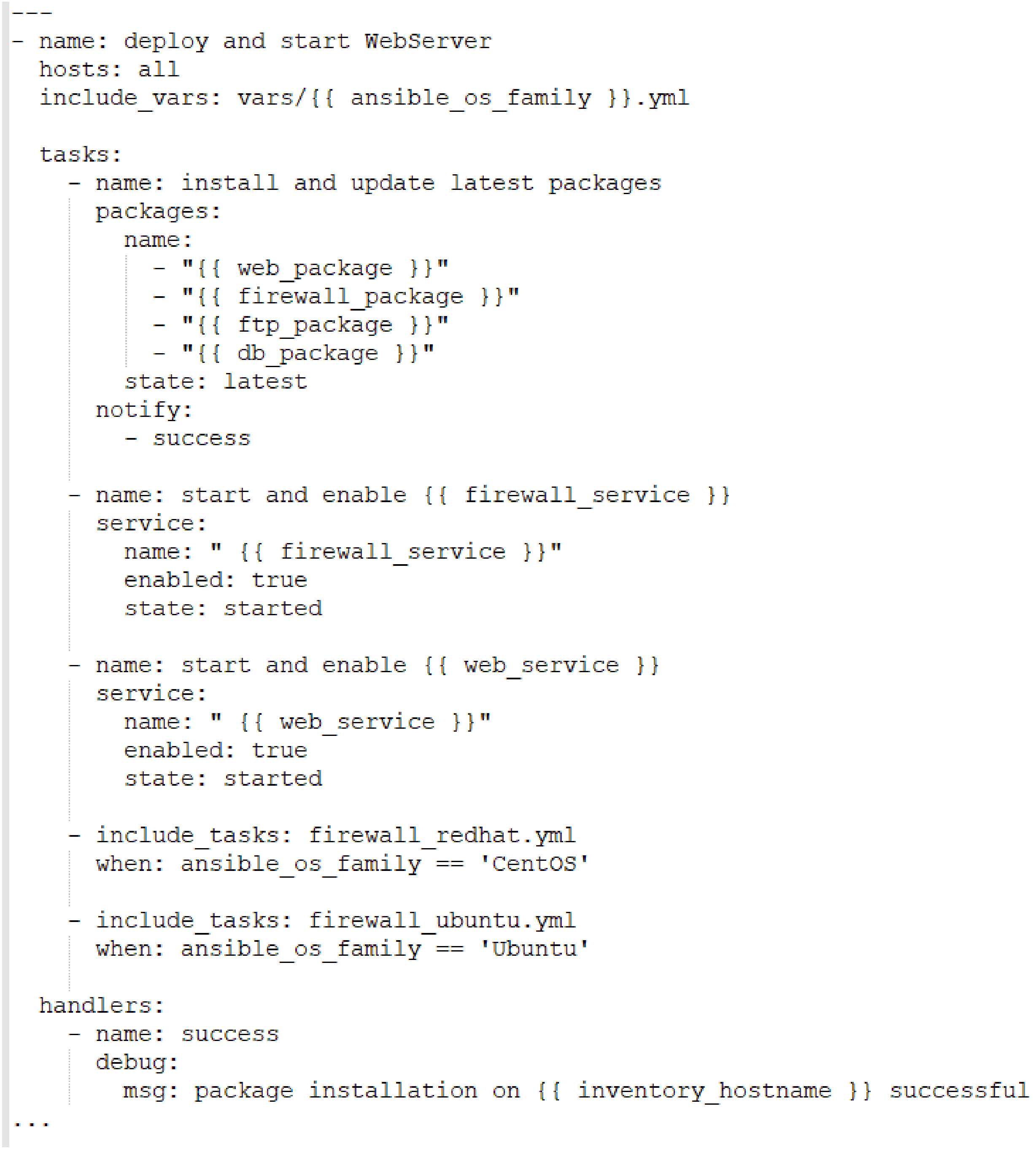
[ansible@controller vars]$ vim vars_ubuntu.yml

[ansible@controller vars]$ cd ..



```
---
[packages]
web_package: apache2
ftp_package: vsftpd
db_package: mysql-servser
firewall_package: ufw

[services]
web_service: apache2
ftp_service: vsftpd
db_service: mysql
firewall_service: ufw
...
```

[ansible@controller biglab3]$ vim mainplaybook.yml



```
---
- name: deploy and start WebServer
  hosts: all
  include_vars: vars/{{ ansible_os_family }}.yml

  tasks:
    - name: install and update latest packages
      packages:
        name:
          - "{{ web_package }}"
          - "{{ firewall_package }}"
          - "{{ ftp_package }}"
          - "{{ db_package }}"
        state: latest
      notify:
        - success

    - name: start and enable {{ firewall_service }}
      service:
        name: " {{ firewall_service }}"
        enabled: true
        state: started

    - name: start and enable {{ web_service }}
      service:
        name: " {{ web_service }}"
        enabled: true
        state: started

    - include_tasks: firewall_redhat.yml
      when: ansible_os_family == 'CentOS'

    - include_tasks: firewall_ubuntu.yml
      when: ansible_os_family == 'Ubuntu'

  handlers:
    - name: success
      debug:
        msg: package installation on {{ inventory_hostname }} successful
...
```

2- Ansible Roles

A role is a structure that is applied to more advanced Ansible projects. Working with roles makes it easier to standardize the task you want to perform in Ansible.

You can of course use includes any way you want, but it is better if you do it in a structured way, because if you do it in a structured way then everybody knows where to find what, and that is exactly the idea behind Ansible roles.

2.1 Organizing Ansible contents

1. Even simple projects should have their own directory structure, to make sure that every project is very clearly and distinctively defined. And within the directory structure you will have an own Ansible configuration, an inventory, as well as playbooks.

2. If a project grows bigger, variable files, as well as includes, may be used.

3. roles can be used to standardize and easily re-use specific parts of Ansible.You should consider a role a complete project that is dedicated to a specific task that is going to be included from the main playbook.

4. Ansible best practices:
https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html

Directory Layout: https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#directory-layout

Alternative Directory Layout:
https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#alternative

-directory-layout

Use Dynamic Inventory With Clouds:
https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#use-dyna mic-inventory-with-clouds

How to Differentiate Staging vs Production:
https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#how-to-dif ferentiate-staging-vs-production

Group And Host Variables:
https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#group-and

-host-variables

2.2 Roles Structure

1. Ansible roles provide uniform ways to load tasks, handlers and variables from external files, and a role typically corresponds to the type of service that is offered (web, database,docker containers, etc.)

2. roles, you will use a very specific directory structure, with locations for default values, for handlers, for tasks, for templates and for variables, and by using this structure.

3. while working with roles, generic profiles are defined in a role. For specific groups of servers, or servers, specific playbooks may be created to include one or more roles, and to manage what should happen, default variables are set in the role, which can be overwritten at playbook level. You should notice that default variables can be overwritten anywhere.
That means at a playbook level, but also for example when using inventory variables. The idea is just if you define a default variable again at a lower level, the lower level will always win.

4. To make working with roles easier, community roles can be downloaded from Ansible Galaxy. It's like an open source repository of all the roles that are available.

5. Roles are defined in a roles directory, which is created in the current project's directory.

6. An important part of roles are the Jinja2 templates, as they allow working with flexible parameters that are set as variables or facts, and make sure that host's specific configuration is pushed to the specific host.

2.3 Roles directory structure contents

default: which contains a main.yml with default values for variables files: which

contains static files that are referenced by role tasks.

handlers: which is used to store handlers in case the main.yml is calling any handlers.

meta: which contains a main.yml with information about the role including author, license, platforms and dependencies.

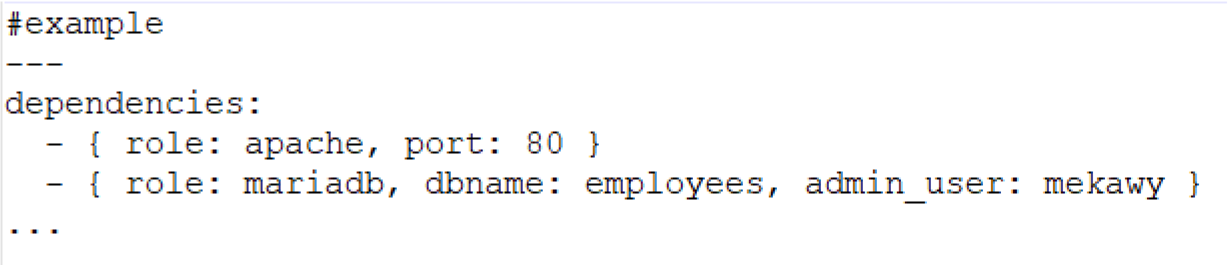
tasks: with a main.yml with task definitions

vars: with a main.yml that includes role variable definitions.

2.4 Role variables & Dependencies

You will define them in a vars/main.yml. These variables have a high priority and cannot be overwritten by inventory variables. Default variables can be defined in defaults/main.yml, and they have the lowest precedence, which means that they can be overwritten at any level. You should use default variables only if you intend to have the variable overwritten somewhere else. Overriding variables is common, as roles are used as templates where specific variables may be overridden in specific playbooks.

Roles can have dependencies, so roles may include other roles as well, and the dependencies are written to meta/main.yml within the role.All of the dependencies that should be met for this role to function.

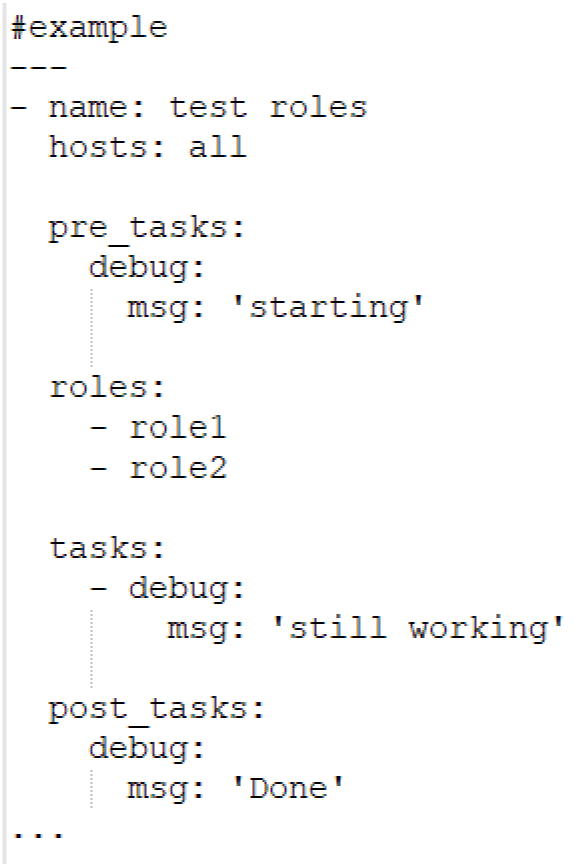


```
#example
---
dependencies:
  - { role: apache, port: 80 }
  - { role: mariadb, dbname: employees, admin_user: mekawy }
...
```

2.5 Order of execution

Normally tasks in a role execute before the tasks of a playbook that is using them.So the tasks in a role really define that everything is there at the moment that you start running the playbook.

There are two solutions that you can use to override that: 1.pre-tasks: are performed before the roles are applied 2.post-tasks: are performed after completing all the roles.



```
#example
---
- name: test roles
  hosts: all

  pre_tasks:
    debug:
      msg: 'starting'

  roles:
    - role1
    - role2

  tasks:
    - debug:
        msg: 'still working'

  post_tasks:
    debug:
      msg: 'Done'
...
```

2.6 Creating Roles

Creating roles involves a couple of steps. 1.First you need to create the role structure. 2.Then you define the role content.

3.And next you use the role in a playbook.

Notice::

Use the ansible-galaxy --offline utility to automate creating the role directory structure

Eachrolewillhaveitsowndirectorywithspecificsubdirectoriesthatexistina~/roles directory and not in specific project directories.

Subdirectories that are not used may be empty.

2.7 Roles locations

Ansible will look for roles in different locations:

1. as a default, Ansible will use the current project directory.

2. Alternatively,itwilllookinthehomedirectory~/.ansible/roles[Ansibleroles.Ifyouwork with Ansible Galaxy, as we will discuss further in this course, you will notice that Ansible Galaxy will put roles automatically in there.]

3. Next, it look in etc/ansible/roles

4. finally, it will look in user/share/ansible/roles

And in case of conflict, the most specific location wins. Notice::

You might want to consider putting your roles somewhere centrally. There's something to say for that. Because roles may be used by multiple playbooks. So if you put your roles in etc/ansible/roles, for example, you can work with links in the Linux file system to make sure that the role is available in the project directory as well.

#roles lab

#the role structure for the motd role. Which is a role that we are going to explore as an example in this lesson. So motd will be the name of the role, and then we have the defaults, the files, the handlers, the tasks, and the templates. So you can see the default has a main.yml, tasks has a main.yml, templates has a motd.j2 because templates is for j2 templates. Files and handlers is not used in this role, so they are empty

```
[ansible@controller ~]$ mkdir roles-lab

[ansible@controller roles-lab]$ cp /home/ansible/inventory .

[ansible@controller roles-lab]$ cp /home/ansible/ansible.cfg .

[ansible@controller roles-lab]$ vim motd-role.yml [ansible@controller

roles-lab]$ mkdir roles [ansible@controller roles-lab]$ cd roles/

[ansible@controller roles]$ mkdir motd

[ansible@controller roles]$ cd motd/

[ansible@controller motd]$ mkdir defaults files meta tasks templates tests vars [ansible@controller motd]$

vim README.md

Test roles [ansible@controller

motd]$ ls

defaults files meta README.md tasks templates tests vars

[ansible@controller roles-lab]$ pwd

/home/ansible/roles-lab [ansible@controller

roles-lab]$ tree

.

├── ansible.cfg

├── inventory

├── motd-role.yml
```

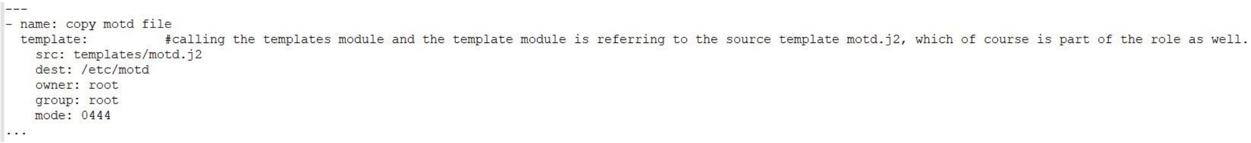
```
└── roles
    └── motd
        ├── defaults
        ├── files
        ├── meta
        ├── README.md
        ├── tasks
        ├── templates
        ├── tests
        └── vars
```

9 directories, 4 files

[ansible@controller roles-lab]$ cd roles/motd/tasks/

#start by defining the modules to call in tasks main.yml. So in the case of the motd role, it may be used to copy a template file over to the managedhosts.

[ansible@controller tasks]$ vim main.yml



```
---
- name: copy motd file
  template:         #calling the templates module and the template module is referring to the source template motd.j2, which of course is part of the role as well.
    src: templates/motd.j2
    dest: /etc/motd
    owner: root
    group: root
    mode: 0444
...
```

#The next phase would be to define the motd.j2 template. So you would put that in roles/motd/templates/motd.j2.

#Notice that in templates, you can use Ansible facts as well as variables. If default variables must be used, then you would set them in roles motd defaults main.yml so there may be a dependency here if it's not an Ansible fact.

#Notice, by the way, that this today would not be completely dynamic, but it will be substituted by the date when the target file is generated.

notice that in this template, we are using two Ansible facts and one variable, which is system_manager. And this variable needs to be defined somewhere else.

```
[ansible@controller tasks]$ cd ../templates/

[ansible@controller templates]$ vim motd.j2

Welcome to {{ ansible_hostname }}.

Today is {{ ansible_date_time.date }}.

Access to this system is for authorized users only Contact{{

system_manager}}formoreinformation.
```

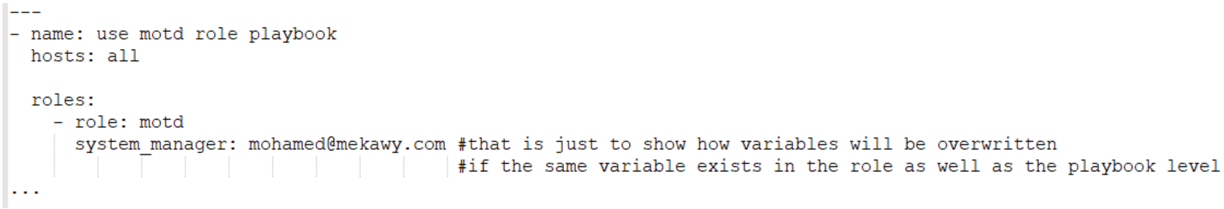
#Define defaults variables, system_manager variable that would be in role/motd/defaults/main.yml where we have system_manager set to mekawy@mekawy.com

```
[ansible@controller templates]$ cd ../defaults/ [ansible@controller

defaults]$ vim main.yml

---

system_manager: mekawy@mekawy.com

...

[ansible@controller defaults]$ cd ../../..

[ansible@controller roles-lab]$ tree
```

```
.
├── ansible.cfg
├── inventory
├── motd-role.yml
└── roles
    └── motd
        ├── defaults
        │   └── main.yml
        ├── files
        ├── meta
        ├── README.md
        ├── tasks
        │   └── main.yml
        ├── templates
        │   └── motd.j2
        ├── tests
        └── vars

9 directories, 7 files
```

#Define main playbook

[ansible@controller roles-lab]$ vim motd-role.yml



```
---
- name: use motd role playbook
  hosts: all

  roles:
    - role: motd
      system_manager: mohamed@mekawy.com #that is just to show how variables will be overwritten
                                         #if the same variable exists in the role as well as the playbook level
...
```

[ansible@controller roles-lab]$ ansible-playbook motd-role.yml PLAY [use

motd role playbook]
****

****

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

[managed1.mekawy.com]

ok: [managed2.mekawy.com] ok:

[controller.mekawy.com]

TASK [copy motd file]
****
** changed:

[managed1.mekawy.com]

changed: [managed2.mekawy.com] changed:

[controller.mekawy.com]

PLAY RECAP
****

****

```
controller.mekawy.com        : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed1.mekawy.com          : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0


managed2.mekawy.com          : ok=2    changed=1    unreachable=0    failed=0    skipped=0

rescued=0    ignored=0
```

```
[ansible@controller roles-lab]$ ssh ansible@managed1.mekawy.com Last
login: Sun May 16 00:08:35 2021 from controller.mekawy.com Welcome to
managed1.

Today is 2021-05-16.

Access to this system is for authorized users only Contact
mohamed@mekawy.com for more information.
```

2.8 Ansible Galaxy

Ansible Galaxy, which is on galaxy.ansible.com, is the community resource for getting and publishing roles.

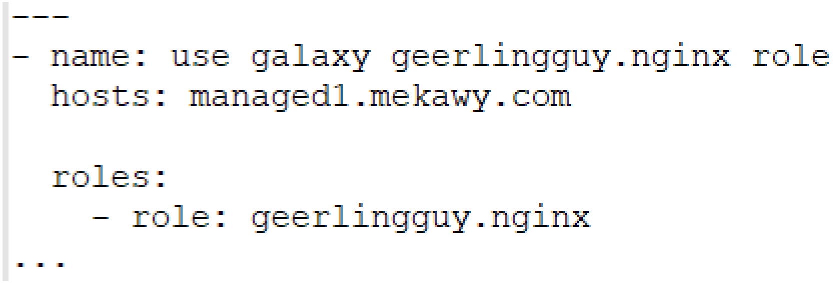
https://galaxy.ansible.com/

#example https://galaxy.ansible.com/geerlingguy/nginx

```
[ansible@controller ~]$ ansible-galaxy install geerlingguy.nginx
- downloading role 'nginx', owned by geerlingguy
- downloading role from
https://github.com/geerlingguy/ansible-role-nginx/archive/3.0.0.tar.gz

- extracting geerlingguy.nginx to /home/ansible/.ansible/roles/geerlingguy.nginx

- geerlingguy.nginx (3.0.0) was installed successfully


[ansible@controller roles]$ cd /home/ansible/.ansible/roles/geerlingguy.nginx [ansible@controller

geerlingguy.nginx]$ tree

.

├── defaults

│ └── main.yml

├── handlers

│ └── main.yml

├── LICENSE

├── meta

│ └── main.yml

├── molecule

│ └── default

│       ├── converge.yml

│       └── molecule.yml

├── README.md

├── tasks
```

```
│ ├── main.yml
│ ├── setup-Archlinux.yml
│ ├── setup-Debian.yml
│ ├── setup-FreeBSD.yml
│ ├── setup-OpenBSD.yml
│  ├──setup-RedHat.yml
│  ├── setup-Ubuntu.yml
│ └── vhosts.yml
├── templates
│  ├──nginx.conf.j2
│  ├── nginx.repo.j2
│ └── vhost.j2
└── vars
   ├── Archlinux.yml
   ├── Debian.yml
   ├── FreeBSD.yml
   ├── OpenBSD.yml
   └── RedHat.yml 8
directories, 23 files
[ansible@controller geerlingguy.nginx]$ cd ~
[ansible@controller ~]$ cd roles-lab/
```

```
[ansible@controller roles-lab]$ vim nginx-role-geerlingguy.yml
```



```
---
- name: use galaxy geerlingguy.nginx role
  hosts: managed1.mekawy.com

  roles:
    - role: geerlingguy.nginx
...
```

```
[ansible@controller roles-lab]$ ansible-playbook nginx-role-geerlingguy.yml

PLAY [use galaxy geerlingguy.nginx role] **************************************** TASK

[Gathering Facts] *********************************************************** ok:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Include OS-specific variables.] ********************** ok:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Define nginx_user.] *********************************** ok:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : include_tasks] ****************************************

included: /home/ansible/.ansible/roles/geerlingguy.nginx/tasks/setup-RedHat.yml for
managed1.mekawy.com

TASK [geerlingguy.nginx : Enable nginx repo.] *********************************** changed:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Ensure nginx is installed.] ************************** changed:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : include_tasks] ****************************************
```

```
skipping:[managed1.mekawy.com]

TASK [geerlingguy.nginx : include_tasks] **************************************** skipping:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : include_tasks] **************************************** skipping:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : include_tasks] **************************************** skipping:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : include_tasks] **************************************** skipping:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Remove default nginx vhost config file (if configured).] *** skipping:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Ensure nginx_vhost_path exists.] ********************* ok:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Add managed vhost config files.] ********************* TASK

[geerlingguy.nginx : Remove managed vhost config files.] ****************** TASK

[geerlingguy.nginx : Remove legacy vhosts.conf file.] ********************* ok:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Copy nginx configuration in place.] ****************** changed:

[managed1.mekawy.com]

TASK [geerlingguy.nginx : Ensure nginx service is running as configured.] ******
```

```
fatal: [managed1.mekawy.com]: FAILED! => {"changed": false, "msg": "Unable to start service nginx: Job for
nginx.service failed because the control process exited with error code. See \"systemctl status
nginx.service\" and \"journalctl -xe\" for details.\n"}

RUNNING HANDLER [geerlingguy.nginx : reload nginx] *****************************

PLAY RECAP
*******************************************************************

managed1.mekawy.com        : ok=9    changed=3    unreachable=0    failed=1    skipped=8
rescued=0    ignored=0
```

2.8.1 Ansible Galaxy CLI Utility

Ansible Galaxy CLI Utility is a very convenient utility. And because it allows you to directly interface with Ansible Galaxy from the commands line.

1. ansible-galaxy search: will search for roles.the argument is provided with the

ansible-galaxy search command, it will search in the role description. --author, --platform,

--galaxy tags

```
  ansible-galaxy search '          install mariadb'

ansible-galaxy search '            install mariadb' --platforms el
```

2. ansible-galaxy info: which provides information about roles. ansible-galaxy info f500.mariadb55

3. ansible-galaxy install: downloads a role and installs it on the control nodes in the direct link ~/.ansible/roles

```
  ansible-galaxy install geerlingguy.nginx
```

4. ansible-galaxy list: shows installed ansible-galaxy roles, but you should notice that it's only looking in its own default directories.

```
ansible-galaxy list
```

5. ansible-galaxy remove: to remove roles

```
ansible-galaxy remove f500.mariadb55
```

6. ansible-galaxy init: to create directory structure that helps you to start developing a new role.ansible-galaxy init directly interacts with the Ansible Galaxy website API. If you don't want to do that, you can use --offline to work completely offline.While using ansible-galaxy init, you can specify a username and role name as arguments.

```
ansible-galaxy init --offline mekawy.myrole
```

Notice :: It is better to go with creating a roles directory called top level in the home directory of the Ansible user, and creating symbolic links in the product directories that need access to the specific roles.

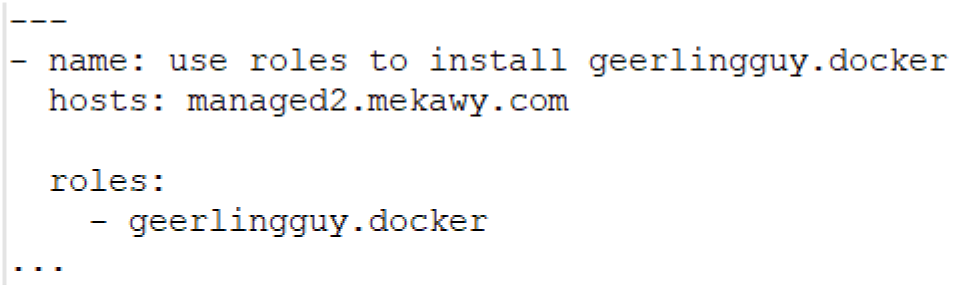
#ansible-galaxy Lab

Requirements:

Use Ansible Galaxy to find a role that installs Docker on CentOS, and use this role to perform the installation.

Solutions: https://galaxy.ansible.com/geerlingguy/docker

```
[ansible@controller ~]$ ansible-galaxy install geerlingguy.docker

- downloading role 'docker', owned by geerlingguy

- downloading role from
https://github.com/geerlingguy/ansible-role-docker/archive/3.1.2.tar.gz

- extracting geerlingguy.docker to /home/ansible/.ansible/roles/geerlingguy.docker

- geerlingguy.docker (3.1.2) was installed successfully

[ansible@controller ~]$ ansible-galaxylist

# /home/ansible/.ansible/roles

- geerlingguy.nginx, 3.0.0

- geerlingguy.docker,    3.1.2    #

/usr/share/ansible/roles          #

/etc/ansible/roles

[ansible@controller roles-lab]$ vim docker-geerlingguy.yml
```



```
---
- name: use roles to install geerlingguy.docker
  hosts: managed2.mekawy.com

  roles:
    - geerlingguy.docker
...
```

```
[ansible@controller roles-lab]$ ansible-playbook docker-geerlingguy.yml PLAY [use roles

to install geerlingguy.docker]
*****************************************************************************

**************************
```

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

[managed2.mekawy.com]

TASK [geerlingguy.docker : include_tasks]
************************************************************************

******************************

included: /home/ansible/.ansible/roles/geerlingguy.docker/tasks/setup-RedHat.yml for
managed2.mekawy.com

TASK [geerlingguy.docker : Ensure old versions of Docker are not installed.]
*******************************************************************  ok:

[managed2.mekawy.com]

TASK [geerlingguy.docker : Add Docker GPG key.]
************************************************************************
*********************** changed:

[managed2.mekawy.com]

TASK [geerlingguy.docker : Add Docker repository.]
************************************************************************

*********************

changed: [managed2.mekawy.com]

TASK [geerlingguy.docker : Configure Docker Nightly repo.]
************************************************************************

*************

changed: [managed2.mekawy.com]
```

```
TASK [geerlingguy.docker : Configure Docker Test repo.]
**************************************************************************
****************

changed: [managed2.mekawy.com]

TASK [geerlingguy.docker : Ensure container-selinux is installed.]
**************************************************************************
*****

skipping: [managed2.mekawy.com]

TASK [geerlingguy.docker : Ensure containerd.io is installed.]
**************************************************************************
*********

skipping: [managed2.mekawy.com]

TASK [geerlingguy.docker : include_tasks]
**************************************************************************
*****************************

skipping: [managed2.mekawy.com]

TASK [geerlingguy.docker : Install Docker.]
**************************************************************************
*************************** changed:

[managed2.mekawy.com]

TASK [geerlingguy.docker : Ensure Docker is started and enabled at boot.]
************************************************************************ changed:

[managed2.mekawy.com]

RUNNING HANDLER [geerlingguy.docker : restart docker]
**************************************************************************
******************
```

```
changed: [managed2.mekawy.com]

TASK [geerlingguy.docker : include_tasks]
****************************************************************************

******************************

included: /home/ansible/.ansible/roles/geerlingguy.docker/tasks/docker-compose.yml for
managed2.mekawy.com

TASK [geerlingguy.docker : Check current docker-compose version.]
****************************************************************************

******

ok: [managed2.mekawy.com]

TASK [geerlingguy.docker : Delete existing docker-compose version if it's different.]
************************************************************** skipping:

[managed2.mekawy.com]

TASK [geerlingguy.docker : Install Docker Compose (if configured).]
****************************************************************************

****

changed: [managed2.mekawy.com]

TASK [geerlingguy.docker : include_tasks]
****************************************************************************
******************************

skipping: [managed2.mekawy.com]

PLAY RECAP
****************************************************************************

***************************************************************

managed2.mekawy.com          : ok=13  changed=8       unreachable=0      failed=0     skipped=5
rescued=0      ignored=0
```

```
#Verification

[root@managed2 ~]# docker pull nginx Using
default tag: latest
latest: Pulling from library/nginx 69692152171a: Pull
complete 49f7d34d62c1: Pull complete
5f97dc5d71ab: Pull complete cfcd0711b93a: Pull
complete be6172d7651b: Pull complete
de9813870342: Pull complete
Digest: sha256:df13abe416e37eb3db4722840dd479b00ba193ac6606e7902331dcea50f4f1f2
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest

[root@managed2 ~]# docker ps -a

CONTAINER ID   IMAGE         COMMAND   CREATED   STATUS         PORTS      NAMES

[root@managed2 ~]# docker run -d -it nginx
eae84ce61df9f25559bad3cab9c013426248da19fcc97eed403eb2f22d27e802 [root@managed2
~]# docker ps -a
CONTAINER ID   IMAGE         COMMAND                 CREATED         STATUS         PORTS
NAMES
```

```
eae84ce61df9   nginx          "/docker-entrypoint.…" 17 seconds ago Up 16 seconds 80/tcp
relaxed_lamarr

[root@managed2 ~]# docker stop eae84ce61df9 eae84ce61df9

[root@managed2 ~]# docker ps -a

CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS                    PORTS
NAMES

eae84ce61df9   nginx          "/docker-entrypoint.…"  About a minute ago  Exited (0) 7 seconds ago
               relaxed_lamarr
```

3- Ansible Vault

Ansible Vault is a solution. You can use it to encrypt and decrypt data files that are used by Ansible. by default, passwords, API keys, and other sensitive data are not stored in a secure way in Ansible.

Use the ansible-vault utility. This utility creates, edits, encrypts, decrypts, and views file.

To do the encryption, Vault is using an external Python library. You should realize though that it has not officially been audited by an external party.

Ansible Vault is not the only solution that you can use to encrypt sensitive data. There are other solutions as well in external products like HashiCorp Vault or Microsoft Azure Key Vault or AWS Key Management Service.

1. ansible-vault create <filename> : command to create an encrypted file.

2. ansible-vault create -- vault-password-file=vault-pw secret.yml : to use the password. So, that would create a file with the name secret.yml, and instead of prompting for a password, it would use the password that is invault-pw

A vault password file can be used to avoid entering the password on the console. So, a password file is just a text file that contains the password. Obviously the password file contains sensitive data, so you need to make sure that the password file is in a protected area and has restrictive permissions.

The password file itself is a text file that contains the password. So make sure you store it in a secure location.

3. ansible-vault edit secret.yml : to edit an existing encrypted file

4. ansible-vault view secret.yml : if you want to see the content of an encrypted file

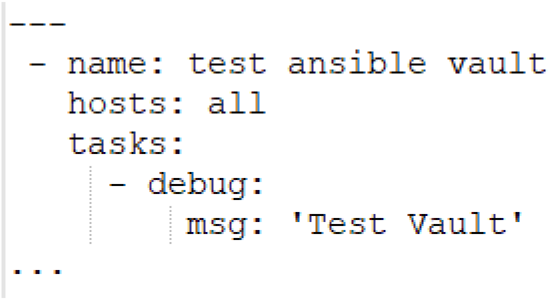
5. ansible-vault rekey secret.key : to sen a new password on an encrypted file.It is allowed to use multiple file names in one command using ansible-vault rekey.

6. ansible-vault --new-vault-password-file=new-file secret.yml when using the password file.

7. ansible-vault encrypt filename.yml : to encrypt an existing file, to save the encrypted file to a new name, you can use the option--output=newfile

8. ansibe-vault decrypt filename.yml : to decrypt an existing file #example

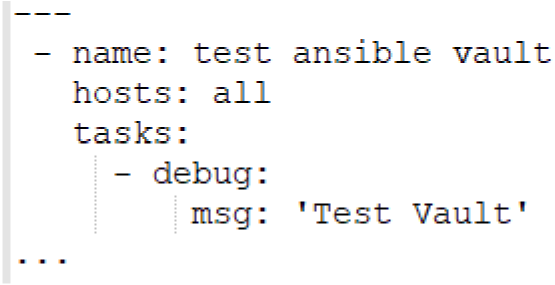
```
[ansible@controller ~]$ mkdir valut
[ansible@controller ~]$ cd valut/
[ansible@controller valut]$ ansible-vault create secret.yml New
Vault password:ansible
Confirm New Vault password:ansible
```



```
---
 - name: test ansible vault
   hosts: all
   tasks:
     - debug:
         msg: 'Test Vault'
...
```

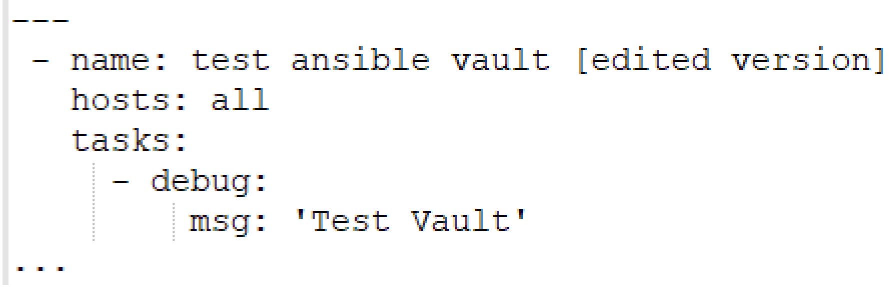
```
[ansible@controller valut]$ cat secret.yml

$ANSIBLE_VAULT;1.1;AES256

65323731616561626635366531626233393131623730616432636566313433363834373531323963

65386430393431376264313230313134366534303663383 70a363062303338353737363763663936

63663563303630303264333531383834363336373338323164366330333333343464316163393563

6234333534356563310a6363363363646533666436353139393231646630383963316238396 26232

63663839393565616366353535386235323336616465313063343034303265313630393165383730

31306230366536623864643538326432373633356131316337643838373739643063363032343434

30666631623561346538613763373930353662313461643330663065336232373137643466326233

63343533353931626335303838376362646532386134386638353361316330326465373933323037

3966323531613766366561343361323437626638346664623564373466393134393
```

```
[ansible@controller valut]$ ansible-vault view secret.yml Vault
password:ansible

[ansible@controller valut]$ ls -l secret.yml
```



```
---
 - name: test ansible vault
   hosts: all
   tasks:
     - debug:
         msg: 'Test Vault'
...
```

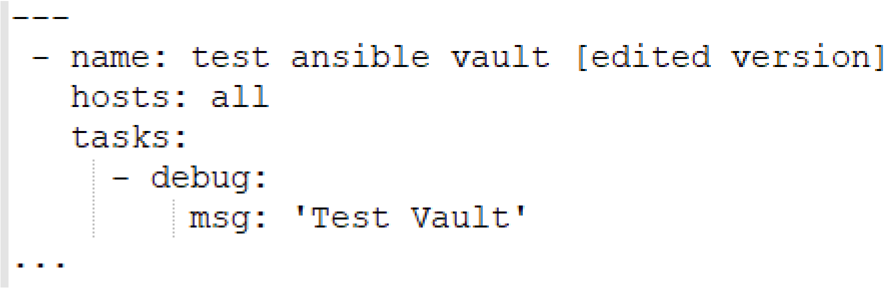
```
-rw-------. 1 ansible ansible 743 May 16 02:02secret.yml

[ansible@controller valut]$ ansible-vault edit secret.yml Vault
password:

[ansible@controller valut]$ ls -l secret.yml
```



```
---
 - name: test ansible vault [edited version]
   hosts: all
   tasks:
     - debug:
         msg: 'Test Vault'
...
```

```
-rw-------. 1 ansible ansible 808 May 16 02:05secret.yml


[ansible@controller valut]$ ansible-vault rekey secret.yml Vault
password:ansible
New Vault password:ansible2 Confirm New
Vault password:ansible2
```

```
Rekey successful

[ansible@controller valut]$ ansible-vault decrypt secret.yml --output=nomoresecret.yml Vault

password:ansible2

Decryption successful [ansible@controller

valut]$ ls nomoresecret.yml secret.yml

[ansible@controller valut]$ cat nomoresecret.yml
```



```
---
 - name: test ansible vault [edited version]
   hosts: all
   tasks:
     - debug:
         msg: 'Test Vault'
...
```

```
[ansible@controller valut]$ cat secret.yml

$ANSIBLE_VAULT;1.1;AES256

62366261663233326565653331366138393832663065346363303663646439383534616235
376337

3163333364663966346335356561363931306263666131620a623736376531356366643961
373964

39333735383461373339656433316538326435343037626530336165626633363139343139
626164

3935386665386564630a623338363531313962383862346638366537343737386536653735
643135

30653961333034363939643337313532636636393661343539373236396532353535393166
633463
```

```
37326132323662313633643961366235666263663031373962633462336538363236303066
613864

64626665623966626135626262613764633330313436333463366431353466616538373966
623831

38333331343266626530333132646230316132623536663761616631303033386634633261
646239

32373065303762343234333161613364666237346165613533663030373530373834383561
643964

643432393336363935383666346530393236643837653631353
```

3.1 Execute playbooks using Ansible Vault

1. Vault can be used to encrypt sensitive data in variables as well as in playbooks. If you want to use this in Ansible playbooks environments then you should make sure that sensitive data is stored in encrypted files and include the encrypted files in the unencrypted playbook to make it secure

2. For inventory variables, I would advise you to create a host_vars or group_vars directory. In that directory, create a subdirectory with the name of the host group or host, and in there, if you wanna use inventory variables, store an unencrypted vars file along with an encrypted vars file.

3. For playbook variables, you can put them in an encrypted separate file that is included using vars_file.

#ansible vault lab

Requirements:

1. use ansible-vault to create a playbook that creates new users and define the users to be created together with their passwords in a variable file that is to be included.

2. In the task that uses the user module you can use the following to make sure that an SHA512 hashed password is written based on the input in the variable file.

password: "{{ item.pw | password_hash('sha512') }}"

3. create a password file so that you don't have to enter a password when prompted for it. Solution:

[ansible@controller ~]$ mkdir vault-lab [ansible@controller ~]$

cd vault-lab/ [ansible@controller vault-lab]$ cp ~/inventory .

[ansible@controller vault-lab]$ cp ~/ansible.cfg .

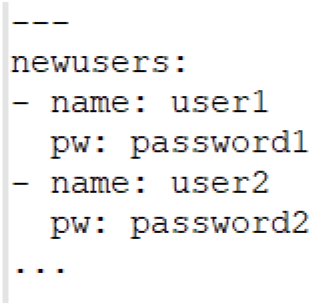
[ansible@controller vault-lab]$ mkdir vars [ansible@controller

vault-lab]$ cd vars/

[ansible@controller vars]$ ansible-vault create secret.yml New Vault

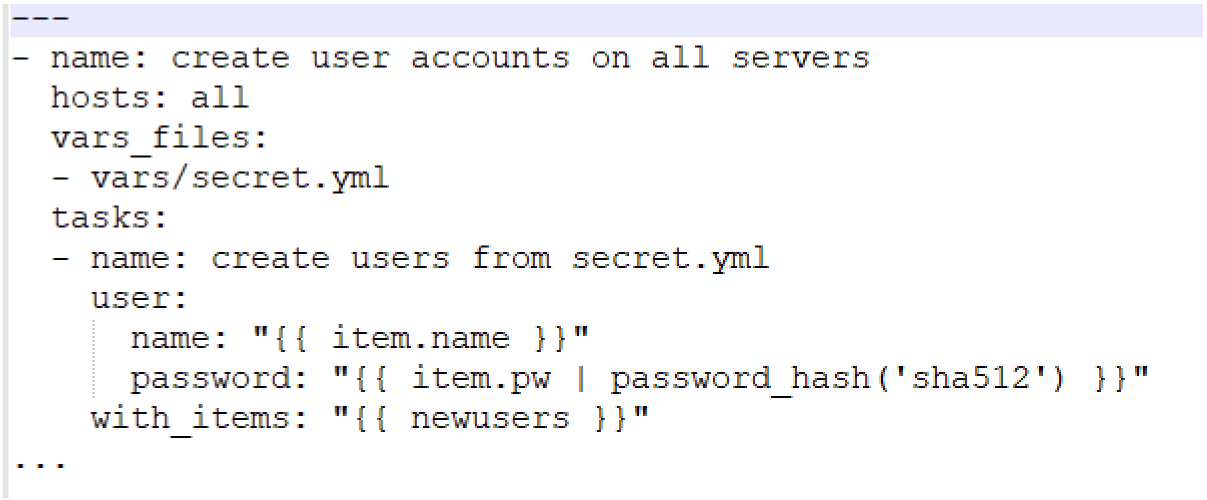
password:ansible

Confirm New Vault password:



```
---
newusers:
- name: user1
  pw: password1
- name: user2
  pw: password2
...
```

[ansible@controller vault-lab]$ vim createusers.yml



```
---
- name: create user accounts on all servers
  hosts: all
  vars_files:
  - vars/secret.yml
  tasks:
  - name: create users from secret.yml
    user:
      name: "{{ item.name }}"
      password: "{{ item.pw | password_hash('sha512') }}"
    with_items: "{{ newusers }}"
...
```

#create and manage permissions on vault password file

```
[ansible@controller vault-lab]$ echo ansible > vaultpw

[ansible@controller vault-lab]$ ls

ansible.cfg createusers.yml inventory vars vaultpw [ansible@controller

vault-lab]$ ls -l

total 16

-rw-rw-r--. 1 ansible ansible 169 May 16 02:28 ansible.cfg

-rw-rw-r--. 1 ansible ansible 278 May 16 02:37 createusers.yml

-rw-rw-r--. 1 ansible ansible 125 May 16 02:28 inventory

drwxrwxr-x. 2 ansible ansible 24 May 16 02:32 vars

-rw-rw-r--.1ansibleansible 9May1602:39vaultpw

[ansible@controllervault-lab]$chmod600vaultpw

[ansible@controller vault-lab]$ ls-l

total 16
```

```
-rw-rw-r--. 1 ansible ansible 169 May 16 02:28 ansible.cfg

-rw-rw-r--. 1 ansible ansible 278 May 16 02:37 createusers.yml

-rw-rw-r--. 1 ansible ansible 125 May 16 02:28 inventory

drwxrwxr-x. 2 ansible ansible 24 May 16 02:32 vars

-rw-------. 1 ansible ansible  9 May 16 02:39 vaultpw

[ansible@controller vault-lab]$ ansible-playbook --vault-password-file=vaultpw
createusers.yml

PLAY [create user accounts on all servers]
*****************************************************************************

******************************

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

[managed1.mekawy.com]

ok: [managed2.mekawy.com] ok:

[controller.mekawy.com]

TASK [create users from secret.yml]
*****************************************************************************

**************************************

changed: [managed1.mekawy.com] => (item={u'name': u'user1', u'pw': u'password1'}) changed:

[managed2.mekawy.com] => (item={u'name': u'user1', u'pw': u'password1'}) changed:

[controller.mekawy.com] => (item={u'name': u'user1', u'pw': u'password1'}) changed:

[managed1.mekawy.com] => (item={u'name': u'user2', u'pw': u'password2'}) changed:

[managed2.mekawy.com]=>(item={u'name':u'user2',u'pw':u'password2'})
```

```
changed: [controller.mekawy.com] => (item={u'name': u'user2', u'pw': u'password2'})

PLAY RECAP
****************************************************************************
***************************************************************

controller.mekawy.com          : ok=2    changed=1     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0

managed1.mekawy.com            : ok=2    changed=1     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0

managed2.mekawy.com            : ok=2    changed=1     unreachable=0     failed=0     skipped=0
rescued=0     ignored=0

#verification

[ansible@controller vault-lab]$ ssh -o PreferredAuthentications=password
user1@managed1.mekawy.com #set preferred authentications to temporarily override the authentication
method to use a password

user1@managed1.mekawy.com's password:

Welcome to managed1.

Today is 2021-05-16.

Access to this system is for authorized users only Contact

mohamed@mekawy.com for more information.

[user1@managed1 ~]$ exit

logout

Connection to managed1.mekawy.com closed.
```

4- Ansible Optimization and Troubleshooting

4.1 Ansible Optimization

4.1.1 Selecting host patterns

Host patterns are all about what you're addressing in a playbook or from the command line while running a playbook.

The simplest host pattern is just the name of the host.

You can also use IP addresses, but the only requirement is that the IP address must be in the inventory.

You can use groups as well, groups as specified in the inventory.

Notice :: That when using groups, the group "all" is implicit. So you don't have to define a group "all" in the inventory, it's automatically there and it refers to all hosts.

There's also the group "ungrouped" which is for hosts that are not a member of any group, which may occur in the inventory as well.

You can use * as a wildcard, but if you use * as a wildcard don't forget to quote ' '.Quoting is recommended in all cases to ensure that the shell that is interpreting the Ansible command doesn't interpret special characters. That's all about command line parsing, you know? Star in a bash shell is a function of all files in the current directory and when the bash shell analyzes your Ansible command, it finds a star it will substitute with all files in the current directory, so that's going wrong. So put it between single quotes to make sure that it's the Ansible command that is interpreting the star and not the bash shell. i.e. ansible '*.mekawy.com' -i inventory --list-hosts

You can use comma-separated lists of hosts to specify a logical list.can use different items like host names or groups or an IP address in a logical list where the items are separated by commas. i.e ansible managed1.mekawy.com,192.168.1.9 -i inventory --list-hosts

& can be used as a logical and, which means that hosts must match that item also. i.e. ansible 'web,&prod' -inventory --lists-hosts will match machines in the group web only if they are also a member of the group "prod".

! can be used as a logical not, i.e. ansible 'web,!web1.example.com' - inventory --list-hosts will match machines in the group web except for web1.example.com whis is a member of web group as well.

4.1.2 Configuring delegation

First we need to try to understand why you need delegation? So the reason is that sometimes Ansible needs to configure more than just the managed hosts. That may be the case in a monitoring environment, for example, where remote hosts needs to be added to the environment or a DNS server that needs to be modified after adding a server to the configuration. So the idea is that you would manage the monitoring node or the DNS server, but not all the nodes that need to communicate yet.

You can always run tasks directly on all the hosts that are involved, but that is slow. Also because fact gathering needs to happen for all the hosts that are involved even if you can disable fact gathering.

Delegation can happen to different host types.

You can do it on the local machine, where the local machine is the Ansible control node. And you should realize that the Ansible control node, is not necessarily a part of the managed environment. But sometimes, you might want to include it anyway.

You can delegate to hosts outside of the play Host

maybe within or without the inventory.

delegate_to module

If you use delegate_to to delegate module, it will run on the host specified by delegate_to.

Facts available will be the ones of the original host and not the delegated host. That's because of fact gathering that doesn't happen on delegated host.

The task has the context of the original target host, but gets executed on the host the task is delegated to. And this can be useful to tell the another host to do something with the managed host. Like, think of removing or adding a host to or from a load balancer or DNS.

delegate_to: hostname to use delegation.

local_action as a shortcut to delegate_to: localhost if you want to learn something on local host (ansible master conroller node). This action will result in an action that's executed on the Ansible mater node.

Notice:: that the localhost entry is implicit and you don't have to define it in the inventory.

```
[ansible@controller delegation]$ ansible localhost -m command -a hostname localhost |

CHANGED | rc=0 >>

controller.mekawy.com
```

Addressing hosts that are in the inventory is straightforward. You just have to address the hostname. And to address hosts outside of the inventory, use the add_hosts module.

Don't forget to configure the delegate_to hosts with user account, sudo credentials, and SSH keys because if Ansible needs to do something on a delegated host, Ansible must be capable of accessing the delegated host. So you need your Ansible user including SSH keys for the connectivity and sudo credentials so that credentials can be escalated.

#example

[ansible@controller ~]$ mkdir delegation/

[ansible@controller ~]$ cd delegation/

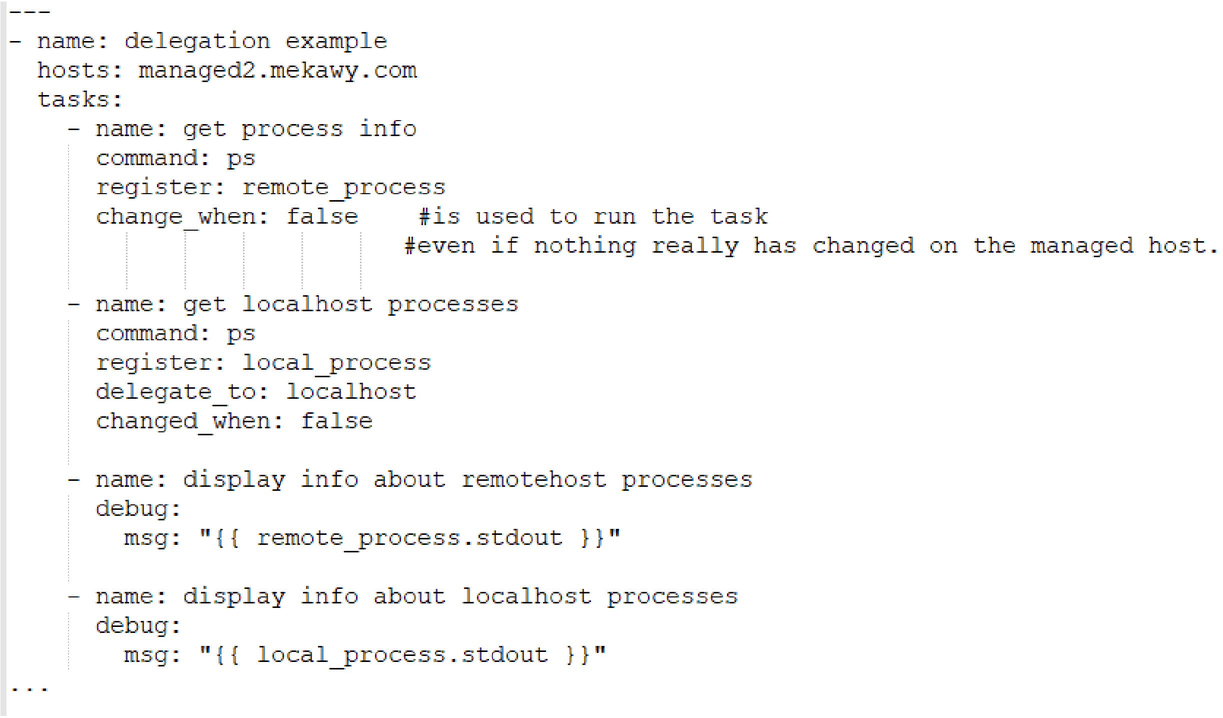
[ansible@controller ~]$ cp ~/inventory .

[ansible@controller ~]$ cp ~/ansuble.cfg .

[ansible@controller ~]$ vim delegate.yml

4.1.3 Delegation outside the Inventory

Why would you wanna do that? Well, in the inventory, every host that is in the inventory can be addressed by using all. And if you do delegation, you might want to have hosts that are not in the inventory. The advantage of not including hosts in the inventory is that when you use all to run tasks on managed host, the hosts outside of the inventory are not included.



```
---
- name: delegation example
  hosts: managed2.mekawy.com
  tasks:
    - name: get process info
      command: ps
      register: remote_process
      change_when: false    #is used to run the task
                           #even if nothing really has changed on the managed host.

    - name: get localhost processes
      command: ps
      register: local_process
      delegate_to: localhost
      changed_when: false

    - name: display info about remotehost processes
      debug:
        msg: "{{ remote_process.stdout }}"

    - name: display info about localhost processes
      debug:
        msg: "{{ local_process.stdout }}"
...
```

So, you can clearly distinguish between the delegated host and everything else.

When you are addressing a host that is not in the inventory, credentials are needed anyway. These credentials normally are related to the fact that hosts are in the inventory. But the delegated host, as we have discussed before, also needs to be configured to be connected by Ansible, and that is sudo and ssh.

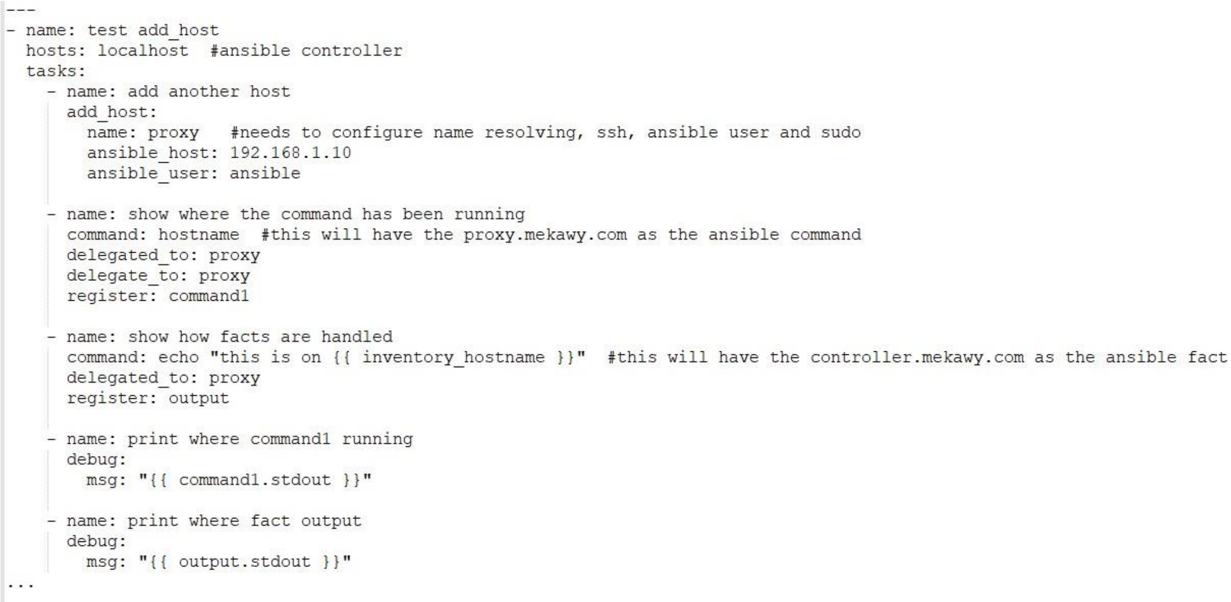
add_hosts module

When accessing a host outside of the inventory, a temporary entry in the inventory must be created by using add_hosts.Add_host is happening from within the playbook, not in the inventory.

Ansible then will use the same connection type and details used for the manged host to connect to the delegating host.

#example

```
[ansible@controller delegation]$ vim delegate_addhost.yml
```



```
---
- name: test add_host
  hosts: localhost  #ansible controller
  tasks:
    - name: add another host
      add_host:
        name: proxy   #needs to configure name resolving, ssh, ansible user and sudo
        ansible_host: 192.168.1.10
        ansible_user: ansible

    - name: show where the command has been running
      command: hostname  #this will have the proxy.mekawy.com as the ansible command
      delegated_to: proxy
      delegate_to: proxy
      register: command1

    - name: show how facts are handled
      command: echo "this is on {{ inventory_hostname }}"  #this will have the controller.mekawy.com as the ansible fact
      delegated_to: proxy
      register: output

    - name: print where command1 running
      debug:
        msg: "{{ command1.stdout }}"

    - name: print where fact output
      debug:
        msg: "{{ output.stdout }}"
...
```

[ansible@controller delegation]$ ansible-playbook -vvv delegate_addhost.yml

delegate_facts: True

the default behavior regarding facts is that facts are gathered on the host where the playbook is running and not on the delegated host

if you do want to run the facts gathering on the delegated host, then you use delegate_facts:True.That will make sure that you will gather facts from the delegate_to host as well, and that the facts that you are using as variables in your playbook apply to the delegated host and not to the original host.

4.1.4 Configure parallelism

Running tasks in parallel will make Ansible faster. Especially if there is a large amount of notes that needs to be managed. As default behavior, Ansible can and will run tasks in parallel, on all hosts.

By default, tasks can run on five hosts at once. Andif that is not sufficient you can use forks=nn, in etc/ansible/ansible.cfg to increase this number. Alternatively, you could also use the --fork option, with the ansible-playbook or ansible command, to not make it a default, but run it for individual commands.

"serial" keyword that you could use in the playbook, to reduce the number of parallel tasks, to a value that is lower than what is specified with the forks option.So for example if you have 100 web service, and only ten should be updated at a specific time, you may set forks to 100, and serial to ten, within that playbook task to only update ten at a time.

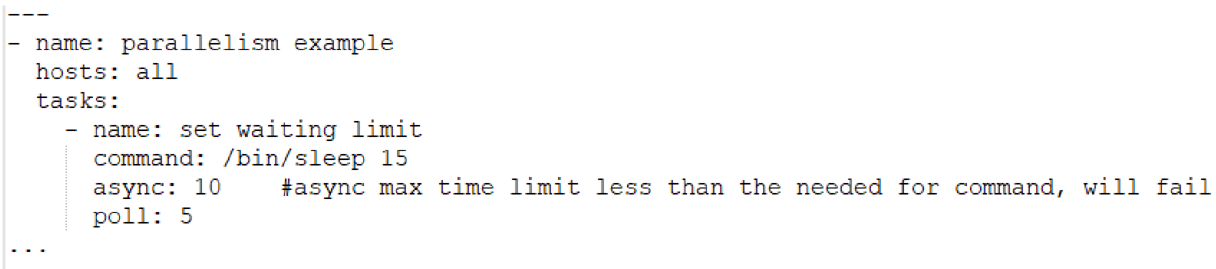
Normally, Ansible waits for the completion of tasks, before starting the next task. And that can take long time. The "async" keyword in a task can be used to run a task in the background.For example, async:3600 tells Ansible to give the task an hour to complete. And note that this will be the maximum amount of time permitted for the job to run. So if you go above that, the job will fill.

Using async allows the next task to be started, so it will make playbooks more efficient. And it's recommended for things like backup jobs, yum updates, large file downloads, and anything where you expect the play to take a longtime.

Poll:10 indicates that Ansible will poll every ten seconds to see if the command has completed.

#example

[ansible@controller delegation]$ vim waitforme.yml



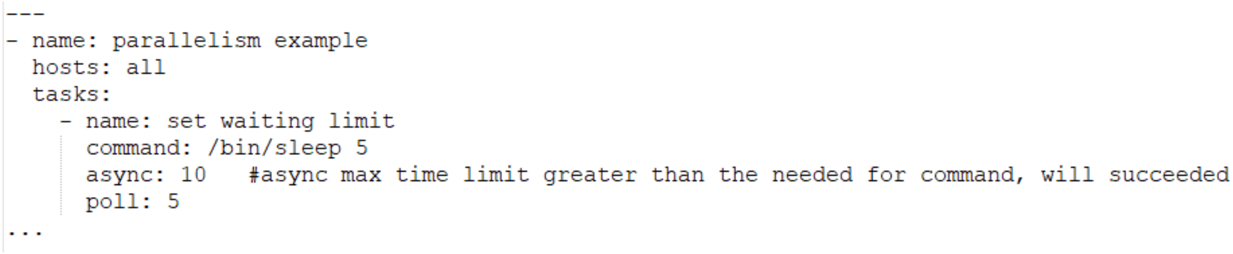
```
---
- name: parallelism example
  hosts: all
  tasks:
    - name: set waiting limit
      command: /bin/sleep 15
      async: 10    #async max time limit less than the needed for command, will fail
      poll: 5
...
```

[ansible@controller delegation]$ ansible-playbook waitforme.yml

```
PLAY [parallelism example]
******************************************************************************

************************************************

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

[managed2.mekawy.com]

ok: [managed1.mekawy.com] ok:

[controller.mekawy.com]

TASK [set waiting limit]
******************************************************************************

**************************************************

fatal:[managed1.mekawy.com]:FAILED!=>{"changed":false,"msg":"asynctaskdidnot complete within the
requested time - 10s"}

fatal:[managed2.mekawy.com]:FAILED!=>{"changed":false,"msg":"asynctaskdidnot complete within the
requested time - 10s"}

fatal: [controller.mekawy.com]: FAILED! => {"changed": false, "msg": "async task did not complete within the
requested time - 10s"}

PLAY RECAP
******************************************************************************

*****************************************************************

controller.mekawy.com        : ok=1    changed=0    unreachable=0    failed=1    skipped=0
rescued=0    ignored=0

managed1.mekawy.com          : ok=1    changed=0    unreachable=0    failed=1    skipped=0
rescued=0    ignored=0
```

```
managed2.mekawy.com           : ok=1     changed=0     unreachable=0     failed=1     skipped=0
rescued=0     ignored=0
```

```
[ansible@controller delegation]$ vim waitforme.yml
```



```
---
- name: parallelism example
  hosts: all
  tasks:
    - name: set waiting limit
      command: /bin/sleep 5
      async: 10   #async max time limit greater than the needed for command, will succeeded
      poll: 5
...
```

```
[ansible@controller delegation]$ ansible-playbook waitforme.yml PLAY

[parallelism example]
*********************************************************************************

**************************************************

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

[managed1.mekawy.com]

ok: [managed2.mekawy.com] ok:

[controller.mekawy.com]

TASK [set waiting limit]
*********************************************************************************
*************************************************

changed: [managed2.mekawy.com]

changed: [managed1.mekawy.com] changed:

[controller.mekawy.com]
```

```
PLAY RECAP
**************************************************************************************

*********************************************************************

controller.mekawy.com        : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed1.mekawy.com          : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com          : ok=2    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

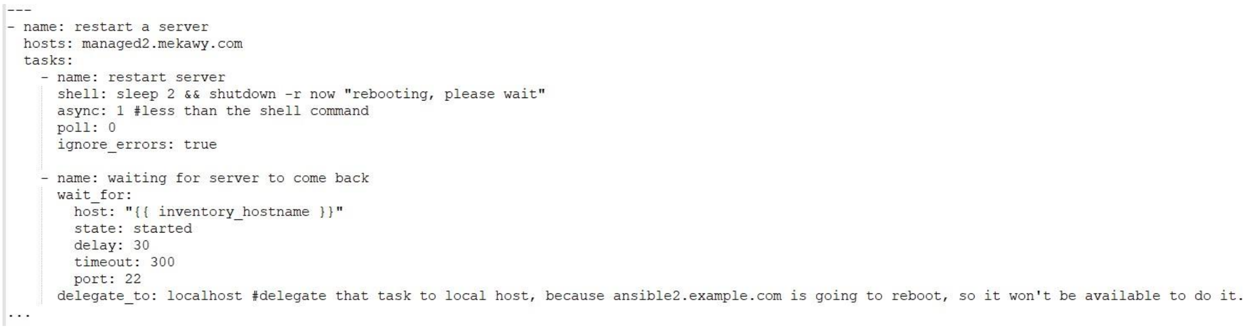
wait_for module

The wait_for module can be used in a task to check if a certain condition was met. So using this module may be useful to verify successful restart of servers, and stuff like that.

Use poll:0 in a task to tell Ansible not to wait for completion of this task, but to move on to the next task. And you can add ignore_errors as well, to prevent an error conditioning arising, and have this task fail.

#example

[ansible@controller delegation]$ vim reboot.yml



```
---
- name: restart a server
  hosts: managed2.mekawy.com
  tasks:
    - name: restart server
      shell: sleep 2 && shutdown -r now "rebooting, please wait"
      async: 1 #less than the shell command
      poll: 0
      ignore_errors: true

    - name: waiting for server to come back
      wait_for:
        host: "{{ inventory_hostname }}"
        state: started
        delay: 30
        timeout: 300
        port: 22
      delegate_to: localhost #delegate that task to local host, because ansible2.example.com is going to reboot, so it won't be available to do it.
...
```

[ansible@controller delegation]$ ansible-playbook reboot.yml

```
PLAY [restart a server]
***************************************************************************************

******************************************************

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

[managed2.mekawy.com]

TASK [restart server]
***************************************************************************************
****************************************************** changed:

[managed2.mekawy.com]

TASK [waiting for server to come back]
***************************************************************************************
************************************ ok:

[managed2.mekawy.com]

PLAY RECAP
***************************************************************************************

*********************************************************************

managed2.mekawy.com           : ok=3    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0


#verification

[ansible@controller delegation]$ ssh managed2

Lastlogin:TueMay1802:22:282021fromcontroller.mekawy.com Welcome

to managed2.
```

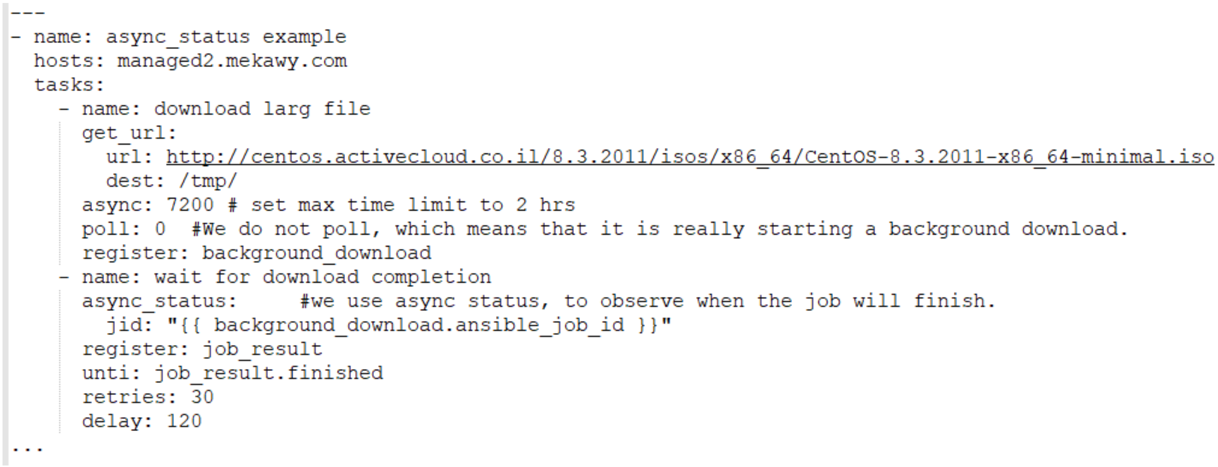
```
Today is 2021-05-16.

Access to this system is for authorized users only Contact

mohamed@mekawy.com for more information.

[ansible@managed2 ~]$ uptime

 02:23:24 up 0 min, 1 user, load average: 1.63, 0.41, 0.14

[ansible@managed2 ~]$ exit

logout

Connection to managed2 closed.
```

async_status module

For a start-and-forget type of task, you can use the async_status module to figure out where it currently is at. This allows you to finish everything in the playbook, and close the run when you get positive results back from async status.

#example

```
[ansible@controller delegation]$ vim download.yml
```



```
---
- name: async_status example
  hosts: managed2.mekawy.com
  tasks:
    - name: download larg file
      get_url:
        url: http://centos.activecloud.co.il/8.3.2011/isos/x86_64/CentOS-8.3.2011-x86_64-minimal.iso
        dest: /tmp/
      async: 7200 # set max time limit to 2 hrs
      poll: 0  #We do not poll, which means that it is really starting a background download.
      register: background_download
    - name: wait for download completion
      async_status:     #we use async status, to observe when the job will finish.
        jid: "{{ background_download.ansible_job_id }}"
      register: job_result
      unti: job_result.finished
      retries: 30
      delay: 120
...
```

```
[ansible@controller delegation]$ ansible-playbook download.yml
```

Big Lab

Requirements:

The server managed2.mekawy.com must be configured as a web server, and the server managed1.mekawy.com must be configured as a proxy server.

delegation must be used from the managed2 server to configure the proxy. Make sure that managed1.mekawy.com is ready to be managed by Ansible and included in the inventory.

se the example Jinja2 templates to create the required httpd content and configuration on both servers.

use delegate_to in combination with with_items to perform the delegation to the managed1 server.

The main playbook, web.yml, should be used for the following tasks. installs

httpd on all hosts involved.

configures firewalld to accept incoming http traffic.

uses the provided templates to copy the httpd configuration to the target hosts.The configuration templates must be copied to the file /etc/httpd/conf.d/ansible.conf.

to verify the procedure, use a browser to connect to http://managed1.mekawy.com/external. You should see the web page stating, "you are connected to managed2.mekawy.com", which is proof that the proxy is working.

template for managed2.mekawy.com

....

```
#{{ ansible_managed }}
NameVirtualHost *:80

<VirtualHost *:80>
  serverAdmin root@"{{ ansible_fqdn }}"
  DocumentRoot /var/www/html/ ServerName "{{
  ansible_fqdn }}" Errorlogs/"{{ ansible_fqdn }}"-
  error.log
  CustomLog logs/"{{ ansible_fqdn }}"-access.log common
</VirtualHost>
```

template for managed1.mekawy.com [proxy]

....

```
#{{ ansible_managed }}

ProxyPass "/external" "http://{{ ansible_fgdn }}" ProxyPassReverse
"/external" "http://{{ansible_fqdn }}"
```

template for index.html

....

```
Your are connected to {{ansible_fqdn}}
```

Solution:

```
[ansible@controller delegation]$ vim inventory [all]
controller.mekawy.com
managed1.mekawy.com
managed2.mekawy.com

[webservers]
managed2.mekawy.com

[proxyservers]
managed1.mekawy.com

[ansible@controller delegation]$ mkdir templates [ansible@controller
delegation]$ cd templates/
[ansible@controller templates]$ vim managed2.mekawy.com-httpd.conf.j2 #{{
ansible_managed }}
NameVirtualHost *:80
```

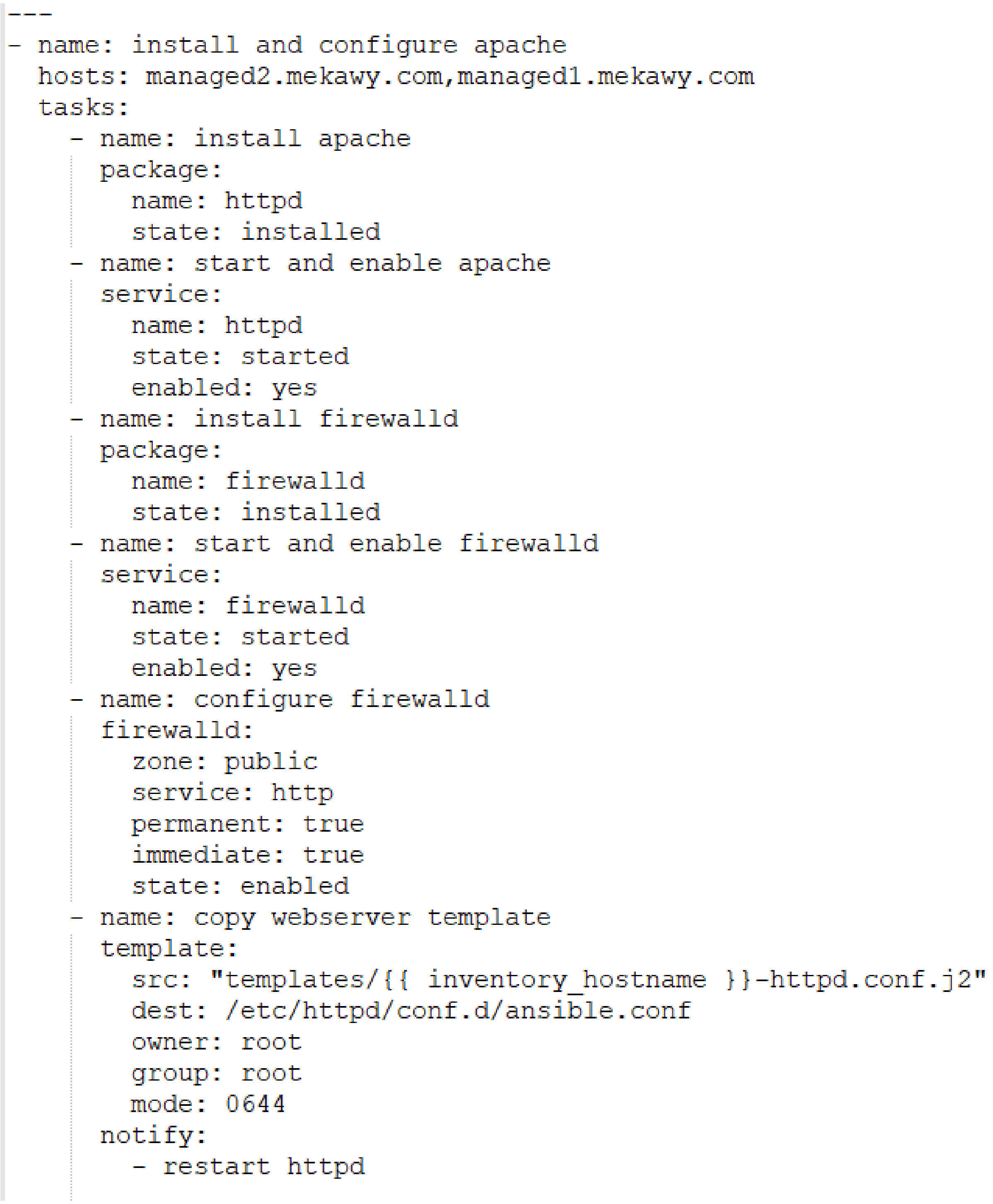
```
<VirtualHost *:80>
  serverAdmin root@"{{ ansible_fqdn }}"
  DocumentRoot /var/www/html/ ServerName "{{
  ansible_fqdn }}" Errorlogs/"{{ ansible_fqdn }}"-
  error.log
  CustomLog logs/"{{ ansible_fqdn }}"-access.log common
</VirtualHost>
```

```
[ansible@controller templates]$ vim managed1.mekawy.com-httpd.conf.j2 #{{
ansible_managed }}
```

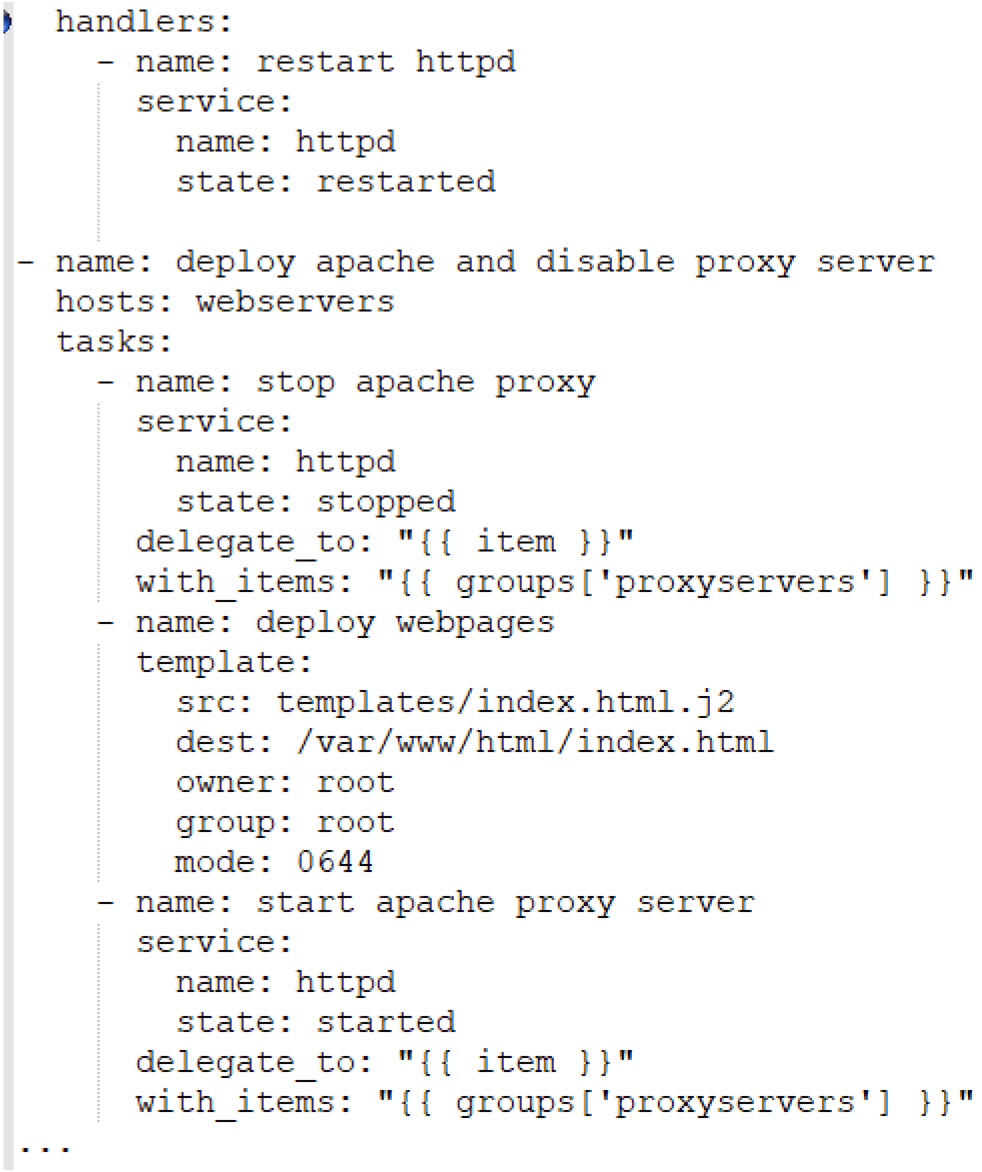
```
ProxyPass "/external" "http://{{ ansible_fqdn }}" ProxyPassReverse
"/external" "http://{{ansible_fqdn }}"
```

```
[ansible@controller templates]$ vim index.html.j2 Your are
connected to {{ ansible_fqdn }}
```

```
[ansible@controller templates]$ cd ..
[ansible@controllerdelegation]$vimweb.yml
```



```yaml
---
- name: install and configure apache
  hosts: managed2.mekawy.com,managed1.mekawy.com
  tasks:
    - name: install apache
      package:
        name: httpd
        state: installed
    - name: start and enable apache
      service:
        name: httpd
        state: started
        enabled: yes
    - name: install firewalld
      package:
        name: firewalld
        state: installed
    - name: start and enable firewalld
      service:
        name: firewalld
        state: started
        enabled: yes
    - name: configure firewalld
      firewalld:
        zone: public
        service: http
        permanent: true
        immediate: true
        state: enabled
    - name: copy webserver template
      template:
        src: "templates/{{ inventory_hostname }}-httpd.conf.j2"
        dest: /etc/httpd/conf.d/ansible.conf
        owner: root
        group: root
        mode: 0644
      notify:
        - restart httpd
```



```
  handlers:
    - name: restart httpd
      service:
        name: httpd
        state: restarted

- name: deploy apache and disable proxy server
  hosts: webservers
  tasks:
    - name: stop apache proxy
      service:
        name: httpd
        state: stopped
      delegate_to: "{{ item }}"
      with_items: "{{ groups['proxyservers'] }}"
    - name: deploy webpages
      template:
        src: templates/index.html.j2
        dest: /var/www/html/index.html
        owner: root
        group: root
        mode: 0644
    - name: start apache proxy server
      service:
        name: httpd
        state: started
      delegate_to: "{{ item }}"
      with_items: "{{ groups['proxyservers'] }}"
...
```

```
[ansible@controller delegation]$ ansible-playbook web.yml --syntax-check playbook: web.yml
```

```
[ansible@controller templates]$ ansible-playbook web.yml
```

```
[ansible@controller delegation]$ !ans
```

```
ansible-playbook web.yml

PLAY [install and configure apache]
*************************************************************************************

**************************************

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

[managed1.mekawy.com]

ok: [managed2.mekawy.com]

TASK [install apache]
*************************************************************************************
******************************************************* ok:

[managed1.mekawy.com]

ok: [managed2.mekawy.com]

TASK [start and enable apache]
*************************************************************************************
********************************************* ok:

[managed1.mekawy.com]

fatal: [managed2.mekawy.com]: FAILED! => {"changed": false, "msg": "Unable to start service httpd: Job for
httpd.service failed because the control process exited with error code. See \"systemctl status
httpd.service\" and \"journalctl -xe\" for details.\n"}

TASK [install firewalld]
*************************************************************************************
**************************************************** ok:

[managed1.mekawy.com]
```

```
TASK [start and enable firewalld]
*******************************************************************************
*************************************** ok:
[managed1.mekawy.com]
TASK [configure firewalld]
*******************************************************************************
********************************************** ok:
[managed1.mekawy.com]
TASK [copy webserver template]
*******************************************************************************
******************************************
changed: [managed1.mekawy.com]
RUNNING HANDLER [restart httpd]
*******************************************************************************
*****************************************
changed: [managed1.mekawy.com]
PLAY [deploy apache and disable proxy server]
*******************************************************************************
****************************

PLAY RECAP
*******************************************************************************
******************************************************************

managed1.mekawy.com          : ok=8    changed=2    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

managed2.mekawy.com          : ok=2    changed=0    unreachable=0    failed=1    skipped=0
rescued=0    ignored=0
```

4.2 Ansible Troubleshooting

4.2.1 Ansible logging

By default Ansible doesn't write any log file anywhere and that is because Ansible is quite verbose in writing information to the STDOUT.

If you want Ansible to write somewhere then you need to specify "log_path" in the default section of ansible.cfg which will force writing log files to a specific location. So this log_path is followed by the name of the log file that you want to create.

Alternatively instead of writing it to the ansible.cfg you can set the $ANSIBLE_LOG_PATH variable. While doing so there is one problem though, and the problem is that only the root user can log to /var/log. So you should consider creating log files in the local playbook directory, or provide write permissions on the var/log directory for your Ansible user.

Error messages that are written to the STDOUT will be written to the log file. Also, if you are going to write Ansible log files and you expect them to be very verbose then you should consider to configure Linux "logrotate" on the Ansible log files so that they are rotated after a specific time or when they get too big.

```
#example [ansible@controller

~]$ ls

ansible.cfg  biglab2  blocks          delegation  errors      inclusions  jinja2  loops           playbook1
playbook3      roles-lab uninstall  vault-lab

biglab       biglab3  conditionals  ec2           handlers  inventory   lab1         nestedloops
playbook2  registervar  tags              valut

[ansible@controller ~]$ cd errors/

[ansible@controller errors]$ ls

ansible.cfg cleanweb.yml force_handlers.yml index.html inventory
```

```
[ansible@controller errors]$ vim ansible.cfg [defaults]
remote_user=ansible
host_key_checking=false
inventory=inventory log_path=
errorlog

[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False
```

```
[ansible@controller errors]$ ansible-playbook force_handlers.yml PLAY

[force handlers example]
*********************************************************************************************
***************************************************

TASK [Gathering Facts]
*********************************************************************************************
*********************************************************** ok:
[managed1.mekawy.com]
ok: [managed2.mekawy.com] ok:
[controller.mekawy.com]
```

```
TASK [install httpd]

**************************************************************************************
********************************************************* ok:

[controller.mekawy.com]

ok: [managed2.mekawy.com] ok:

[managed1.mekawy.com]

TASK [copy index.html]

**************************************************************************************
******************************************************* ok:

[managed1.mekawy.com]

ok: [managed2.mekawy.com] ok:

[controller.mekawy.com]

TASK [copy nothing intented to fail]

**************************************************************************************

*****************************************

An exception occurred during task execution. To see the full traceback, use -vvv. The error was: If you are
using a module and expect the file to exist on the remote, see the remote_src option

fatal: [managed1.mekawy.com]: FAILED! => {"changed": false, "msg": "Could not find or access
'/tmp/nothing' on the Ansible Controller.\nIf you are using a module and expect the file to exist on the
remote, see the remote_src option"}

An exception occurred during task execution. To see the full traceback, use -vvv. The error was: If you are
using a module and expect the file to exist on the remote, see the remote_src option
```

```
fatal: [managed2.mekawy.com]: FAILED! => {"changed": false, "msg": "Could not find or access
'/tmp/nothing' on the Ansible Controller.\nIf you are using a module and expect the file to exist on the
remote, see the remote_src option"}

An exception occurred during task execution. To see the full traceback, use -vvv. The error was: If you are
using a module and expect the file to exist on the remote, see the remote_src option

fatal: [controller.mekawy.com]: FAILED! => {"changed": false, "msg": "Could not find or access
'/tmp/nothing' on the Ansible Controller.\nIf you are using a module and expect the file to exist on the
remote, see the remote_src option"}

PLAY RECAP
******************************************************************************

*****************************************************************

controller.mekawy.com         : ok=3    changed=0    unreachable=0    failed=1    skipped=0
rescued=0    ignored=0

managed1.mekawy.com           : ok=3    changed=0    unreachable=0    failed=1    skipped=0
rescued=0    ignored=0

managed2.mekawy.com           : ok=3    changed=0    unreachable=0    failed=1    skipped=0
rescued=0    ignored=0

[ansible@controller errors]$ ls

ansible.cfg cleanweb.yml errorlog force_handlers.yml index.html inventory [ansible@controller errors]$
cat errorlog

2021-05-18 04:37:40,807 p=3053 u=ansible n=ansible | PLAY [force handlers example]
******************************************************************************

*********************************************
```

```
2021-05-18 04:37:40,823 p=3053 u=ansible n=ansible | TASK [Gathering Facts]
***************************************************************************************
********************************************************

2021-05-18 04:37:43,547 p=3053 u=ansible n=ansible | ok: [managed1.mekawy.com] 2021-05-18
04:37:43,572 p=3053 u=ansible n=ansible | ok: [managed2.mekawy.com] 2021-05-18 04:37:43,622
p=3053 u=ansible n=ansible | ok: [controller.mekawy.com]

2021-05-18 04:37:43,645 p=3053 u=ansible n=ansible | TASK [install httpd]
***************************************************************************************
**********************************************************

2021-05-18 04:38:21,400 p=3053 u=ansible n=ansible | ok: [controller.mekawy.com] 2021-05-18
04:38:23,275 p=3053 u=ansible n=ansible | ok: [managed2.mekawy.com] 2021-05-18 04:38:29,832
p=3053 u=ansible n=ansible | ok: [managed1.mekawy.com]

2021-05-18 04:38:29,851 p=3053 u=ansible n=ansible | TASK [copy index.html]
***************************************************************************************
********************************************************

2021-05-18 04:38:32,541 p=3053 u=ansible n=ansible | ok: [managed1.mekawy.com] 2021-05-18
04:38:32,547 p=3053 u=ansible n=ansible | ok: [managed2.mekawy.com] 2021-05-18 04:38:32,569
p=3053 u=ansible n=ansible | ok: [controller.mekawy.com]

2021-05-18 04:38:32,592 p=3053 u=ansible n=ansible | TASK [copy nothing intented to fail]
***************************************************************************************
****************************************

2021-05-18 04:38:32,885 p=3053 u=ansible n=ansible | An exception occurred during task execution. To see
the full traceback, use -vvv. The error was: If you are using a module and expect the file to exist on the
remote, see the remote_src option
```

```
2021-05-18 04:38:32,885 p=3053 u=ansible n=ansible | fatal: [managed1.mekawy.com]: FAILED! =>
{"changed": false, "msg": "Could not find or access '/tmp/nothing' on the Ansible Controller.\nIf you are using
a module and expect the file to exist on the remote, see the remote_src option"}

2021-05-18 04:38:32,899 p=3053 u=ansible n=ansible | An exception occurred during task execution. To see
the full traceback, use -vvv. The error was: If you are using a module and expect the file to exist on the
remote, see the remote_src option

2021-05-18 04:38:32,900 p=3053 u=ansible n=ansible | fatal: [managed2.mekawy.com]: FAILED! =>
{"changed": false, "msg": "Could not find or access '/tmp/nothing' on the Ansible Controller.\nIf you are using
a module and expect the file to exist on the remote, see the remote_src option"}

2021-05-18 04:38:32,959 p=3053 u=ansible n=ansible | An exception occurred during task execution. To see
the full traceback, use -vvv. The error was: If you are using a module and expect the file to exist on the
remote, see the remote_src option

2021-05-18 04:38:32,959 p=3053 u=ansible n=ansible | fatal: [controller.mekawy.com]: FAILED! => {"changed":
false, "msg": "Could not find or access '/tmp/nothing' on the Ansible Controller.\nIf you are using a module
and expect the file to exist on the remote, see the remote_src option"}

2021-05-18 04:38:32,961 p=3053 u=ansible n=ansible | PLAY RECAP
******************************************************************************

*****************************************************************

2021-05-18 04:38:32,961 p=3053 u=ansible n=ansible | controller.mekawy.com : ok=3 changed=0
                unreachable=0     failed=1    skipped=0    rescued=0    ignored=0

2021-05-18 04:38:32,961 p=3053 u=ansible n=ansible | managed1.mekawy.com                  : ok=3
changed=0    unreachable=0     failed=1    skipped=0    rescued=0    ignored=0

2021-05-18 04:38:32,961 p=3053 u=ansible n=ansible | managed2.mekawy.com                  : ok=3
changed=0    unreachable=0     failed=1    skipped=0    rescued=0    ignored=0
```

4.2.2 Troubleshooting Playbooks

1. Connectivity errors, which are issues with managed servers. If you have a connectivity error, then the problem is not really in the playbook, the problem's on the server. Typically, connectivity errors arise when a server is not reachable, and when, for example, you are using wrong credentials.

2. If the error is in the playbook itself, then a nice start is to use ansible-playbook --syntax-check on your playbook to check. And if any errors were found, then you can check the output of this commands.

3. Also, very important is to analyze output messages that are provided with the ansible-playbook command. It basically the best starting point for troubleshooting issues. What is in the output?

   3.1 The play header: The play header shows which play is executed.

   3.2 The task header: shows which tasks

   3.3 The play recap: is giving a play summary.

4. Now by default you get quite some information. If you want more information

   4.1 -vshows output data

   4.2 -vv shows output and input data

   4.3 -vvv includes information about connections to managed hosts

   4.4 -vvvv include information about everything, including users involved, scripts that have been executed.

5. Adding the option --step, which will execute steps interactively and one by one, so that you can easily follow everything that's happening. i.e. ansible-playbook --step myplaybook.yml

6. Use --start-at-task option to start the execution of the playbook at specific task. i.e. ansible-playbook myplaybook.yml --start-at-task ="install vsftpd service"

Some best practices in playbook development

1. You should use -name in the task to describe the task purpose.

2. Include comments, and do include comments a lot so that you can understand what you were trying to accomplish, because playbooks can get quite complicated.

3. Use white space as well in the playbook for increased readability.

4. Work with small, manageable playbooks and use notable includes. Basically you should use roles.

debug: module

The debug module can be used in the playbook to analyze what variables are doing.

You can use the msg: statement to show any information #example

```
- debug:
    msg: "The amount of free memory is {{ ansible_memfree_mb}}"
```

var:output statement

the var:output statement, which can be used to see the output of a specific variable. #example

```
- debug:
    var: output
   verbosity: 2 #will set the verbosity level at to two, which is equal to -vv
```

4.2.3 Troubleshooting Managed Hosts

1. The ansible-playbook command has a --check option, which will perform a test without modifying anything on the target host.i.e ansible-playbook --check myplaybook.yml The ansible modules that you are using must offer support though, for the check module and it makes it kind of hard to use in some cases. And if no support is offered, nothing will be shown.

2. Use check_mode, prior to release 2.2 it was known as always-run, to specify for individual tasks, if it needs to be executed in check_mode.

2.1 check_mode: yes if this task needs to be executed

2.2 check_mode: no if this task doesn't need to be executed

3. Use --check --diff to see all changes that will be applied to managed hosts using Jinja2 templates.

4. Some modules provide additional information about host status.

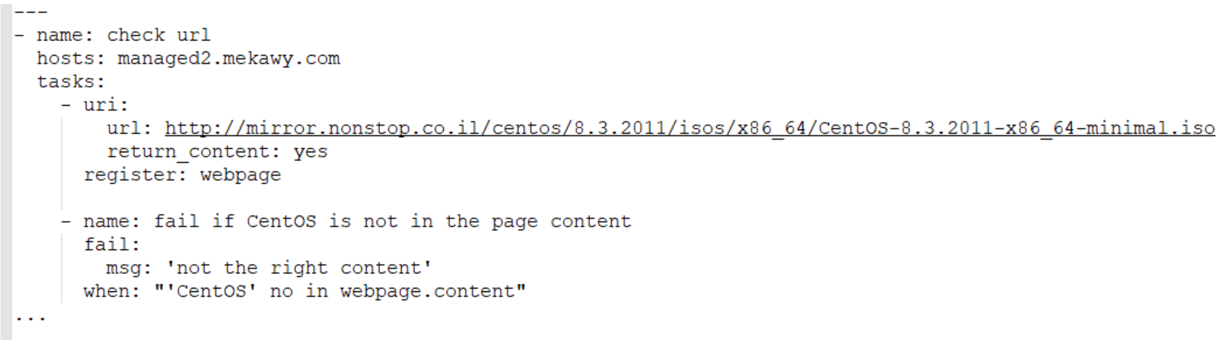
4.1 The uri module, for example, can connect to a URL and check for specific content. So, you can find out what is returned by using this module.

4.2 The script module supports execution of scripts on managed hosts. You should note that the script module must be on the control node and will be transferred and executed on the managed host. So, the script itself is a file that must be available on you local machine. And using the script module makes it possible to do very specific, script based tasks on a managed host.

4.3 The stat module can check that files are present.And the assert module can work on the outcome of the stat module.

#example

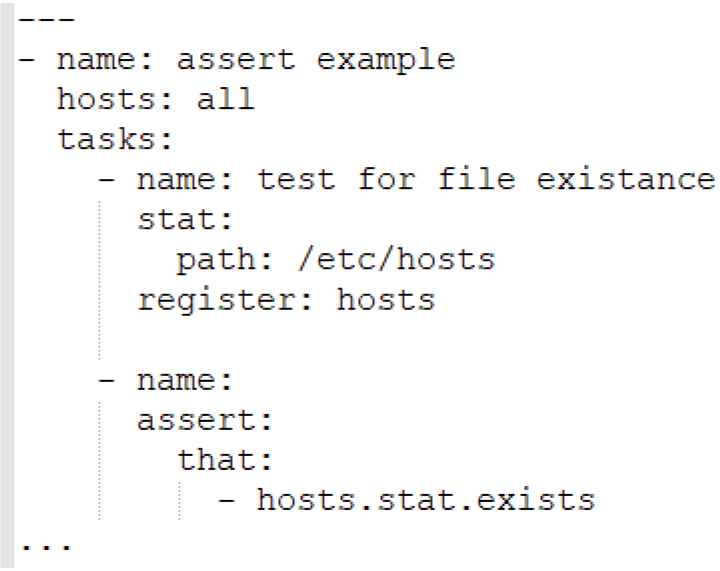
[ansible@controller errors]$ vim uricheck.yml



```
---
- name: check url
  hosts: managed2.mekawy.com
  tasks:
    - uri:
        url: http://mirror.nonstop.co.il/centos/8.3.2011/isos/x86_64/CentOS-8.3.2011-x86_64-minimal.iso
        return_content: yes
      register: webpage

    - name: fail if CentOS is not in the page content
      fail:
        msg: 'not the right content'
      when: "'CentOS' no in webpage.content"
...
```

#example

[ansible@controller errors]$ vim assert.yml



```
---
- name: assert example
  hosts: all
  tasks:
    - name: test for file existance
      stat:
        path: /etc/hosts
      register: hosts

    - name:
      assert:
        that:
          - hosts.stat.exists
...
```

[ansible@controller errors]$ ansible-playbook assert.yml PLAY

[assert example]

****

****

TASK [Gathering Facts]

****

*** ok:

[managed1.mekawy.com]

```
ok: [managed2.mekawy.com] ok:
[controller.mekawy.com]

TASK [test for file existance]
***********************************************************************************
******************************************** ok:
[managed1.mekawy.com]
ok: [managed2.mekawy.com] ok:
[controller.mekawy.com]

TASK [assert]
***********************************************************************************
************************************************************** ok:
[controller.mekawy.com] => {
    "changed": false,

    "msg": "All assertions passed"

}

ok:[managed1.mekawy.com]=>{
    "changed": false,

    "msg": "All assertions passed"

}

ok:[managed2.mekawy.com]=>{
    "changed": false,

    "msg": "All assertions passed"

}
```

```
PLAY RECAP
*************************************************************************************

********************************************************************

controller.mekawy.com         : ok=3    changed=0     unreachable=0     failed=0    skipped=0
rescued=0     ignored=0

managed1.mekawy.com              : ok=3    changed=0     unreachable=0   failed=0      skipped=0
rescued=0     ignored=0


managed2.mekawy.com              : ok=3    changed=0     unreachable=0   failed=0      skipped=0

rescued=0     ignored=0
```

Thanks

References

1. https://docs.ansible.com/
2. Automating with Ansible by Sander Van Vugt Course
3. https://github.com/ansible/ansible/tree/devel/examples
4. https://github.com/ansible/ansible/
5. https://galaxy.ansible.com/