

<b>Name: Gayao, Froilan M.</b>	<b>Date Performed:10/04/24</b>
<b>Course/Section: CPE31S4</b>	<b>Date Submitted:10/04/24</b>
<b>Instructor: Engr. Robin Valenzuela</b>	<b>Semester and SY: 1st 24 -25</b>
<b>Activity 6: Targeting Specific Nodes and Managing Services</b>	
<p><b>1. Objectives:</b></p> <ul style="list-style-type: none"> <li>1.1 Individualize hosts</li> <li>1.2 Apply tags in selecting plays to run</li> <li>1.3 Managing Services from remote servers using playbooks</li> </ul>	
<p><b>2. Discussion:</b></p> <p>In this activity, we try to individualize hosts. For example, we don't want apache on all our servers, or maybe only one of our servers is a web server, or maybe we have different servers like database or file servers running different things on different categories of servers and that is what we are going to take a look at in this activity.</p> <p>We also try to manage services that do not automatically run using the automations in playbook. For example, when we install web servers or httpd for CentOS, we notice that the service did not start automatically.</p> <p><b>Requirement:</b></p> <p>In this activity, you will need to create another Ubuntu VM and name it Server 3. Likewise, you need to activate the second adapter to a host-only adapter after the installations. Take note of the IP address of the Server 3. Make sure to use the command <i>ssh-copy-id</i> to copy the public key to Server 3. Verify if you can successfully SSH to Server 3.</p>	
<b>Task 1: Targeting Specific Nodes</b>	
<ul style="list-style-type: none"> <li>1. Create a new playbook and named it site.yml. Follow the commands as shown in the image below. Make sure to save the file and exit.</li> </ul>	

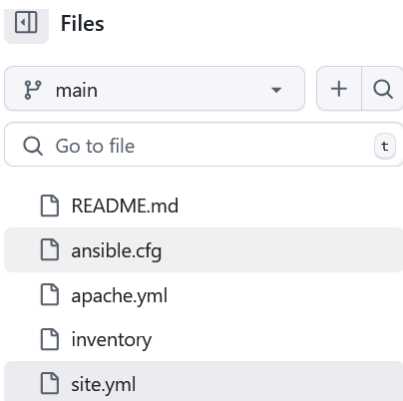
```

---
- hosts: all
  become: true
  tasks:

    - name: install apache and php for Ubuntu servers
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
        update_cache: yes
      when: ansible_distribution == "Ubuntu"

    - name: install apache and php for CentOS servers
      dnf:
        name:
          - httpd
          - php
        state: latest
      when: ansible_distribution == "CentOS"

```



ACT6 / site.yml

PooKYZZZ Create site.yml

Code Blame 20 lines (19 loc) · 445 Bytes

```

1  ---
2  - hosts: all
3    become: true
4    tasks:
5      - name: Install Apache and PHP for Ubuntu servers
6        apt:
7          name:
8            - apache2
9            - libapache2-mod-php
10         state: latest
11         update_cache: yes
12         when: ansible_distribution == "Ubuntu"
13
14     - name: Install Apache and PHP for CentOS servers
15       dnf:
16         name:
17           - httpd
18           - php
19         state: latest
20         when: ansible_distribution == "CentOS"

```

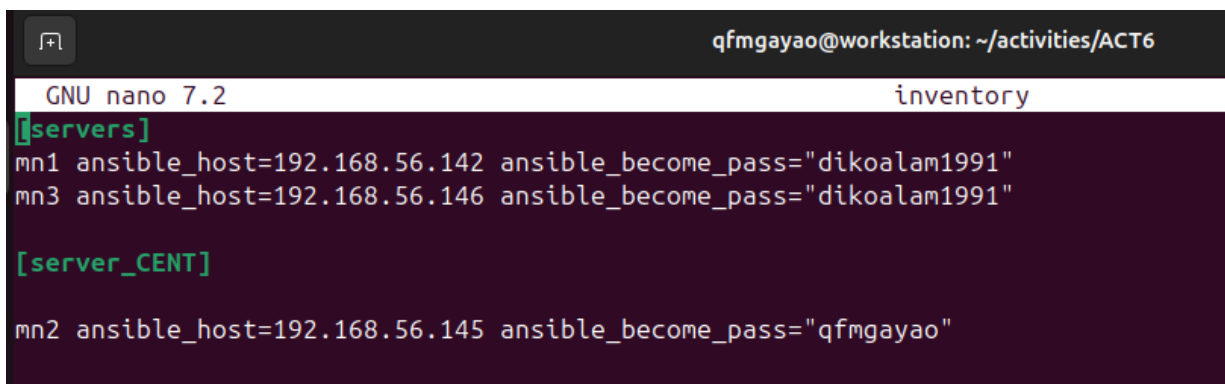
In this, I created a new yml which we will use for this activity, also instead of making the yml inside the ubuntu, I will just git pull the edit from github to my local repository.

2. Edit the inventory file. Remove the variables we put in our last activity and group according to the image shown below:

```
[web_servers]
192.168.56.120
192.168.56.121

[db_servers]
192.168.56.122

[file_servers]
192.168.56.123
```



```
qfmgayao@workstation: ~/activities/ACT6
GNU nano 7.2 inventory
[servers]
mn1 ansible_host=192.168.56.142 ansible_become_pass="dikoalam1991"
mn3 ansible_host=192.168.56.146 ansible_become_pass="dikoalam1991"

[server_CENT]
mn2 ansible_host=192.168.56.145 ansible_become_pass="qfmgayao"
```

In this, I don't have a file server because I only have 3 servers which are 2 ubuntu servers, and 1 centOS

Make sure to save the file and exit.

Right now, we have created groups in our inventory file and put each server in its own group. In other cases, you can have a server be a member of multiple groups, for example you have a test server that is also a web server.

3. Edit the *site.yml* by following the image below:

```

---
- hosts: all
  become: true
  pre_tasks:
    - name: install updates (CentOS)
      dnf:
        update_only: yes
        update_cache: yes
        when: ansible_distribution == "CentOS"

    - name: install updates (Ubuntu)
      apt:
        upgrade: dist
        update_cache: yes
        when: ansible_distribution == "Ubuntu"

- hosts: web_servers
  become: true
  tasks:
    - name: install apache and php for Ubuntu servers
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
        when: ansible_distribution == "Ubuntu"

    - name: install apache and php for CentOS servers
      dnf:
        name:
          - httpd
          - php
        state: latest
        when: ansible_distribution == "CentOS"

```

Make sure to save the file and exit.

The *pre-tasks* command tells the ansible to run it before any other thing. In the *pre-tasks*, CentOS will install updates while Ubuntu will upgrade its distribution package. This will run before running the second play, which is targeted at *web\_servers*. In the second play, apache and php will be installed on both Ubuntu servers and CentOS servers.

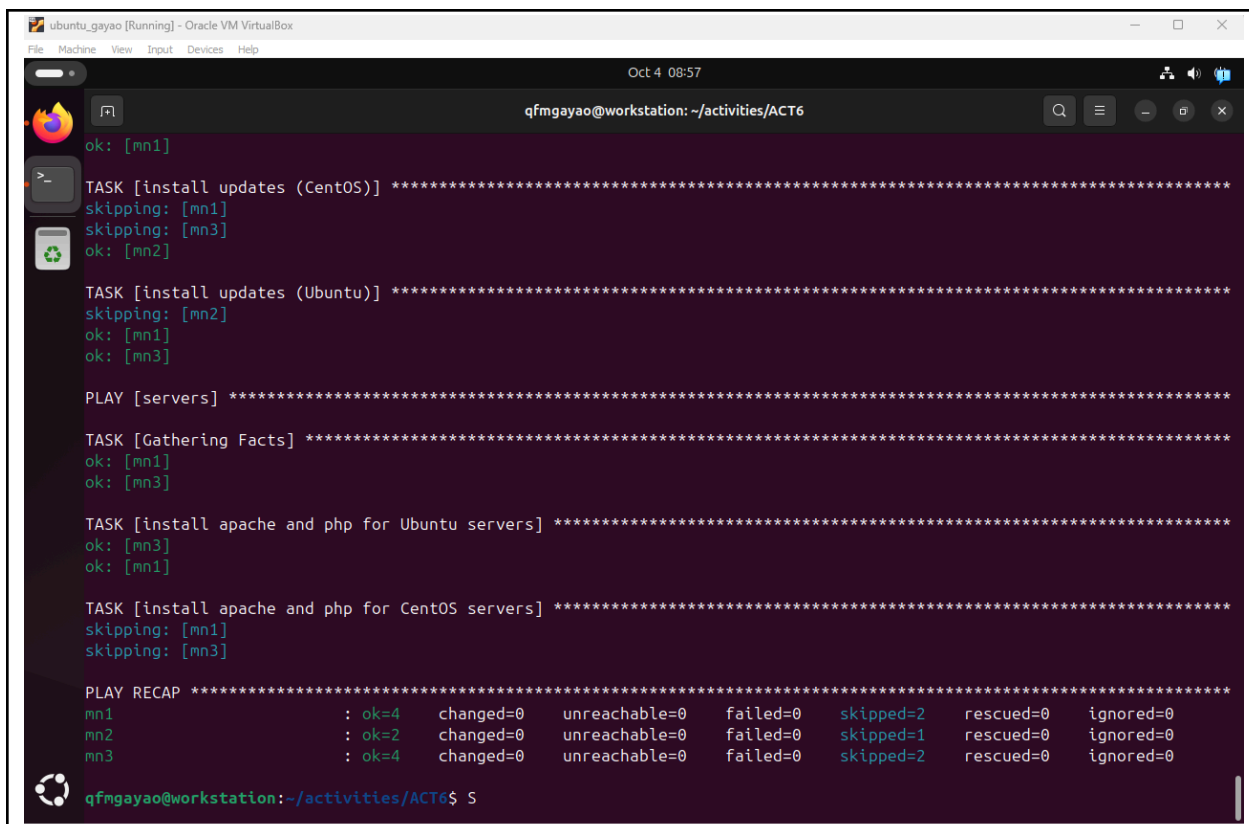
Run the *site.yml* file and describe the result.

```
hosts: all
become: true
pre_tasks:
  - name: install updates (CentOS)
    dnf:
      update_only: yes
      update_cache: yes
      when: ansible_distribution == "CentOS"

  - name: install updates (Ubuntu)
    apt:
      upgrade: dist
      update_cache: yes
      when: ansible_distribution == "Ubuntu"

hosts: servers
become: true
tasks:
  - name: install apache and php for Ubuntu servers
    apt:
      name:
        - apache2
        - libapache2-mod-php
      state: latest
      when: ansible_distribution == "Ubuntu"

  - name: install apache and php for CentOS servers
    dnf:
      name:
        - httpd
        - php
      state: latest
      when: ansible_distribution == "CentOS"
```



```
qfmgayao@workstation: ~/activities/ACT6
ok: [mn1]
TASK [install updates (CentOS)] *****
skipping: [mn1]
skipping: [mn3]
ok: [mn2]
TASK [install updates (Ubuntu)] *****
skipping: [mn2]
ok: [mn1]
ok: [mn3]
PLAY [servers] *****
TASK [Gathering Facts] *****
ok: [mn1]
ok: [mn3]
TASK [install apache and php for Ubuntu servers] *****
ok: [mn3]
ok: [mn1]
TASK [install apache and php for CentOS servers] *****
skipping: [mn1]
skipping: [mn3]
PLAY RECAP *****
mn1      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn2      : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn3      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
qfmgayao@workstation: ~/activities/ACT6$
```

I now have a playbook for Ansible. It does two things. First, if server CentOS, it uses dnf for updates. If Ubuntu, it uses apt for upgrade. Second, for my server, it install apache2 and libapache2-mod-php on Ubuntu. For CentOS, it install httpd and php, but in my server group I only have the ip address of my ubuntu in server so it will skip the httpd.

- Let's try to edit again the *site.yml* file. This time, we are going to add plays targeting the other servers. This time we target the *db\_servers* by adding it on the current *site.yml*. Below is an example: (Note add this at the end of the playbooks from task 1.3.

```
- hosts: db_servers
  become: true
  tasks:

    - name: install mariadb package (CentOS)
      yum:
        name: mariadb-server
        state: latest
      when: ansible_distribution == "CentOS"

    - name: "Mariadb- Restarting/Enabling"
      service:
        name: mariadb
        state: restarted
        enabled: true

    - name: install mariadb package (Ubuntu)
      apt:
        name: mariadb-server
        state: latest
      when: ansible_distribution == "Ubuntu"
```

Make sure to save the file and exit.

Run the *site.yml* file and describe the result.

Files

main

Go to file

README.md

ansible.cfg

apache.yml

inventory

site.yml

ACT6 / site.yml54 lines (48 loc) · 1.19 KBCodeBlame

```
30     - httpd
31     - php
32       state: latest
33       when: ansible_distribution == "CentOS"
34
35 - hosts: server_CENT
36   become: true
37   tasks:
38     - name: install mariadb package (CentOS)
39       yum:
40         name: mariadb-server
41         state: latest
42         when: ansible_distribution == "CentOS"
43
44     - name: Mariadb Restarting/Enabling
45       service:
46         name: mariadb
47         state: restarted
48         enabled: true
49
50     - name: install mariadb package (Ubuntu)
51       apt:
52         name: mariadb-server
53         state: latest
54         when: ansible_distribution == "Ubuntu"
```

In this, I just git pull my code then run my playbook, it installs the latest mariadb-server with yum. Then, it restarts MariaDB service and makes sure it starts when boot up and it skips the option for Ubuntu, it also skips the other task because it's designated to ubuntu. This way, your database server gets updated MariaDB.

5. Go to the remote server (Ubuntu) terminal that belongs to the db\_servers group and check the status for mariadb installation using the command: *systemctl status mariadb*. Do this on the CentOS server also.

Describe the output.



```

PLAY [servers] *****
TASK [Gathering Facts] *****
ok: [mn3]
ok: [mn1]

TASK [install apache and php for Ubuntu servers] *****
ok: [mn1]
ok: [mn3]

TASK [install apache and php for CentOS servers] *****
skipping: [mn1]
skipping: [mn3]

PLAY [server_CENT] *****
TASK [Gathering Facts] *****
ok: [mn2]

TASK [install mariadb package (CentOS)] *****
changed: [mn2]

TASK [Mariadb Restarting/Enabling] *****
changed: [mn2]

TASK [install mariadb package (Ubuntu)] *****
skipping: [mn2]

PLAY RECAP *****
mn1      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn2      : ok=5    changed=2    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn3      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn4      : ok=2    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

qfmgayao@workstation:~/activities/ACT6$

```

```

[qfmgayao@mn2 ~]$ systemctl status mariadb
● mariadb.service - MariaDB 10.5 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: >
   Active: active (running) since Fri 2024-10-04 10:17:42 PST; 1min 24s ago
     Docs: man:mariadbd(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 7435 ExecStartPre=/usr/libexec/mariadb-check-socket (code=exited, >
   Process: 7457 ExecStartPre=/usr/libexec/mariadb-prepare-db-dir mariadb.serv>
   Process: 7571 ExecStartPost=/usr/libexec/mariadb-check-upgrade (code=exited>
  Main PID: 7547 (mariadbd)
    Status: "Taking your SQL requests now..."
     Tasks: 8 (limit: 20510)
    Memory: 77.8M
       CPU: 589ms
    CGroup: /system.slice/mariadb.service
            └─7547 /usr/libexec/mariadbd --basedir=/usr

Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: The second is mysql@localhost>
Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: you need to be the system 'my>
Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: After connecting you can set >
Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: able to connect as any of the>
Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: See the MariaDB Knowledgebase>
Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: Please report any problems at>
Oct 04 10:17:42 mn2 mariadb-prepare-db-dir[7498]: The latest information about >

```

on CentOS server my playbook does this first, it installs the latest mariadb-server with yum. Then, it restarts MariaDB service and makes sure it starts when boot up and it skips the option for Ubuntu, it also skips the other task because it's designated to ubuntu. This way, your database server gets updated MariaDB.

Edit the *site.yml* again. This time we will append the code to configure installation on the *file\_servers* group. We can add the following on our file.

```
- hosts: file_servers
  become: true
  tasks:

    - name: install samba package
      package:
        name: samba
        state: latest
```

Make sure to save the file and exit.

Run the *site.yml* file and describe the result.

Files

main

Go to file

README.md

ansible.cfg

apache.yml

inventory

site.yml

ACT6 / site.yml

CodeBlame63 lines (55 loc) · 1.33 KB

```
39     yum:
40         name: mariadb-server
41         state: latest
42         when: ansible_distribution == "CentOS"
43
44     - name: Mariadb Restarting/Enabling
45       service:
46         name: mariadb
47         state: restarted
48         enabled: true
49
50     - name: install mariadb package (Ubuntu)
51       apt:
52         name: mariadb-server
53         state: latest
54         when: ansible_distribution == "Ubuntu"
55
56
57     - hosts: fileserver
58       become: true
59       tasks:
60         - name: install samba package
61           package:
62             name: samba
63             state: latest
```

```
PLAY [fileserver] *****
TASK [Gathering Facts] *****
ok: [mn4]

TASK [install samba package] *****
changed: [mn4]

PLAY RECAP *****
mn1      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn2      : ok=5    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn3      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn4      : ok=4    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

qfngayao@workstation:~/activities/ACT6$
```

So in this line of code, I installed the samba package and according to my search, samba is used for efficient file sharing which will make things easier and faster.

The testing of the *file\_servers* is beyond the scope of this activity, and as well as our topics and objectives. However, in this activity we were able to show that we can target hosts or servers using grouping in ansible playbooks.

## Task 2: Using Tags in running playbooks

In this task, our goal is to add metadata to our plays so that we can only run the plays that we want to run, and not all the plays in our playbook.

1. Edit the *site.yml* file. Add tags to the playbook. After the name, we can place the tags: *name\_of\_tag*. This is an arbitrary command, which means you can use any name for a tag.

```
---  
  
- hosts: all  
  become: true  
  pre_tasks:  
  
    - name: install updates (CentOS)  
      tags: always  
      dnf:  
        update_only: yes  
        update_cache: yes  
        when: ansible_distribution == "CentOS"  
  
    - name: install updates (Ubuntu)  
      tags: always  
      apt:  
        upgrade: dist  
        update_cache: yes  
        when: ansible_distribution == "Ubuntu"
```

```
- hosts: web_servers
  become: true
  tasks:

    - name: install apache and php for Ubuntu servers
      tags: apache,apache2,ubuntu
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
      when: ansible_distribution == "Ubuntu"

    - name: install apache and php for CentOS servers
      tags: apache,centos,httpd
      dnf:
        name:
          - httpd
          - php
        state: latest
      when: ansible_distribution == "CentOS"
```

```
- hosts: db_servers
  become: true
  tasks:

    - name: install mariadb package (CentOS)
      tags: centos, db, mariadb
      dnf:
        name: mariadb-server
        state: latest
        when: ansible_distribution == "CentOS"

    - name: "Mariadb- Restarting/Enabling"
      service:
        name: mariadb
        state: restarted
        enabled: true

    - name: install mariadb package (Ubuntu)
      tags: db, mariadb, ubuntu
      apt:
        name: mariadb-server
        state: latest
        when: ansible_distribution == "Ubuntu"

- hosts: file_servers
  become: true
  tasks:

    - name: install samba package
      tags: samba
      package:
        name: samba
        state: latest
```

Make sure to save the file and exit.

Run the *site.yml* file and describe the result.

```
- hosts: all
  become: true
  pre_tasks:
    - name: install updates (CentOS)
      tags: always
      dnf:
        update_only: yes
        update_cache: yes
      when: ansible_distribution == "CentOS"

    - name: install updates (Ubuntu)
      tags: always
      apt:
        upgrade: dist
        update_cache: yes
      when: ansible_distribution == "Ubuntu"
```

```
- hosts: servers
  become: true
  tasks:

    - name: install apache and php for Ubuntu serve
      tags: apache, apache2, ubuntu
      apt:
        name:
          - apache2
          - libapache2-mod-php
        state: latest
      when: ansible_distribution == "Ubuntu"

    - name: install apache and php for CentOS serve
      tags: apache, apache2, ubuntu
      dnf:
        name:
          - httpd
          - php
        state: latest
      when: ansible_distribution == "CentOS"
```



```
40     - hosts: server_CENT
41       become: true
42       tasks:
43
44         - name: install mariadb package (CentOS)
45           tags: centos, db, mariadb
46           dnf:
47             name: mariadb-server
48             state: latest
49             when: ansible_distribution == "CentOS"
50
51         - name: Mariadb Restarting/Enabling
52           tags: db, mariadb, ubuntu
53           service:
54             name: mariadb
55             state: restarted
56             enabled: true
57
58         - name: install mariadb package (Ubuntu)
59           apt:
60             name: mariadb-server
61             state: latest
62             when: ansible_distribution == "Ubuntu"
63
64
65     - hosts: fileserver
66       become: true
67       tasks:
68
69         - name: install samba package
70           tags: samba
71           package:
72             name: samba
73             state: latest
```

```

TASK [install apache and php for Ubuntu servers] *****
ok: [mn3]
ok: [mn1]

TASK [install apache and php for CentOS servers] *****
skipping: [mn1]
skipping: [mn3]

PLAY [server_CENT] *****

TASK [Gathering Facts] *****
ok: [mn2]

TASK [install mariadb package (CentOS)] *****
ok: [mn2]

TASK [Mariadb Restarting/Enabling] *****
changed: [mn2]

TASK [install mariadb package (Ubuntu)] *****
skipping: [mn2]

PLAY [fileserver] *****

TASK [Gathering Facts] *****
ok: [mn4]

TASK [install samba package] *****
ok: [mn4]

PLAY RECAP *****
mn1      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn2      : ok=5    changed=1    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn3      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn4      : ok=4    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

qfmgayao@workstation:~/activities/ACT6$ s

```

Nothing changes when I run the playbook with the new code which is tag, from what I search what it's use, it only works when you call it out.

2. On the local machine, try to issue the following commands and describe each result:

2.1 *ansible-playbook --list-tags site.yml*

```
qfmgayao@workstation:~/activities/ACT6$ ansible-playbook --list-tags site.yml
playbook: site.yml

play #1 (all): all    TAGS: []
TASK TAGS: [always]

play #2 (servers): servers    TAGS: []
TASK TAGS: [apache, apache2, ubuntu]

play #3 (server_CENT): server_CENT    TAGS: []
TASK TAGS: [centos, db, mariadb, ubuntu]

play #4 (fileserver): fileserver    TAGS: []
TASK TAGS: [samba]
qfmgayao@workstation:~/activities/ACT6$
```

for this command, it list all task that have task tags which we can see and we can select which one to run without running the entire playbook.

2.2 *ansible-playbook --tags centos --ask-become-pass site.yml*

```

TASK [install updates (CentOS)] *****
skipping: [mn1]
skipping: [mn3]
skipping: [mn4]
ok: [mn2]

TASK [install updates (Ubuntu)] *****
skipping: [mn2]
ok: [mn3]
ok: [mn4]
ok: [mn1]

PLAY [servers] *****

TASK [Gathering Facts] *****
ok: [mn3]
ok: [mn1]

PLAY [server_CENT] *****

TASK [Gathering Facts] *****
ok: [mn2]

TASK [install mariadb package (CentOS)] *****
ok: [mn2]

PLAY [fileserver] *****

TASK [Gathering Facts] *****
ok: [mn4]

PLAY RECAP *****
mn1      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn2      : ok=4    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn3      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn4      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
qfmgayao@workstation:~/activities/ACT6$

```

For this command line, it triggers the tag for all CentOS which in our result it shows that it works for all CentOS and skips the ubuntu parts.

2.3 *ansible-playbook --tags db --ask-become-pass site.yml*

```

TASK [install updates (Ubuntu)] *****
skipping: [mn2]
ok: [mn4]
ok: [mn3]
ok: [mn1]

PLAY [servers] *****

TASK [Gathering Facts] *****
ok: [mn1]
ok: [mn3]

PLAY [server_CENT] *****

TASK [Gathering Facts] *****
ok: [mn2]

TASK [install mariadb package (CentOS)] *****
ok: [mn2]

TASK [Mariadb Restarting/Enabling] *****
changed: [mn2]

PLAY [fileserver] *****

TASK [Gathering Facts] *****
ok: [mn4]

PLAY RECAP *****
mn1      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn2      : ok=5    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn3      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn4      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

qfmgayao@workstation:~/activities/ACT6$

```

for this command line it only selects the tag with db which in my result it shows my CentOS task being called out.

2.4 *ansible-playbook --tags apache --ask-become-pass site.yml*

```

TASK [Gathering Facts] *****
ok: [mn3]
ok: [mn1]

TASK [install apache and php for Ubuntu servers] *****
ok: [mn3]
ok: [mn1]

TASK [install apache and php for CentOS servers] *****
skipping: [mn1]
skipping: [mn3]

PLAY [server_CENT] *****

TASK [Gathering Facts] *****
ok: [mn2]

PLAY [fileserver] *****

TASK [Gathering Facts] *****
ok: [mn4]

PLAY RECAP *****
mn1      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn2      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn3      : ok=4    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn4      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

qfmgayao@workstation:~/activities/ACT6$ s

```

for this command line it only selects the tag with apache which in my result it shows my ubuntu servers task being called out.

2.5 *ansible-playbook --tags "apache,db" --ask-become-pass site.yml*

```

ok: [mn1]
ok: [mn2]
ok: [mn3]

TASK [install updates (CentOS)] *****
skipping: [mn1]
skipping: [mn3]
skipping: [mn4]
ok: [mn2]

TASK [install updates (Ubuntu)] *****
skipping: [mn2]
ok: [mn1]
ok: [mn4]
ok: [mn3]

PLAY [servers] *****

TASK [Gathering Facts] *****
ok: [mn3]
ok: [mn1]

PLAY [server_CENT] *****

TASK [Gathering Facts] *****
ok: [mn2]

PLAY [fileserver] *****

TASK [Gathering Facts] *****
ok: [mn4]

PLAY RECAP *****
mn1      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn2      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn3      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn4      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

```

for this command line it only selects the tag with apache and db which in my result it shows my ubuntu servers and CentOS task being called out.

### Task 3: Managing Services

1. Edit the file site.yml and add a play that will automatically start the httpd on CentOS server.

```
- name: install apache and php for CentOS servers
  tags: apache,centos,httpd
  dnf:
    name:
      - httpd
      - php
    state: latest
  when: ansible_distribution == "CentOS"

- name: start httpd (CentOS)
  tags: apache, centos,httpd
  service:
    name: httpd
    state: started
  when: ansible_distribution == "CentOS"
```

Figure 3.1.1

Make sure to save the file and exit.

You would also notice from our previous activity that we already created a module that runs a service.

```
- hosts: db_servers
  become: true
  tasks:

  - name: install mariadb package (CentOS)
    tags: centos, db,mariadb
    dnf:
      name: mariadb-server
      state: latest
    when: ansible_distribution == "CentOS"

  - name: "Mariadb- Restarting/Enabling"
    service:
      name: mariadb
      state: restarted
      enabled: true
```

Figure 3.1.2

This is because in CentOS, installed packages' services are not run automatically. Thus, we need to create the module to run it automatically.

2. To test it, before you run the saved playbook, go to the CentOS server and stop the currently running httpd using the command *sudo systemctl stop httpd*.

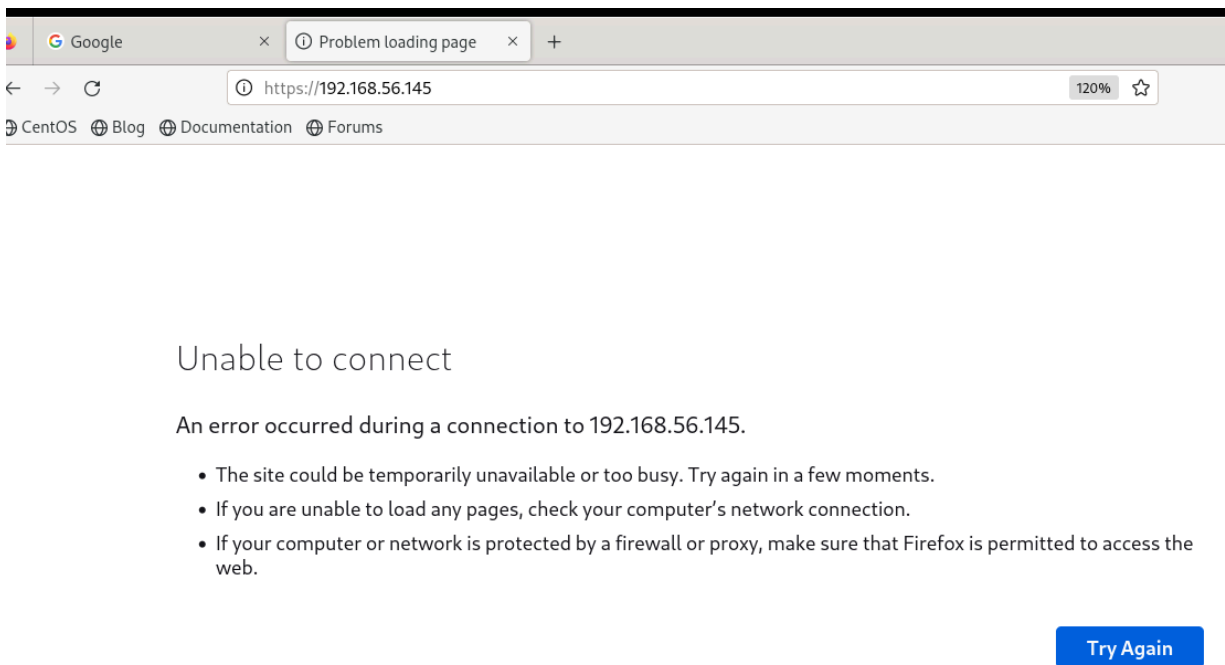


When prompted, enter the sudo password. After that, open the browser and enter the CentOS server's IP address. You should not be getting a display because we stopped the httpd service already.

3. Go to the local machine and this time, run the *site.yml* file. Then after running the file, go again to the CentOS server and enter its IP address on the browser. Describe the result.

To automatically enable the service every time we run the playbook, use the command *enabled: true* similar to Figure 7.1.2 and save the playbook.

```
[qfmgayao@mn2 ~]$ sudo systemctl stop httpd
[sudo] password for qfmgayao:
[qfmgayao@mn2 ~]$
```



```

TASK [Gathering Facts] *****
ok: [mn3]
ok: [mn1]
ok: [mn2]

TASK [start httpd (CentOS)] *****
skipping: [mn1]
skipping: [mn3]
changed: [mn2]

PLAY [server_CENT] *****

TASK [Gathering Facts] *****
ok: [mn2]

PLAY [fileserv] *****

TASK [Gathering Facts] *****
ok: [mn4]

PLAY RECAP *****
mn1      : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn2      : ok=5    changed=1    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
mn3      : ok=3    changed=0    unreachable=0    failed=0    skipped=2    rescued=0    ignored=0
mn4      : ok=3    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0

qfmgayao@workstation:~/activities/ACT6$

```

Google

HTTP Server Test Page

x

+

←

→

↺

🔒

192.168.56.145

120%

☆

CentOS

Blog

Documentation

Forums

## If you are a member of the general public:

The website you just visited is either experiencing problems or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting [www.example.com](http://www.example.com), you should send e-mail to "webmaster@example.com".

### Important Note!

The CentOS Project has nothing to do with this website or its content, it just provides the software that makes the website run.

If you have issues with the content of this site, contact the owner of the domain, not the CentOS

**I edited my playbook to make sure the httpd service starts automatically on my CentOS server. I included a task that ensures the service is enabled and starts on boot.**

**Next, I stopped the currently running httpd service on the CentOS server using `sudo systemctl stop httpd` and checked the browser to see if the web page was no longer working.**

**Then, I ran the playbook from my local machine. After running the playbook, I went back to the CentOS server and entered its IP address in the browser. The web page was displayed again, which meant my playbook successfully started and enabled the httpd service. Now, whenever I run the playbook, httpd will always start up.**

#### **Reflections:**

Answer the following:

1. What is the importance of putting our remote servers into groups?
  - Putting servers together in groups makes things easy for us. We can target specific servers which make codes simpler and easy to debug.
2. What is the importance of tags in playbooks?
  - Tags are very helpful because they let us run only parts of the playbook which can save time, it's also good for testing because we can set what to run only by using tags which makes things easy.
  -
3. Why do you think some services need to be managed automatically in playbooks?
  - This ensures the system reliability and uptime by our codes. By defining service states in playbooks we minimize manual inputs or intervention which can prevent downtime, and maintain consistent service availability.

**<https://github.com/PooKYZZZ/ACT6>**

