Lab: Working with Tasks

Introduction:

A task is the smallest unit of action you can automate using an Ansible playbook.

Playbooks typically contain a series of tasks that serve a goal, such as to set up a web server, or to deploy an application to remote environments.

Ansible executes tasks in the same order they are defined inside a playbook.

Objectives:

- **Learning to use ansible-doc**
- **Writing Task to Install httpd Service**
- **Writing Task to Start httpd Service**
- **Verifying httpd service by running ad-hoc command**

1. Learning to use ansible-doc

1.1 Let's create the tasks which create file on managed host.

Before jumping on creating a task lets learn how to take a help from **ansible-doc** which will give a good idea about yum Module and Examples of some task.

```
# ansible-doc yum | grep -i EXAMPLES -A 40
```

Output:

```
[admin@eoc-controller ~]$ ansible-doc yum | grep -i EXAMPLES -A 40
EXAMPLES:

- name: Install the latest version of Apache
  ansible.builtin.yum:
    name: httpd
    state: latest

- name: Install Apache >= 2.4
  ansible.builtin.yum:
    name: httpd>=2.4
    state: present

- name: Install a list of packages (suitable replacement for 2.11 loop deprecation wa
rning)
  ansible.builtin.yum:
    name:
      - nginx
      - postgresql
      - postgresql-server
    state: present

- name: Install a list of packages with a list variable
  ansible.builtin.yum:
    name: "{{ packages }}"
  vars:
    packages:
    - httpd
    - httpd-tools

- name: Remove the Apache package
  ansible.builtin.yum:
    name: httpd
    state: absent

- name: Install the latest version of Apache from the testing repo
  ansible.builtin.yum:
    name: httpd
    enablerepo: testing
    state: present

- name: Install one specific version of Apache
  ansible.builtin.yum:
```

2. Writing Task to Install httpd Service.

2.1 Let's install the httpd package on the webserver group (Managed hosts) in the file name **task1.yml**

Let's select the host group on which installation to be done.

```
- hosts: webservers
  become: yes
```

2.2 Let's create a task which install the httpd package.

```
tasks:
  - name: Install the latest version of Apache
    ansible.builtin.yum:
      name: httpd
      state: latest
```

2.3 Let's view the yaml manifest.

```
# cat -n task1.yml
```

Output:

```
[admin@eoc-controller ~]$ cat -n task1.yml
     1  - hosts: webservers
     2    become: yes
     3    tasks:
     4      - name: Install the latest version of Apache
     5        ansible.builtin.yum:
     6          name: httpd
     7          state: latest
```

2.4 Let's perform a syntax check **task1.yml**.

```
# ansible-playbook --syntax-check task1.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook --syntax-check task1.yml

playbook: task1.yml
```

2.5 Let's run the task which install the httpd package on webserver hosts.

```
# ansible-playbook task1.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook  task1.yml

PLAY [webservers] ***************************************************************

TASK [Gathering Facts] **********************************************************
ok: [eoc-node3]

TASK [Install the latest version of Apache] *************************************
changed: [eoc-node3]

PLAY RECAP **********************************************************************
eoc-node3                  : ok=2    changed=1    unreachable=0    failed=0    skippe
d=0    rescued=0    ignored=0
```

2.6 Let's take help from ansible-doc to know some of the service-related examples where we get an idea how to {**enable, starts, restart, disable**} the services running on the managed hosts.

```
# ansible-doc service | grep -I EXAMPLES -A 40
```

Output:

```
[admin@eoc-controller ~]$ ansible-doc service | grep -I EXAMPLES -A 40
EXAMPLES:

- name: Start service httpd, if not started
  ansible.builtin.service:
    name: httpd
    state: started

- name: Stop service httpd, if started
  ansible.builtin.service:
    name: httpd
    state: stopped

- name: Restart service httpd, in all cases
  ansible.builtin.service:
    name: httpd
    state: restarted

- name: Reload service httpd, in all cases
  ansible.builtin.service:
    name: httpd
    state: reloaded

- name: Enable service httpd, and not touch the state
  ansible.builtin.service:
    name: httpd
    enabled: yes

- name: Start service foo, based on running process /usr/bin/foo
  ansible.builtin.service:
    name: foo
    pattern: /usr/bin/foo
    state: started

- name: Restart network service for interface eth0
  ansible.builtin.service:
    name: network
    state: restarted
    args: eth0
```

3. Writing Task to Start httpd Service.

3.1 In the file **task2.yml** Let's enable and start the httpd service on the **webserver** group (Managed hosts)

```
- hosts: webservers
  become: yes
```

```
tasks:
  - name: httpd is started
    ansible.builtin.service:
      name: httpd
      enabled: true
      state: started
```

3.2 Let's verify the **task2.yml** manifest.

```
# cat -n task2.yml
```

Output:

```
[admin@eoc-controller ~]$ cat -n task2.yml
     1  - hosts: webservers
     2    become: yes
     3    tasks:
     4      - name: httpd is started
     5        ansible.builtin.service:
     6          name: httpd
     7          enabled: true
     8          state: started
```

3.3 Let's perform a syntax check of the **task2.yml** manifest.

```
# ansible-playbook --syntax-check task2.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook  --syntax-check task2.yml

playbook: task2.yml
```

3.4 Let's run the task which enable the httpd service in **webserver** group.

```
# ansible-playbook task2.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook task2.yml

PLAY [webservers] **************************************************************

TASK [Gathering Facts] *********************************************************
ok: [eoc-node3]

TASK [httpd is started] ********************************************************
changed: [eoc-node3]

PLAY RECAP *********************************************************************
eoc-node3                  : ok=2    changed=1    unreachable=0    failed=0    skippe
d=0    rescued=0    ignored=0
```

4. Verifying httpd service by running ad-hoc command.

4.1 let's verify the execution of task on webserver using ad-hoc command.

```
# ansible webservers -m command -a " systemctl status
httpd.service"
```

Output:

```
[admin@eoc-controller ~]$ansible webservers -m command -a " systemctl status httpd.service"
eoc-node3 | CHANGED | rc=0 >>
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2024-01-28 04:48:53 EST; 1min 10s ago
     Docs: man:httpd.service(8)
 Main PID: 19167 (httpd)
   Status: "Running, listening on: port 80"
    Tasks: 213 (limit: 22885)
   Memory: 37.6M
   CGroup: /system.slice/httpd.service
           ├─19167 /usr/sbin/httpd -DFOREGROUND
           ├─19168 /usr/sbin/httpd -DFOREGROUND
           ├─19169 /usr/sbin/httpd -DFOREGROUND
           ├─19170 /usr/sbin/httpd -DFOREGROUND
           └─19171 /usr/sbin/httpd -DFOREGROUND

Jan 28 04:48:53 eoc-node3 systemd[1]: Starting The Apache HTTP Server...
Jan 28 04:48:53 eoc-node3 httpd[19167]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.100.153. Set the 'ServerName' directive globally to suppress this message
Jan 28 04:48:53 eoc-node3 systemd[1]: Started The Apache HTTP Server.
Jan 28 04:48:53 eoc-node3 httpd[19167]: Server configured, listening on: port 80
```

4.2 Let's create a task which create a file called **/tmp/foo** on any host of choice by using touch command in the file **task3.yml.**

```
- hosts: eoc-node1
  become: yes
```

```
  tasks:
    - name: touch a file
      file:
        path: /tmp/foo
        state: touch
```

4.3 Let's view the manifest of **task3.yml**.

```
# cat -n task3.yml
```

Output:

```
[admin@eoc-controller ~]$ cat -n task3.yml
     1  - hosts: eoc-node1
     2    become: yes
     3    tasks:
     4      - name: touch a file
     5        file:
     6          path: /tmp/foo
     7          state: touch
```

4.4 Let's perform a syntax check on the playbook.

```
# ansible-playbook --syntax-check task3.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook --syntax-check task3.yml

playbook: task3.yml
```

4.5 Let's run the task which touch some files.

```
# ansible-playbook task3.yml
```

Output:

```
[admin@eoc-controller ~]$ ansible-playbook task3.yml

PLAY [eoc-node1] ***************************************************************************

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

TASK [touch a file] ************************************************************************
changed: [eoc-node1]

PLAY RECAP *********************************************************************************
eoc-node1                  : ok=2    changed=1    unreachable=0    failed=0    skipped=0    resc
ued=0    ignored=0
```