

Ansible Roles



Task 15

- ☀ Create an Ansible role myapache to configure Httpd WebServer.
- ☀ Create another ansible role myloadbalancer to configure HAProxy LB.
- ☀ We need to combine both of these roles controlling web server versions and solving challenges for host IP's addition dynamically over each Managed Node in HAProxy.cfg file.

What is Roles in ansible?

Roles let you automatically load related vars_files, tasks, handlers, and other **Ansible** artifacts based on a known file structure. Once you group your content in **roles**, you can easily reuse them and share them with other users.

Role directory structure. Storing and finding **roles**.

How Ansible playbook role Benefit us ?

They provide a **skeleton for an independent and reusable collection of variables, tasks, templates, files, and modules** which can be automatically loaded into the **playbook**. **Playbooks** are a collection of **roles**. ... Instead, you can create 10 different **roles**, where each **role** will perform one task.

So, from here my Task starts, where at first I will create my first role of **myapache** to configure Httpd WebServer named task15.1

So for this first, I will need to create a directory

```
mkdir /wst15
```

```
cd /wst15
```

```
Ansible-galaxy role init task15.1
```

```
cd task15.1/
```

```
[root@localhost wst15]# cd task15.1/
[root@localhost task15.1]# ls
defaults  files  handlers  meta  README.md  tasks  templates  tests  vars
[root@localhost task15.1]#
```

A role named task15.1 which actually will work as a webserver is created and ready with 9 modules present with it.

Now go inside the tasks

```
cd /tasks
```

Here already a **main.yml** file will be present so start editing on that one which will work as a tasks section of ansible yml code.

It will look like this,

```
vim main.yml
```

```
---
# tasks file for task15.1
#
- package:
    name: "{{p}}"
    state: present

- copy:
    dest: "/var/www/html/path.html"
    content: "FInally,plaese do  Httpd Configured with help of roles via html"

- service:
    name: "{{p}}"
    state: started
~
```

So, here I used variables which must be needed to set into variables module

```
cd ..
```

```
cd vars/
```

```
vim main.yml
```

```
---  
# vars file for task15.1  
#  
p: "httpd"  
~
```

Here, also main.yml file will be by default created

Now our First role is ready

Coming to the Second Role.

Again the same steps

```
cd /wst15
```

```
Ansible-galaxy role init haproxy
```

```
[root@localhost wst15]# ansible-galaxy init haproxy  
- Role haproxy was created successfully
```

```
cd haproxy/
```

It will also have 9 modules created, so we will move to the tasks, but before that, in this haproxy task, we needed a haproxy.cfg file, which we

can copy by cp command, and one can get that one while installing haproxy manually. Set the destination of that file to tasks, now

Tasks have two files

```
[root@localhost tasks]# ls
haproxy.cfg  main.yml
```

Vim haproxy.cfg

```
#-----
backend app
    balance      roundrobin
    server  app1 192.168.43.35:80 check
-- INSERT --
```

At the end of the file, it needed the backend IP's or the Webserver attached to the load balancer. now,

vim main.yml

```
---
# tasks file for haproxy
- package:
    name: "haproxy"
    state: present
- template:
    src: "haproxy.cfg"
    dest: "/etc/haproxy"
- service:
    name: "haproxy"
    state: restarted
```

After this our Second task is also over, approx all the main problems are solved now.

We are only left with the ansible setup and a file that will combine both the tasks.

So, let's check about the Ansible Repository and its conf.

```
vim /root/ip.txt
```

```
[front]
192.168.43.35 ansible_user=root ansible_ssh_pass=Abhishek ansible_connection=ssh

[back]
192.168.43.194 ansible_user=root ansible_ssh_pass=Abhishek ansible_connection=ssh
~
```

Here, front is my Webserver- where a webpage is configured with IP 35.

And back is my load balancer- with IP 194.

Now:

```
vim /etc/ansible/ansible.cfg
```

```
[defaults]
inventory=/root/ip.txt
host_key_checking=false
roles_path=/wst15
~
```

Coming back to the directory, Create a yml file

```
cd ..
```

```
vim preq.yml
```

```
- hosts: front
  roles:
    - role: "task15.1"
- hosts: back
  roles:
    - role: "haproxy"
```

One is for Webserver

I provided **hosts as my webserver ip group** present in my inventory and the **Role I created for a webserver**.

The second is for Haproxy.

I provided **hosts as my haproxy ip group** present in my inventory and the **Role I created for haproxy**.

Now, we have three files in the directory, and now ready to run the **Ansible-yml** file.

```
[root@localhost wst15]# ls
haproxy  preq.yml  task15.1
```

ansible-playbook preq.yml

```
[root@localhost wst15]# ansible-playbook preq.yml

PLAY [front] *****
TASK [Gathering Facts] *****
ok: [192.168.43.35]

TASK [task15.1 : package] *****
ok: [192.168.43.35]

TASK [task15.1 : copy] *****
changed: [192.168.43.35]

TASK [task15.1 : service] *****
ok: [192.168.43.35]
```

```
PLAY [back] *****
TASK [Gathering Facts] *****
ok: [192.168.43.194]

TASK [haproxy : package] *****
ok: [192.168.43.194]

TASK [haproxy : template] *****
ok: [192.168.43.194]

TASK [haproxy : service] *****
changed: [192.168.43.194]

PLAY RECAP *****
192.168.43.194      : ok=4    changed=1    unreachable=0    failed=0
0      rescued=0    ignored=0
192.168.43.35      : ok=4    changed=1    unreachable=0    failed=0
0      rescued=0    ignored=0
```


Finally, We can see my **Website is launched** with a **Load Balancer** at the **frontend** and **server at the backend**.

Checking which **ports are in listening** from the **Web Server terminal**,
i.e **Backend Server**

Netstat -tnlp

```
[root@localhost ~]# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      789/sshd
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN      1933/haproxy
tcp6       0      0 :::22                  :::*                    LISTEN      789/sshd
[root@localhost ~]#
```

We can check our **Web server is launched**.

<http://192.168.43.194:5000/path.html>



Also as we know here the Ip of my webserver, so directly i can check by,

<http://192.168.43.35/path.html>

Thank you