



ANSIBLE

ANSIBLE BASIC LAB MANUAL

Student Lab Kit v1.1

ABSTRACT

This lab manual is designed for students who are interested in Ansible Basic Automation

Confidential Document

Ansible Roles

Table of Contents

Lab Overview and objectives	2
<i>Guided Tasks</i>	2
Task 1: Role directory structure	2
Task 2: Create directory structure	2
Task 3: Create tasks – part 1	3
Task 4: Create handlers	3
Task 5: Create tasks – part 2	3
Task 6: Create handlers	4
Task 7: Create templates	4
Task 8: Invoke role in a playbook	4
Task 9: Test role	5

Lab Overview and objectives

The purpose of this lab is to understand Ansible role usage and also to practice writing a role for managing our lab environment hosts.

Ansible roles are a feature that facilitates reuse of some files or playbooks, by grouping multiple tasks together into one container in order to do the automation in an effective manner with clean directory structures.

Guided Tasks

Task 1: Role directory structure

There is a predefined structure for an Ansible role, starting from its root directory which is also considered the role name. Inside, it has some directories which contain “.yaml” files. The following directories may be found inside a role:

- tasks - contains the main list of tasks to be executed by the role;
- handlers - contains handlers, which may be used by this role or even anywhere outside this role;
- defaults - default variables for the role;
- vars - other variables for the role;
- files - contains files which can be deployed via this role;
- templates - contains templates which can be deployed via this role;
- meta - defines some meta data for this role. See below for more details.

Ansible provides an option to start creating your own role using `ansible-galaxy init` command, which will create the entire hierarchy of directories and “.yaml” files inside your role directory.

In practice, you don't use all of those directories and it is recommended to create your directory structure by hand, with only the necessary files. Notice that a role should contain at least one directory from the previous list!

Task 2: Create directory structure

We have right now in our environment 2 webserver (ubuntu and centos) with apache installed and custom index pages. Let's suppose that we want to set our `hivemaster` host as a reverse proxy for our `webserver`s, which should balance between the requests between the 2 backend servers, so let's create a role for installing NGINX and configuring load-balancing. Let's start with creating necessary directories:

```
student@ansible-00-01-hivemaster:~/test$ mkdir -p nginx_role/defaults
student@ansible-00-01-hivemaster:~/test$ mkdir -p nginx_role/handlers
student@ansible-00-01-hivemaster:~/test$ mkdir -p nginx_role/tasks
student@ansible-00-01-hivemaster:~/test$ mkdir -p nginx_role/templates
```

Task 3: Create tasks – part 1

Now, that we have the directories let's move on to the .yaml files. Let's create tasks/main.yaml file:

```
student@ansible-00-01-hivemaster:~$ vi nginx_role/tasks/main.yaml
---
- name: install nginx
  package:
    name: nginx
    state: latest
  when: ansible_os_family == 'Debian'
  notify: restart nginx

- name: Configure NGINX
  import_tasks: configure.yaml
```

The first task is to install `nginx` (we customized the installation just for “Debian” distribution).

Task 4: Create handlers

As you can see, we also called a handler (“`restart nginx`”), so let's write it in the appropriate location:

```
student@ansible-00-01-hivemaster:~$ vi nginx_role/handlers/main.yaml
---
- name: restart nginx
  service:
    name: nginx
    state: restarted
    enabled: yes

- name: reload nginx
  service:
    name: nginx
    state: reloaded
```

We added a “reload nginx” handler, which performs a reload of configuration without stopping the NGINX service (reload != restart).

Task 5: Create tasks – part 2

We are going to continue with adding tasks necessary for NGINX configuration. Because we want to implement also the concept of reusable playbooks, we imported the configuration file (`configure.yaml`) into `main.yaml` file:

```
student@ansible-00-01-hivemaster:~$ vi nginx_role/tasks/configure.yaml
---
```

```
- name: copy config file
  template:
    src: load_balancer.conf.j2
    dest: "{{ nginx_conf_path }}/load_balancer.conf"
  notify: restart nginx

- name: Remove default home page served by nginx
  file:
    path: "{{ nginx_path }}/sites-enabled/default"
    state: absent

- name: test nginx configuration
  command: nginx -t
  changed_when: false
  notify: reload nginx
```

Task 6: Create handlers

We used the variables `nginx_conf_path` and `nginx_path` in this playbook which by default points to `/etc/nginx/conf.d` and `/etc/nginx`. We will define these variables in `defaults/main.yml`:

```
student@ansible-00-01-hivemaster:~$ vi nginx_role/defaults/main.yml
---
nginx_conf_path: /etc/nginx/conf.d
nginx_path: /etc/nginx
```

Task 7: Create templates

We also have to create the template for our load balancer:

```
student@ansible-00-01-hivemaster:~$ vi
nginx_role/templates/load_balancer.conf.j2
upstream backend {
    server {{ backend_srv_01 }};
    server {{ backend_srv_02 }};
}
server {
    listen {{ listen_port }};
    location / {
        proxy_pass http://backend;
    }
}
```

Task 8: Invoke role in a playbook

Now let's create a playbook (in `/home/student`) to test the usage of the role:

```
student@ansible-00-01-hivemaster:~$ cd
student@ansible-00-01-hivemaster:~$ vi roletest.yml
---
- name: Test NGINX role
  hosts: hivemaster
  become: true
  vars:
    backend_srv_01: 10.128.0.49
    backend_srv_02: 10.142.15.213
    listen_port: 8081

  tasks:
  pre_tasks:
    - debug:
        msg: 'Starting NGINX installation process.'

  roles:
    - nginx_role

  post_tasks:
    - debug:
        msg: 'Web server has been configured.'
```

Notice that we used `pre_tasks` and `post_tasks` for this role: these ensures that the tasks listed in `pre_tasks` section are executed before running the role and the tasks listed in `post_tasks` after the execution. We also defined the variables necessary for the role: IP addresses for our backend servers and also the listening IP address for NGINX server (`hivemaster`).

*make sure to replace the IPs with your hosts ones

Task 9: Test role

Run the playbook and explore playbook recap:

```
student@ansible-00-01-hivemaster:~$ ansible-playbook roletest.yml

PLAY [Test NGINX role]
*****

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

TASK [debug]
*****
ok: [hivemaster] => {
  "msg": "Starting NGINX installation process."
}

TASK [nginx_role : install nginx]
*****
```

```

changed: [hivemaster]

TASK [nginx_role : copy config file]
*****
changed: [hivemaster]

TASK [nginx_role : Remove default home page served by nginx]
*****
changed: [hivemaster]

TASK [nginx_role : test nginx configuration]
*****
ok: [hivemaster]

RUNNING HANDLER [nginx_role : restart nginx]
*****
changed: [hivemaster]

TASK [debug]
*****
ok: [hivemaster] => {
    "msg": "Web server has been configured."
}

PLAY RECAP
*****
hivemaster      : ok=8    changed=4    unreachable=0    failed=0
skipped=0      rescued=0    ignored=0

```

Now let's test that our NGINX server is listening on specified port (8081). Running the curl command twice you should get 2 different responses (1 from each backend server) because the balancing algorithm is default (Round Robin):

```

student@ansible-00-01-hivemaster:~$ curl localhost:8081
<html>
<head>
<title>Welcome to Ansible</title>
</head>
<body>
<p>
Server details:
Hostname: ansible-00-02-ubuntu
OS: Ubuntu 18.04
</p>
</body>
</html>

student@ansible-00-01-hivemaster:~$ curl localhost:8081
<html>
<head>
<title>Welcome to Ansible</title>
</head>
<body>
<p>
Server details:

```

```
Hostname: ansible-00-03-centos
OS: CentOS 7.7
</p>
</body>
</html>
```

If your curl command doesn't work, please check that your NGINX is listening on port 8081 using `netstat` command:

```
student@ansible-00-01-hivemaster:~$ sudo netstat -napt | grep LISTEN
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
25491/mysqld
tcp        0      0 0.0.0.0:8081            0.0.0.0:*               LISTEN
16129/nginx: master
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
964/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
5240/sshd
```