Ansible Workshop - Exercises

Basics

Get to know Ansible and learn to write your first Ansible Playbooks.

7 - Trigger changes with Handlers

Objective

Get to know *Handlers*, a special task which is defined in its own *play* parameter. A handler is often used to restart services, but it can be used with every module Ansible offers.

Guide

Sometimes when a task does make a change to the system, an additional task or tasks may need to be run. For example, a change to a service's configuration file may then require that the service be restarted so that the changed configuration takes effect.

This is where Ansible's *handlers* come into play. Handlers can be seen as inactive tasks that only get triggered when explicitly invoked using the "notify" statement. Read more about them in the Ansible Handlers documentation.

Step 1 - Handlers

As a an example, let's write a playbook that:

- manages Apache's configuration file `/etc/httpd/conf/httpd.conf` on all hosts in the `web` group
- restarts Apache when the file has changed

First we need the file Ansible will deploy, let's just take the one from node1.

```
scp node1:/etc/httpd/conf/httpd.conf ~/ansible_files/files/.
```

We now have the configuration file for our webserver, we will adjust the file later and copy it back to all webserver hosts later.
Next, create the Playbook `httpd_conf.yml`. Make sure that you are in the directory `~/ansible_files`.

```
1  ---
2  - name: Manage httpd.conf
3    hosts: web
4    become: true
5    handlers:
6      - name: Restart Apache service
7        listen: restart_apache_handler
8        ansible.builtin.service:
9          name: httpd
10         state: restarted
11   tasks:
12     - name: Copy Apache configuration file
13       ansible.builtin.copy:
14         src: httpd.conf
15         dest: /etc/httpd/conf/httpd.conf
16         mode: "0644"
17         owner: apache
18         group: apache
19       notify:
20         - restart_apache_handler
```

So what's new here?

- The `handlers` key (line 5) defines a task that is only run on notification. You can define multiple tasks (handlers) here.
- The `listen` key (line 7) defines the keyword which triggers the handler. You could also use the name of the handler task.
- The `notify` parameter calls the handler (by referencing the *listen* keyword) only when the copy task actually changes the file. That way the service is only restarted if needed - and not each time the playbook is run.

Run the playbook. We didn't change anything in the file yet so there should not be any `changed` lines in the output and of course the handler shouldn't have fired.

- Now change the `Listen 80` line in `~/ansible_files/files/httpd.conf` to:

```
Listen 8080
```

- Run the playbook again. Now the Ansible's output should be a lot more interesting:
  - httpd.conf should have been copied over
  - The handler should have restarted Apache

> **Note**
>
> By default, handlers run after all the tasks in a particular play have been completed.

Apache should now listen on port 8080. Easy enough to verify:

```
[student@ansible-1 ansible_files]$ curl http://node1
curl: (7) Failed to connect to node1 port 80: Connection refused
[student@ansible-1 ansible_files]$ curl http://node1:8080
<body>
<h1>This is a development webserver, have fun!</h1>
</body>
```

> **Warning**
>
> If you are using the local development environment, remember, you are using containers instead of actual VMs! You need to **append the correct port** (e.g. `curl http://node1:8002"` for Port 80, `curl http://node1:8003"` for Port 8080).
> Take a look at the table with the ports overview or execute `podman ps` and check the output.

Run the playbook one last time. As the configuration file is already copied over with the desired configuration state, the handler is not triggered, Apache will keep running.