Handlers

In Ansible, handlers are a way to trigger actions based on the results of tasks. Handlers are similar to tasks, but they are only executed if one or more tasks notify them. This allows you to perform certain actions, such as restarting a service, only when changes have been made that affect the state of the system.

Here is a basic example to illustrate how handlers work:

```
---
- name: Example playbook with handlers
  hosts: your_servers
  tasks:
    - name: Ensure the Nginx package is installed
      apt:
        name: nginx
        state: present
      notify:
        - Restart Nginx

  handlers:
    - name: Restart Nginx
      service:
        name: nginx
        state: restarted
```

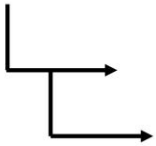
In this example:

1. The task ensures that the Nginx package is installed.
2. If the installation task makes any changes (installs or updates Nginx), it will notify the handler called "Restart Nginx."
3. The handler, defined under the **handlers** section, specifies that it should restart the Nginx service if it is notified.

Handlers are only executed when they are notified by a task, and the notification occurs when the task makes changes to the system. If no tasks notify a particular handler, it won't be executed. This helps prevent unnecessary restarts or other actions when there are no changes to be applied.

You can notify multiple handlers from a single task, and handlers will be executed in the order they are defined in the playbook. Handlers are defined at the same indentation level as tasks, typically after the list of tasks in a playbook.

Handlers provide a way to consolidate and organize the management of services or other actions that need to be taken in response to changes in your infrastructure.



- Handlers are executed at the end of the play once all tasks are finished. In Ansible, handlers are typically used to start, reload, restart, and stop services
- Sometimes you want to run a task only when a change is made on a machine. For example, you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged.
- Remember the case when we had to reload the firewalld because we wanted to enable http service? Yes, that is a perfect example of using handlers
- So basically handlers are tasks that only run when **notified**
- Each handler should have a globally unique name

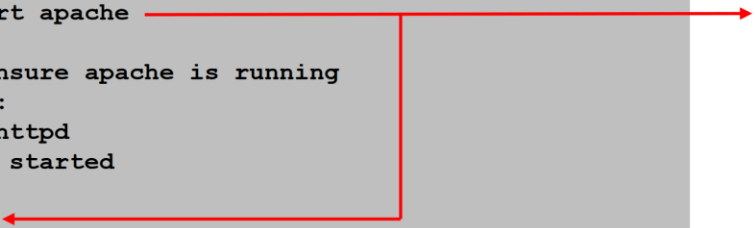
Example

```
---
- name: Verify apache installation
  hosts: localhost
  tasks:
  - name: Ensure apache is at the latest version
    yum:
     name: httpd
     state: latest

  - name: Copy updated apache config file
    copy:
     src: /tmp/httpd.conf
     dest: /etc/httpd.conf
    notify:
    - Restart apache

  - name: Ensure apache is running
    service:
     name: httpd
     state: started

  handlers:
    - name: Restart apache
      service:
       name: httpd
       state: restarted
```



Firewalld example

```
---
- name: Enable service on firewalld
  hosts: localhost
  tasks:

  - name: Open port for http
    firewalld:
     service: http
     permanent: true
     state: enabled
    notify:
    - Reload firewalld

  - name: Ensure firewalld is running
      service:
       name: firewalld
       state: started

  handlers:
    - name: Reload firewalld
      service:
       name: firewalld
       state: reloaded
```