HANDLING TASKS

- In Ansible, handlers are typically used to start, reload, restart, and stop services.
- Sometimes you want a task to run only when a change is made on a machine.
  E.g.: you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged. Ansible uses handlers to address this use case.
- Handlers are tasks that only run **when notified.**
- By default, handlers are executed **last regardless** of their location in the playbook.

A single task and a handler:

```
---
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Install the latest version of Apache
      dnf:
        name: httpd
        state: latest

      notify:
       - Start Apache

  handlers:
    - name: Start Apache
```

```
    service:
      name: httpd
      state: started
...
```

Multiple tasks and handlers:

```
---
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Install the latest version of Apache
      yum:
        name: httpd
        state: latest
    - name: Configure Apache
      copy:
        src: /home/raju/index.html
        dest: /var/www/html
        owner: apache
        group: apache
        mode: 0644
      notify:
      - Configure Firewall
      - Start Apache
```

```
handlers:
  - name: Start Apache
    service:
      name: httpd
      state: started

  - name: Configure Firewall
    firewalld:
      permanent: yes
      immediate: yes
      service: http
      state: enabled
```

Handling Task Failure:

Ansible evaluates the return code of each task to determine whether the task succeeded or faild.

Normally, When a task fails Ansible immediately aborts the test of the play on that host, skipping all subsequent tasks.

Ignoring Task Failure:

By default a task fails, the play is aborted. However, this behavior can be overridden by ignoring faild tasks.

You can use the ignore_error keyword in a task to accomplish this.

Example:

```
---
- hosts: server
  become: true
  become_user: root
  tasks:
    - name: Restart a service
      service:
        name: not a service
        state: restart
    - name: Copy a script
      copy:
        src: /tmp/script.sh
        dest: /opt
...
```

```
$ansible-playbook --syntax-check task1.yml
$ansible-playbook task1.yml -K
```

```
---
```

ignore_errors:

```
$vi task2.sh
- hosts: webservers
  become: true
  become_user: root
```

```
  tasks:
    - name: Restart a service
      service:
        name: not a service
        state: restart
      ignore_errors: yes
    - name: Copy a script
      copy:
        src: /tmp/script.sh
        dest: /opt
...
```

$ansible-playbook --syntax-check task2.yml

$ansible-playbook task1.yml -K

Register:

Ansible register is a way to capture the output from task execution and store it in a variable.

Faild_when:

$vi task3.yml

```
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Restart a service
```

```
    service:
      name: not a service
      state: restart
    ignore_errors: yes
  - name: Copy a script
    copy:
      src: /tmp/script.sh
      dest: /tmp
  - name: Run the script
    shell: sh /tmp/script.sh
    register: command_result
  - debug: msg="{{ command_result.stdout }}"
```

$vi task4.sh

```
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Restart a service
      service:
        name: not a service
        state: restart
      ignore_errors: yes
    - name: Copy a script
```

```
    copy:
      src: /tmp/script.sh
      dest: /tmp
  - name: Run the script
    shell: sh /tmp/script.sh
    register: command_result
    failed_when: "'raju' in command_result.stdout"
  - debug: msg="{{ command_result.stdout }}"
  - name: Restart the HTTPD
    service:
      name: httpd
      state: restarted
```

changed_when:

The changed_when keyword can be used to control when a task reports that it has changed.

```
---
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Restart a service
      service:
        name: not a service
        state: started
```

```
    ignore_errors: yes
  - name: Copy a script
    copy:
      src: /tmp/script.sh
      dest: /tmp
  - name: Run the script
    shell: sh /tmp/script.sh
    register: command_result
    changed_when: "'success' in command_result.stdout"
    notify:
     - restart_apache
 handlers:
   - name: restart_apache
     service:
       name: httpd
       state: restarted
...
```