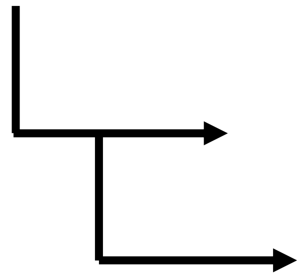
Complete Ansible Automation Training

Additional Features in Ansible

Handlers



- 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

Handlers

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
```

httpd service will be restart at the end

By: Imran Afzal
www.utclisolutions.com

Firewalld Example

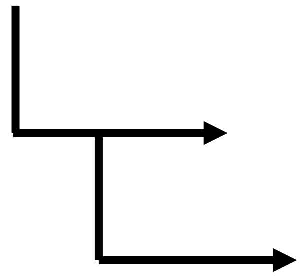
Handlers

```
---
- 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
```



By: Imran Afzal
www.utclisolutions.com

Conditions

- Condition execution allow Ansible to take actions on its own based on certain conditions
- Under condition certain values must be met before executing a tasks
- We can use the WHEN statement to make Ansible automation more smart

Example:

```
- name: Playbook description
  hosts: localhost

  tasks:
  - name: Start a service
    when: A == "B"
    service:
     name: servicename
     state: started
```

By: Imran Afzal
www.utclisolutions.com

Conditions

```
# vim httpbycondition.yml

---
- name: Install Apache WebServer
  hosts: localhost

  tasks:
    - name: Install Apache on Ubuntu Server
      apt-get:
        name: apache2
        state: present
      when: ansible_os_family == “Ubuntu“

    - name: Install Apache on CentOS  Server
      yum:
        name: httpd
        state: present
      when: ansible_os_family == "RedHat"
```

Ansible built-in variable

How to get a list of all
Ansible built-in variables

Variables are gathered from facts

Gather list of facts of a host

```
# ansible localhost -m setup
```

By: Imran Afzal
www.utclisolutions.com

Loops

- A loop is a powerful programming tool that enables you to execute a set of commands repeatedly

- We can automate specific task but what if that task itself repetitive?
  - e.g. Changing permissions on hundreds of files
  - Creating multiple users at once
  - Installing many packages on hundreds of servers

- Loops can work hand in hand with conditions as we loop certain task until that condition is met

- When creating loops, Ansible provides these two directives: **`loop`** and **`with_*`** keyword.

Users Example

Loops

- To create multiple users in Linux command line we use “for loop”

```
e.g.
# for u in jerry kramer eliane; do useradd $u; done
```

```
vim userloop.yml

---
- name: Create users
  hosts: localhost

  tasks:
1  - name: Create jerry
     user:
       name: jerry

2  - name: Create kramer
     user:
       name: kramer

3  - name: Create eliane
     user:
       name: eliane
```

1 Adding loop parameter

```
vim userbyloop1.yml

---
- name: Create users thru loop
  hosts: localhost

  tasks:
  - name: Create users
    user:
      name: ”{{ item }}””
    loop:
      - jerry
      - kramer
      - eliane
```

2 Adding variable

```
vim userbyloop2.yml

---
- name: Create users thru loop
  hosts: localhost
  vars:
   users: [jerry,kramer,eliane]

  tasks:
  - name: Create users
    user:
      name: ‘{{item}}‘
    with_items: ‘{{users}}’
```

By: Imran Afzal
www.utclisolutions.com

Packages Example

Loops

- To install multiple packages in Linux command line we use “for loop”

```
e.g.
# for p in ftp telnet htop; do yum install $p -y; done
```

❶ Adding variable and calling variables through item parameter

```
vim installbyloop1.yml

---
- name: Install packages thru loop
  hosts: localhost
  vars:
   packages: [ftp,telnet,htop]

  tasks:
  - name: Install package
    yum
      name: '{{items}}'
      state: present
    with_items: '{{packages}}'
```

❷ Adding variable and calling variables directly

```
vim installbyloop2.yml

---
- name: Install packages thru loop
  hosts: localhost
  vars:
   packages: [ftp,telnet,htop]

  tasks:
  - name: Install packages
    yum
      name: '{{packages}}'
      state: present
```

By: Imran Afzal
www.utclisolutions.com