ANSIBLE

YAML

- The **YAML** stands for **Yet Another Markup Language** which includes human readable content and often used in configuration files

Following are the features of YAML:

- Compare to XML or JSON, YAML is less complex and provides same features.
- It provides configuration settings without need to learn complex code types such as CSS, JavaScript or PHP.

Basic Rules of YAML

- You must end the YAML files with **.yaml** (or) **.yml** extension.
- YAML must be case sensitive.
- YAML doesn't support use of tabs. Instead of tabs, it uses spaces which are supported universally.

Basic Data Types of YAML

- YAML supports some basic data types which can be used with programming languages such as:
- **Scalars**: strings or numbers.
- **Sequences**: arrays or lists.
- **Mappings**: hashes or dictionaries.

Example Yaml File

```
---
- Martin:
      name: Martin smith
      job: Developer
      skills:
          - python
          - perl
- John:
     name: John pal
     job:   Developer
     skills:
          - lisp
          - fortran
```

Playbook Basics

Hosts and Users : each play in a playbook, you get to choose which machines in your infrastructure to target and what remote user to complete the steps (called tasks)

Ex:

```
---
- hosts: webservers
  remote_user: root
```

Playbook Basics

Tasks list: Each play contains a list of tasks. Tasks are executed in order, one at a time, against all machines matched by the host pattern. the playbook will runs in top to bottom

Ex:

```
tasks:
    - name: make sure apache is running
      service: name=httpd state=started
```

```
#vi  playbooks/pb1.yml
-   hosts: all
    remote_user: root
  tasks:
    - name: copy files
      copy:
        src: /root/sample.txt
        dest: /tmp/sample.txt
    - name: msg display
      debug:
        msg: Files are Copied!
#ansible-playbook playbooks/pb1.yml
```

#vi playbooks/pb2.yml

```
- hosts: ubnt
  remote_user: root
  gather_facts : False
  tasks:
   - name: install apache
     apt: name=apache2 state=latest
   - name: install tomcat
     apt: name=tomcat7 state=latest
- hosts: cent
  gather_facts : False
  tasks:
   - name: install httpd
     yum:  name=httpd  state=latest
```

#ansible-playbook playbooks/pb2.yml

Handlers: Running Operations On Change

```
- hosts: ubnt
  tasks:
   - name: install apache
     apt: name=apache2 state=latest
     notify:
       - start apache
   - name: install tomcat
     apt: name=tomcat7 state=latest
     notify:
       - start tomcat
  handlers:
   - name: start apache
     service: name=apache2 state=started
   - name: start tomcat
     service: name=tomcat7 state=started
```

When Condition

```
# ansible all -m setup -a "filter=ansible_distribution“
# ansible all -m setup -a "filter=ansible_pkg_mgr“

- hosts: all
  remote_user: root
  tasks:
    - name: install apache
      apt: name=apache2 state=latest
      when: ansible_distribution == "Ubuntu"

    - name: install httpd
      yum: name=httpd state=latest
      when: ansible_distribution == "CentOS"
```

When Condition

```
- hosts: all
  remote_user: root
  tasks:
   - name: to update packages
     raw: apt-get update
     when: ansible_distribution=='Ubuntu'

   - name: to update packages
     raw: yum -y update
     when: ansible_distribution=='CentOS‘
 - name: to install apache2
     apt: name=apache2 state=latest
     when: ansible_distribution=='Ubuntu'
     notify:
        - start apache
```

When Condition

```
- name: to install httpd
    yum: name=httpd state=latest
    when: ansible_distribution=='CentOS'
    notify:
       - start httpd

  - name: to deploy file
    copy:
      src: /root/index.html
      dest: /var/www/html/index.html

  handlers:
   - name: start apache
    service: name=apache2 state=started

  - name:  start httpd
    service: name=httpd state=started
```

playbook

```
- hosts: all
  become: true
  tasks:
  - name: to update package
    raw: apt-get update
    when: ansible_distribution == 'Ubuntu'

  - name: to install mysql
    apt: name=mysql-server state=latest
    when: ansible_distribution == 'Ubuntu'
    notify:
       - start mysql

 - name: to update package
    raw: yum -y update
    when: ansible_distribution == 'CentOS'
```

playbook

```
- name: to install mariadb
  yum: name=mariadb-server state=latest
  when: ansible_distribution == 'CentOS'
  notify:
     - start mariadb

 handlers:
 - name: start mysql
   service: name=mysql state=started

 - name: start mariadb
   service: name=mariadb state=running
```

Ansible Variables

a Variable is an Element which can hold a specific value.

Variable names should be letters, numbers, and underscores.

Variables should always start with a letter.

Ex:

```
- hosts: all
  remote_user: root
  vars:
        pack1: net-tools
        pack2: wget
    tasks:
     - name: Install Package {{pack1}}
       yum: name="{{pack1}}" state=latest
       when: ansible_distribution=="CentOS“

     - name: Install Package {{pack2}}
       apt: name="{{pack2}}" state=latest
       when: ansible_distribution=="Ubuntu"
```

vars_prompt

It will take user input while executing playbook with vars_prompt and store into a variable.

```
- hosts: centos
  remote_user : root
  vars_prompt:
    - name: pack_name
      prompt: Give Package Name
      private: no
  tasks:
    - name: Install Package {{pack_name}}
      yum:
        name: "{{pack_name}}"
        state: latest
```

```
- hosts: all
  remote_user: root
  vars_prompt:
     - name: pack_name1
       prompt: Enter Package-1
       private: yes

     - name: pack_name2
       prompt: Enter Package-2
       private: no
  tasks:
     - name: install {{pack_name1}}
       apt: name="{{pack_name1}}" state=latest
       when: ansible_pkg_mgr=="apt"

     - name: install {{pack_name2}}
       yum: name="{{pack_name2}}" state=latest
       when: ansible_pkg_mgr=="yum”
```

Ansible loops

- Ansible loops can do many things in one task, such as create a lot of users, copy a set of files ,install a lot of packages.

Ex:

```
- hosts: all
  remote_user: root
  tasks:
    - name: copy {{item}}
      copy:
        src: /root/files/{{item}}
        dest: /tmp/{{item}}
      with_items:
        - abc1.txt
        - abc2.txt
        - abc3.txt
```

To add multiple users using Ansible Loops:

```
- hosts: all
  remote_user: root
  tasks:
    - name: add group
      group: name="sales" state=present

    - name: Add user {{item}}
      user: name={{item}} groups="sales“ state=“present”
      with_items:
        - user1
        - user2
        - user3
```