WORKING WITH PLAYBOOKS

- Playbooks are the files where ansible code is written. playbooks are written in YAML format.
- Playbooks are one of the core features of ansible and tell ansible what to execute.
- Ansible uses YAML syntax for expressing ansible playbooks because it is very easy for humans to understand, read and write a than other common data formats like XML or JSON.
- Each playbook is an aggregation of one or more plays in it. Playbooks are structured using plays.
- There can be more than one play inside a playbook.

YAML SYNTAX:

- It provides a basic overview of correct YAML syntax, which is how Ansible playbooks (our configuration management language) are expressed.
- We use YAML because it is easier for humans to read and write than other common data formats like XML or JSON. Further, there are libraries available in most programming languages for working with YAML.

YAML BASICS

- For Ansible, nearly every YAML file starts with a list. Each item in the list is a list of key/value pairs, commonly called a “hash” or a “dictionary”. So, we need to know how to write lists and dictionaries in YAML.
- There’s another small quirk to YAML. All YAML files (regardless of their association with Ansible or not) can optionally begin with --- and end with .... This is part of the YAML format and indicates the start and end of a document.
- All members of a list are lines beginning at the same indentation level starting with a "- " (a dash and a space):

```
---
# A list of tasty fruits
- Apple
- Orange
- Strawberry
- Mango
...
```

A dictionary is represented in a simple key: value form (the colon must be followed by a space):

```
# An employee record
jai:
  name: jai
  job: Developer
  skill: cloud
```

More complicated data structures are possible, such as lists of dictionaries, dictionaries whose values are lists or a mix of both:

```
# Employee records
- jai:
    name: jai developer
    job: Developer
    skills:
      - python
      - perl
      - pascal
```

```
- ram:
    name: ram admin
    job: administrator
    skills:
      - linux
      - aws
      - azure
```

EXAMPLES

Using AD-HOC command:

$ansible -m user -a "name=jai uid=1010 state=present" webservers --become -K

##################################

$cd /etc/ansible

$vi simple.yml

```
---
- hosts: webservers
  become: true
  become_user: root
  tasks:
      - name: User Account Creation
        user:
          name: jai
          uid: 1020
          state: present
...
```

Executing Playbook:

$ansible-playbook sample.yml -K

$id jai [from agent1]

NOTE: RUN Second time

It will ignore it

Modification:

```
- hosts: webservers
  become: true
  become_user: root
  tasks:
      - name: User Account Creation
        user:
          name: jai
          uid: 1021
          state: present
...
```

$ansible-playbook sample.yml -K

$id jai

DRY-RUN:

When ansible-playbook is executed with --check it will not make any changes on remote systems.

Instead, any module instrumented to support 'check mode' will report what changes they would have made rather than making them.

$ansible-playbook -c sample.yml -K

CHECK-SYNTAX ERROR:

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

INSTALLING APACHE:

$vi apache.yml

```
---
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Installation of HTTPD
      yum:
        name: httpd
        state: present
    - name: Creation of index.html page
      copy:
        content: "WELCOME TO DEVOPS"
        dest: /var/www/html/index.html
    - name: Starting HTTPD Service
      service:
        name: httpd
        state: started
        enabled: true
...
```

$ansible-playbook --syntax-check apache.yaml --step

INSTALLATION OF HTTPD & FIREWALLD (MULTI PLAYS):

$vi multipack.yml

```
---
- hosts: webservers
  become: true
  become_user: root
  tasks:
    - name: Installation of HTTPD & Firewalld Packages
      yum:
        name:
         - httpd
         - firewalld
    - name: Creating index.html
      copy:
        content: "WELCOME TO DEVOPS\n"
        dest: /var/www/html/index.html
    - name: Firewalld Start & Enabled
      service:
        name: firewalld
        state: started
        enabled: true
    - name: Firewalld permits access to httpd service
      firewalld:
        service: http
```

```
      permanent: true
      state: enabled
      immediate: yes
   - name: Httpd Start & Enabled
     service:
      name: httpd
      state: started
      enabled: true
- name: Test Intranet Web Server Connctivity
  hosts: webservers
  become: no
  tasks:
   - name: connect to Intranet Web Server
     uri:
      url: http://Node1
      return_content: yes
      status_code: 200
...
$ansible-playbook --syntax-check multi-pack.yml
$ansible-playbook multi-pack.yml -K
$curl  http://Node1
#firewall-cmd --list-all
#firewall-cmd --remove-service=http
#fiewall-cmd --list-all
```