```
==========================
Configuration Management
==========================

=> Installing required softwares in the machines

=> Copy required files from one machine to another machine

=> OS Patching/Updates

=> We can perform configuration management in 2 ways

                1) Manual Configuration Management

                2) Automated Configuration Management

============================================
Problems with Manual Configuration Mgmt
============================================

1) Repeating same work

2) Time Consuming

3) Human Errors

4) Longer Time-to-Market

5) Complex Rollbacks

Note: To overcome these problems we are going to automate configuration management in the project.

=> To automate configuration management we have several tools in the market

                        1) Puppet

                        2) Chef

                        3) Ansible (trending)


================
What is Ansible
================

-> It is an open source software developed by Michael DeHaan and its ownership is under RedHat.

=> Ansible was written in Python language.

-> Ansible is an automation tool that provides a way to define configuration as code.


=======================
Ansible Architecture
=======================

1) Control Node

2) Managed Nodes / Host Nodes

3) Host Inventory File

4) Playbooks
```

=> The machine which contains ansible software is called as Controlling Node.

=> The machines which are managing by Controlling Node are called as Managed Nodes/Host Nodes.

=> Host inventory file contains managed nodes information.

=> Playbook is a YML/YAML which contains set of tasks. IT is used to represent configuration as code.

===============
Ansible Setup
===============

### Steps in Git Repo : https://github.com/ashokitschool/DevOps-Documents/blob/main/11-Ansible-Setup.md

@@@ Ansible Setup Video Reference : https://youtu.be/bm1J4ED-ZUo?si=zEULHcMHY4bibUhY

===========================
Ansible Ad-Hoc Commands
===========================

=> To run ad-hoc commands we will follow below syntax

```
Syntax :  $ ansible <all/group-name/private-ip> -m <module-name> -a <args>

ex-1 : $ ansible all -m ping

ex-2 : $ ansible webservers -m ping

ex-3 : $ ansible dbservers -m ping
```

=> We have several modules in ansible to perform configuration management

```
                1) ping
                2) shell
                3) yum / apt
                4) service
                5) copy
```

```
$ ansible all -m shell -a "date"

$ ansible all -m shell -a "whoami"

$ ansible all -m shell -a "uptime"

$ ansible webservers -m yum -a "name=git" -b
```

===================
Ansible Playbooks
===================

=> Playbook is a YAML file

=> Playbook contains one or more tasks

=> Using playbook we can define what tasks to performed and where tasks to be performed.

=> We will give playbook as input for ansible control node to perform tasks in managed nodes / host nodes.

Note: To write Ansible playbooks, we should learn YAML first.

================
YML or YAML
================

=> YML/YAML stands for Yet another markup language.

=> It is used to store the data in human & machine readable format.

=> YML/YAML files will have extension as .yml or .yaml

> Official Website : https://yaml.org/

Note: indent spacing is very important in YML

============================
01 - Sample YML file data
============================

```
---
id: 101
name: Ashok
gender: Male
hobbies:
 - chess
 - music
 - cricket
 - chatting
...
```

===========================
02 - Sample YML file data
===========================

```
---
person:
 id: 101
 name: Ashok
 address:
  city: Hyd
  state: TG
  country: India
 hobbies:
  - cricket
  - music
...
```

=============================

Write YML file to represent employee data with company and job details.

emp -> id, name, company and job

company -> name

job -> exp, salary

```
---
emp:
 id: 101
 name: Raj
 company:
  name: TCS
 job:
  exp: 2 years
  salary: 2.5 LPA
...


==================
Writing Playbooks
==================

=> Playbook contains 3 sections

    1) Host Section

    2) Variable Section

    3) Task Section

=> Host Section Represents target machines to execute tasks.

=> Variables Section is used to declare variables required for playbook execution.

=> Task section is used to define what operations we want to perform using Ansible.

Note: In single playbook we can specify multiple tasks also.

=> To execute playbook we will use below syntax

        $ ansible-playbook <playbook-yml>

=================================
Playbook to ping managed nodes
=================================

---
- hosts: all
  tasks:
  - name: ping all managed nodes
    ping:
...

# It will check the syntax of a playbook
$ ansible-playbook <playbook-yml> --syntax-check

# It will display which hosts would be effected by a playbook before run
$ ansible-playbook <playbook-yml> --list-hosts

# Run playbook
$ ansible-playbook <playbook-yml>

# confirm each task before running with (N)o/(y)es/(c)ontinue
$ ansible-playbook <playbook-yml-file> --step

# Run the playbook in verbose mode
$ ansible-playbook <playbook-yml-file> -vvv


=============================================
Playbook to create a file in managed nodes
=============================================

---
- hosts: all
  tasks:
  - name: create a file
    file:
     path: /home/ansible/f1.txt
     state: touch
...
```

```
================================
Playbook to copy data to file
================================
```

```
---
- hosts: all
  tasks:
  - name: copy data to file
    copy: content="welcome to ashokit\n" dest="/home/ansible/f1.txt"
...
```

```
================================
Playbook to host static website
================================
```

1) install httpd package

2) create/copy index.html file

3) start httpd service

```
---
- hosts: webservers
  become: true
  tasks:
  - name: install httpd package
    yum:
     name: httpd
     state: latest
  - name: copy index.html file
    copy:
     src: index.html
     dest: /var/www/html/index.html
  - name: start httpd service
    service:
     name: httpd
     state: started
...
```

```
================
Handlers & Tags
================
```

-> In playbook, all tasks will be executed by default in sequential order.

=> Using Handlers we can execute tasks based on other tasks status.

Note: If 2nd task status is changed then only execute 3rd task.

-> Handlers are used to notify the tasks to execute.

=> 'notify' keyword we will use to inform handler to execute.

```
---
- hosts: webservers
  become: true
  tasks:
  - name: install httpd package
    yum:
     name: httpd
     state: latest
  - name: copy index.html file
    copy:
     src: index.html
     dest: /var/www/html/index.html
    notify:
      start httpd service
  handlers:
  - name: start httpd service
    service:
     name: httpd
     state: started
...
```

-> Using Tag we can map task to a tag-name

-> Using tag name we can execute particular task and we can skip particular task available in our playbook.

```
---
- hosts: webservers
  become: true
  tasks:
  - name: install httpd package
    yum:
     name: httpd
     state: latest
    tags:
    - install
  - name: copy index.html file
    copy:
     src: index.html
     dest: /var/www/html/index.html
    tags:
    - copy
    notify:
      start httpd service
  handlers:
  - name: start httpd service
    service:
     name: httpd
     state: started
...
```

```
# to display all tags available in playbook
$ ansible-playbook handlers_tags.yml --list-tags

# Execute a task whose tag name is install
$ ansible-playbook handlers_tags.yml --tags "install"

# Execute the tasks whose tags names are install and copy
$ ansible-playbook handlers_tags.yml --tags "install,copy"

# Execute all the tasks in playbook by skipping install task
$ ansible-playbook handlers_tags.yml --skip-tags "install"
```

=================================
What is gather facts in ansible
=================================

=> In Ansible, gathering facts refers to the process of collecting information about the target machines before executing tasks.

```
    Ex: OS, memory, cpu architecture etc....

=> This information will be collected automatically using "setup" module.

=================================
What is debug keyword in ansible
=================================

=> debug keyword is used to print a msg when playbook is getting executed.

---
- hosts: all
  gather_facts: yes
  tasks:
  - name: ping nodes
    ping:
  - name:  print os family
    debug:
     msg: "The os is {{ansible_os_family}}"
...


======================================
What is register keyword in ansible
======================================

=> register keyword in ansible allow you to capture the output of a task and store it into a variable
for later use.

Note: one task output we can register and we can use it in another task like below

---
- hosts: localhost
  tasks:
  - name: get date
    command: date
    register: date_output

  - name: print date
    debug:
     msg: "Current Date {{date_output.stdout}}"
...

==============================
Error Handling in Playbooks
==============================

=> If we get any error in task execution then playbook execution will be terminated abnormally
 (in the middle).

=> If we get any error in first task execution then remaining tasks will not be executed.

=> We can handle errors in playbook and we can continue remaining tasks execution using
'ignore_errors' concept.

---
- hosts: localhost
  tasks:
  - name: get date
    command: dates
    ignore_errors: yes
  - name: get whoami
    command: whoami
...
```

```
============
Variables
============

=> Variables are used to store the data in key-value format

  Ex: id=100
      name=ashok
      age=20
      gender=male

=> In Ansible, we can use variables in 4 ways

1) Runtime variables

2) Playbook variables

3) Group variables

4) host variables

==================
Runtime Variables
==================

=> We can pass variable value in runtime like below

---
- hosts: webservers
  become: true
  tasks:
  - name: install package
    yum:
     name: "{{package_name}}"
     state: latest
...

$ ansible-playbook <yml> --extra-vars package_name=httpd

===================
Playbook Variables
===================

=> We can declare variable value with in the playbook like below

---
- hosts: webservers
  become: true
  vars:
   package_name: httpd
  tasks:
  - name: install package
    yum:
     name: "{{package_name}}"
     state: latest
  - name: Print a msg
    debug:
     msg: "{{package_name}} installed sucessfully"
...


=========================================================================
Requirement : Write ansible playbook to install below softwares
```

```
ungrouped servers : httpd

webservers group : java

db servers group : git
============================================================

=> To achieve above requirement we need to use group_vars and host_vars concept

=> We need to supply variable value based on group name and based on host name

---
- hosts: all
  become: true
  tasks:
  - name: install package
    yum:
     name: "{{package_name}}"
     state: latest
...

============
Group Vars
============

=> group_vars concept is used to specify variable value for group of managed nodes as per inventory
file group name.

=> Managed nodes we are configuring host inventory file like below


172.31.5.184

[webservers]
172.31.5.185

[dbservers]
172.31.5.186

=> While executing above playbook for webservers group i want to pass one package name and for
dbservers group i want to pass another package name.

Note: We need to create variables based on group name like below

ex:

webservers.yml
dbservers.yml

Note: group_vars related yml files we should create in host inventory file location

host inventory file location : /etc/ansible/hosts

webservers group variable file : /etc/ansible/group_vars/webservers.yml

dbservers group variable file : /etc/ansible/group_vars/dbservers.yml


===============
Host Variables
===============

=> host variables are used to specify variable value at host level (or) machine level
=> host vars we will create in below location

Location : /etc/ansible/host_vars

      mn-3  : /etc/ansible/host_vars/mn-3.yml

      mn-4  : /etc/ansible/host_vars/mn-4.yml


Note-1: host variables will take precendence over group variables

Note-2: Variables defined in playbook override both host_vars and group_vars.


    playbook variable ====> host var  ====> group var


=================================================================
Write a playbook to install java in different OS family machines
=================================================================

MN-1 : Amazon linux ==> Red Hat family  (yum)

MN-2 : Ubuntu ==> Debian family (apt)

Note: In this scenario our task should execute based on os_family. To check conditions in playbook we
will use 'when' keyword.


---
- hosts: all
  gather_facts: yes
  tasks:
  - name: install java in Red Hat family
    yum:
     name: java
     state: latest
    when: ansible_os_family == 'Red hat'

  - name: install java in Debian family
    apt:
     name: java
     state: latest
    when: ansible_os_family == 'Debian'
...

===============
Ansible Vault
===============

=> It is used to secure our playbooks

=> Using Ansible vault concept, we can encrypt & decrypt our playbooks

Encryption : Convert data from readable format to un-readable format

DeCryption : Convert data from un-readable format to readable format

# Encrypt our playbook
$ ansible-vault encrypt <yml-file-name>

Note: To encrypt a playbook we need to set one vault password

# see encrypted playbook
cat <yml-file-name>
```

```
# see orignal content of playbook
ansible-vault view <yml-file-name>

# to edit encrypted playbook
ansible-vault edit <yml-file-name>

# how to run encrypted playbook
ansible-playbook <yml-file-name> --ask-vault-pass


===============
Ansible Roles
===============

=> If we write more functionalities in single playbook then it will become difficult to manage that
playbook.

=> By Using Roles concept we can break down large playbooks into smaller chunks.

=> Below playbook we will divide into small chunks using Role concept

---
- hosts: webservers
  become: true #use it if you need sudo priviliges
  tasks:
  - name: install httpd package
    yum:
     name: httpd
     state: latest
  - name: copy index.html file
    copy:
     src: index.html
     dest: /var/www/html/index.html
  - name: start httpd service
    service:
     name: httpd
     state: started
...

# To create a role we can use below command

Syntax : $ ansible-galaxy init <role-name>

==========================
Working with Ansible Role
==========================

### Step-1: Connect with control node and switch to ansible user

$ sudo su ansible
$ cd ~

### Step-2 : Create a role using 'ansible-galaxy'

$ mkdir roles

$ cd roles

$ ansible-galaxy init apache

$ sudo yum install tree

$ tree apache
```

```
### Step-3 : Create tasks inside "tasks/main.yml" like below

---
# tasks file for apache
- name: install httpd
  yum:
    name: httpd
    state: latest
- name: copy index.html
  copy:
    src=index.html
    dest=/var/www/html/
  notify:
    - restart apache
...

### Step-4 : Copy required files into "files" directory

Note: keep index.html file in files directory

### Step-5 : configure handlers in "handler/main.yml"

---
# handlers file for apache
- name: restart apache
  service:
    name: httpd
    state: restarted
...

Note: With above 5 steps our "apache"  role is ready now we can execute that role like below

### Step-6 : Create main playbook to invoke role using role name

$ cd ~
$ vi invoke-roles.yml

---
- hosts: all
  become: true
  roles:
    - apache
...

Note: Roles will provide abstraction for ansible configuration in a modular and re-usable format.

==========================
Ansible Classes Summary
==========================

1) What is Configuration management

2) Ansible Introduction

3) Ansible Architecture

4) Ansible Setup

5) Ansible Ad-Hoc commands

6) Ansible Modules

7) YML file

8) Playbooks
```

```
 9) Handlers & Tags

 10) Variables
      - Playbook variables
      - Runtime variables
      - host vars
      - group vars

 11) Ansible Vault

 12) Ansible Roles

 14) gather_facts + register + debug + when

 15) ignore errors in playbook execution

 16) Ansible Tower (theory)
```