ANSIBLE

- Ansible is an open source software that automates software provisioning, configuration management, and application deployment.
- Ansible is commonly used for tasks like software installation, configuration, and system updates across multiple servers or devices in a network.
- Orchestration, Security and compliance.
- Uses YAML Scripting language which works on KEY-VALUE PAIR
- Ansible GUI is called as Ansible Tower. It was just Drag and Drop.
- It helps reduce manual work, improve consistency, and save time in managing complex environments.

The Keys Features of Ansible:

Agentless: There is no software or agent to be installed on the client that communicates back to the server.

Simple and extensible: Ansible is written in Python and uses YAML for playbook language, both of which are considered relatively easy to learn.

PLAYBOOK:

Ansible playbooks are a way to send commands to remote computers in a scripted way. Instead of using Ansible commands individually to remotely configure computers from the command line, you can configure entire complex environments by passing a script to one or more systems.

1. Playbooks in ansible are written in YAML language

2. It is human readable & serialization language commonly used for configuration files.

3. You can write codes consists of vars, tasks, handlers, files, templates and roles.

4. Each playbook is composed of one or more modules in a list.

5. Playbooks are mainly divided into sections like

6. **TARGET SECTION:** Defines host against which playbooks task has to be executed.

7. **VARIABLE SECTION:** Defines variables.

8. **TASK SECTION:** action you are performing.

WRITE A PLAYBOOK TO INSTALL PACKAGES USING YUM MODULE IN DEV GROUP:

```yaml
---
- hosts: test
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: Install Git using YUM
      yum:
        name: git
        state: present
```

🔍 Explanation:

- **hosts: test**: Refers to a group defined in your inventory file.
- **user: ansible**: The SSH user for remote login.
- **become: yes**: Uses privilege escalation (like sudo).
- **yum**: **Module** to install Git using the package manager.
- **state**: **present**: Ensures Git is installed (if not already).

PLAYBOOK TO INSTALL WEB SERVER & START THE WEB SERVER:

```yaml
---
- hosts: all
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install web server in all slaves
      action: yum name=httpd state=present

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

PLAYBOOK USING MULTIPLE VARIABLES:

```yaml
---
- hosts: dev
  connection: ssh

  vars:
    abc: git
    xyz: maven

  tasks:
    - name: Install Git
      yum:
        name: "{{ abc }}"
        state: present

    - name: Install Maven
      yum:
        name: "{{ xyz }}"
        state: present
```

PLAYBOOK TO ADD VARIABLES DYNAMICALLY:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install git
      action: yum name='{{abc}}' state=present
```

for single var: ansible-playbook one.yml --extra-vars "abc=git"

for multiple vars: ansible-playbook one.yml --extra-vars "abc=git def=maven"

PLAYBOOK TO ADD MULTIPLE USERS:

```yaml
---
# LOOPS
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: Add list of users in my nodes
      user:
        name: "{{ item }}"
        state: present
      with_items:
        - raham
        - mustafa
        - shafi
        - nazeer
```

PLAYBOOK USING HANDLERS:

```yaml
---
# HANDLER
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: Install httpd server on CentOS
      yum:
        name: httpd
        state: installed
      notify: restart httpd

  handlers:
    - name: restart httpd
      service:
        name: httpd
        state: restarted
```

PLAYBOOK USING CONDITIONS:

```yaml
---
# CONDITIONS
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: Install apache server for Debian family
      command: apt-get -y install apache2
      when: ansible_os_family == "Debian"

    - name: Install apache server for RedHat family
      command: yum install httpd -y
      when: ansible_os_family == "RedHat"
```

WRITE A PLAYBOOK USING TAGS:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: Installing git
      yum:
        name: git
        state: present
      tags: install

    - name: Uninstalling git
      yum:
        name: git
        state: absent
      tags: uninstall
```

TO EXECUTE A SINGLE TASK: ansible-playbook abc.yml --tags tagname

TO EXECUTE A MULTIPLE TASK: ansible-playbook abc.yml --tags tagname1,tagname2

TO SKIP A TASK: ansible-playbook abc.yml --skip-tags “uninstall”

PLAYBOOK FOR CREATING A FILE:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: creating a file
      file:
        path: "jenkins.txt"
        state: touch
```

PLAYBOOK FOR CREATING A DIRECTORY:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: creating a file
      file:
        path: "folder"
        state: directory
```

PLAYBOOK FOR ENTERING A DATA IN A FILE:

```yaml
---
- hosts: dev
  tasks:
    - name: inserting a data in a file
      copy:
        dest: "devops.txt"
        content: |
          hi this is devops file
          we are inserting the data ij a file
          using ansible playbook
```

PLAYBOOK TO CHANGE THE PERMISSIONS OF A FILE:

```yaml
---
- hosts: dev
  tasks:
    - name: change permissions to a file
      file:
        path: "devops.txt"
        state: touch
        mode: 777
```

PLAYBOOK TO COPY A FILE:

```yaml
---
- hosts: dev
  connection: ssh
  become: yes

  tasks:
    - name: Copy Jenkins file from control node to remote host
      copy:
        src: jenkins.yml
        dest: /home/ansible/jenkins.yml
```

✅ src refers to the file path on the **Ansible control node**.

✅ dest should ideally specify an absolute path on the **remote node**.

PLAYBOOK TO DEPLOY A WEBSITE:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: install httpd
      action: yum name=httpd state=present

    - name: restart httpd
      service: name=httpd state=restarted

    - name: create a file
      file:
        path: "/var/www/html/index.html"
        state: touch

    - name: enter data in a file
      copy:
        dest: "/var/www/html/index.html"
        content: |
          <h1>this is my webapplication, i have deployed using ansible </h1>
```

PLAYBOOK TO GET A CODE FROM GITHUB(PRIVATE-REPO):

```yaml
---
- hosts: localhost
  become: yes

  tasks:
    - name: link
      git:
         repo: 'https://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@github.com/devops0014/ansible.git'
         dest: "/home/mygitcode"
```

SYNTAX: token@github.com/username/repo.git

PLAYBOOK USING DEBUG MODULE:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - debug:
        msg: "os family for {{ansible_fqdn}} is {{ansible_os_family}}"
```

PLAYBOOK USING DEBUG MODULE + REGISTER:

```yaml
---
- hosts: dev
  user: ansible
  become: yes
  connection: ssh

  tasks:
    - name: get users
      command: cat /etc/passwd
      register: output

    - debug:
        msg: "users list in the ansible is {{output.stdout}}"
```

ANSIBLE ROLES:

Roles in Ansible are a way to **organize playbooks** into reusable, modular components. They help break down complex configurations into smaller, manageable pieces.

Structure of a Role:

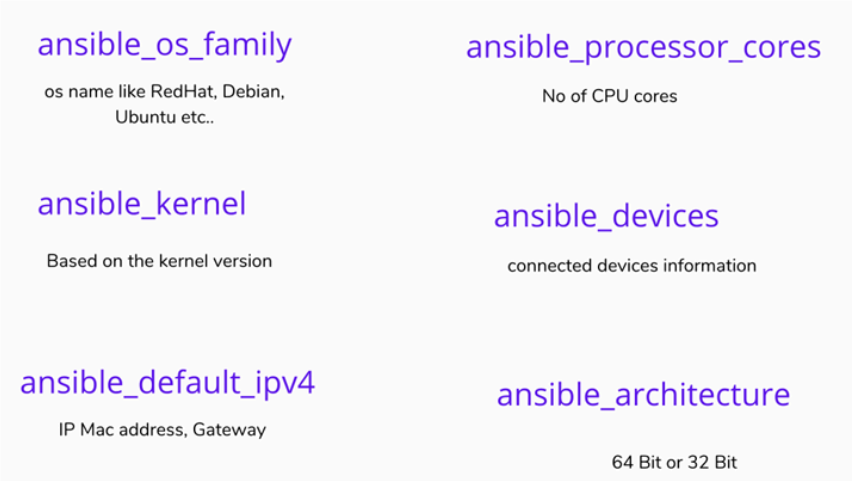
```
roles/
└── myrole/
    ├── tasks/
    ├── handlers/
    ├── files/
    ├── templates/
    ├── vars/
    ├── defaults/
    └── meta/
```

Using a Role in a Playbook:

```yaml
# myrole/tasks/main.yml
- name: Install nginx
  apt:
    name: nginx
    state: present
```

```yaml
# site.yml
- hosts: web
  become: yes
  roles:
    - myrole
```

ANSIBLE SETUP MODULES:



ADHOC COMMANDS:

Ansible ad-hoc commands are quick, one-time instructions you give to Ansible on the command line to perform simple tasks on remote servers. These commands are not part of Ansible's usual automation playbook and are typically used for tasks like running a single command, checking server status, or making minor changes without writing full automation scripts. Ad-hoc commands are handy for immediate, one-o tasks.

EX: ansible Test -ba “yum install httpd -y”

ANSIBLE MODULES:

Ansible modules are like individual commands or tools that perform specific tasks on target machines. They are the building blocks for Ansible automation. Modules can do things like create files, install software, restart services, and more.

EX: ansible Test -b -m yum -a “pkg=httpd state=present” (install: present)

ANSIBLE GALAXY:

Ansible Galaxy is a website and command-line tool for sharing and managing collections of Ansible roles and playbooks. In simple terms, it's like an online marketplace or repository for Ansible automation content.

- ansible-galaxy init raham
- ansible-galaxy search elasticsearch
- ansible-galaxy search elasticsearch --author alikins
- ansible-galaxy install alikns.elasticsearch
- cd /home/ansible/.ansible/roles

ANSIBLE VALUT:

Ansible Vault is a feature of the Ansible automation tool that is used to securely encrypt sensitive data, such as passwords, API keys, and other secrets, so that they can be safely stored and shared within Ansible playbooks and roles.

USE CASES:

- Encryption
- Secure Storage
- Password Prompt
- Automation
- Secrets Management

COMMANDS FOR ANSIBLE PASSWORD

- ansible-vault create vault.yml : creating a new encrypted playbook.
- ansible-vault edit vault.yml : Edit the encrypted playbook.
- ansible-vault rekey vault.yml : To edit the password.
- ansible-vault view vault.yml : To view the playbook without decrypt.
- ansible-vault encrypt vault.yml : To encrypt the existing playbook.
- ansible-vault decrypt vault.yml : To decrypt the encrypted playbook.