ANSIBLE CLASS - 3

ANSIBLE TAGS:

```
---
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: installing git
      action: yum name=git state=present
      tags: install
    - name: uninstalling git
      action: 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”

ANSIBLE CONDITIONS

```
--- # CONDITIONS
- hosts: remo
  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"
```

ANSIBLE HANDLERS

```yaml
--- # HANDLER
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
        - name: install httpd server on centos
          action: yum name=httpd state=installed
          notify: restart httpd
  handlers:
          - name: restart httpd
            action: service name=httpd state=restarted
~
```

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.

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-off tasks.

- ansible remo -a “ls” [remo: Group name, -a: argument, ls: command]
- ansible remo [0] -a “touch file1”
- ansible all -a “touch file2”
- ansible remo -a “sudo yum install httpd -y”
- ansible remo -ba “yum install httpd -y” (b: become you will become sudo user)
- ansible remo -ba “yum remove httpd -y”
- ansible remo -b -m yum -a “pkg=httpd state=present” (install: present)
- ansible remo -b -m yum -a “pkg=httpd state=latest” (update: latest)
- ansible remo -b -m yum -a “pkg=httpd state=absent” (uninstall: absent)
- ansible remo -b -m service -a “name=httpd state=started” (started: start)
- ansible remo -b -m user -a “name=raj” (to check go to that servers and sudo cat /etc/passwd).
- ansible remo -b -m copy -a “src=filename dest=/tmp” (to check go to that server and give ls /tmp)

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.