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.

WHY ANSIBLE:

While managing the multiple servers its hard to keep their configuration identical. If you have multiple servers which needs to configure the same setup in all. while doing the one to one server their might be a chances to miss some configuration steps in some servers. Thats why automation tools come into play! The automation tools like Ansible, Chef, Puppet and SaltStack all are based on a same principle.

DESCRIBE THE DESIRED STATE OF THE SYSTEM

MASTER-SLAVE CONCEPT :

STEP-1: LAUNCH 5 INSTANCE (1-MASTER, 4-SLAVE)

STEP-2: INSTALL ANSIBLE, PYTHON AND PIP ON MASTER SERVER

amazon-linux-extras install ansible2 -y

yum install python-pip -y

STEP-3: SET A PASSWORD TO USER IN ROOT SERVER (passwd root)

STEP-4: NOW WE HAVE TO SAY YES TO PASSWORD AUTHNETICATION

vi /etc/ssh/sshd_config ----> 63 line (63gg)

change the password authentication from **no to yes**

STEP-5: RESTART SSHD (systemctl restart sshd)

NOTE: REPETE ALL THESE STEPS ON ALL SLAVE SERVERS FROM STEP-3, 4 & 5

STEP-6: GENERATE A KEY IN ANSIBLE USER ON MASTER SERVER (ssh-keygen)

It will generate 2 keys (public & private)

STEP-7: COPY THE PUBLIC KEY TO ALL SLAVE SERVERS (ssh-copy-id root@slave_ip)

NOW ITS TIME TO CHANGE ANSIBLE CONFIGURATIONS:

STEP-8 : ENBALE ANSIBLE INVENTORY AND SUDO USER (vi /etc/ansible/ansible.cfg)

```
[defaults]

# some basic default values...

inventory      = /etc/ansible/hosts
#library        = /usr/share/my_module
#module_utils   = /usr/share/my_module
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plu
#forks          = 5
#poll_interval  = 15
sudo_user      = root
#ask_sudo_pass = True
#ask_pass      = True
#transport      = smart
#remote_port    = 22
#module_lang    = C
#module_set_locale = False
```

save & quit from the file

STEP-14: ADD INVENTORIES (vi /etc/ansible/hosts)

```
# Ex 2: A collection of hosts belonging to the 'webservers' group
[dev]
172.31.34.110
172.31.35.94

[test]
172.31.38.217
172.31.40.252
## [webservers]
```

HERE dev & test is the group names

save & quit from the file

STEP-15: TO CHECK WITH SLAVE SERVER CONNECTION

to check the connection : ansible all --list-hosts

- To see the list of hosts in inventory : ansible all --list-hosts
- To see the list of particular group hosts in inventory : ansible group_name --list-hosts

- To see the 1st hosts in inventory : ansible all[0] --list-hosts
- To check the network connection between master & slave : ansible all -m ping

PLAYBOOKS:

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 perfoming.

1. **WRITE A PLAYBOOK TO INSTALL GIT IN DEV GROUP:**

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

2. WRITE A PLAYBOOK TO INSTALL JAVA1.8.0 ON ALL THE SERVERS

```
---
- hosts: all
  user: ansible
  become: yes
  connection: ssh
  tasks:
    - name: install java
      action: yum name=java-1.8.0-openjdk state=present
```

3. WRITE A PLAYBOOK TO INSTALL WEB SERVER & START THE WEB SERVER:

```
---
- 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
```

4. WRITE A PLAYBOOK WITH VARIABLE:

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

  vars:
    abc: git

  tasks:
    - name: install git in my slave server
      action: yum name={{abc}} state=present
```

5. WRITE A PLAYBOOK WITH MULTIPLE VARIABLES:

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

  vars:
    abc: git
    xyz: maven

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

    - name: install maven
      action: yum name={{xyz}} state=present
```

6. WRITE A PLAYBOOK TO ADD VARIABLES DYNAMICALLY:

```
---
- 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"

7. WRITE A PLAYBOOK TO INSTALL PACKAGES ON DIFFERENT WAYS:

```yaml
---
- hosts: test
  connection: ssh

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

    - name: install java1.8.0
      yum: name=java-1.8.0-openjdk state=present

    - name: install java11
      command: amazon-linux-extras install java-openjdk11 -y
```

8. Passing a Varaible file - A Varaible can be defined in a variable file and can be passed to a playbook using the include

one.yml

```yaml
---
- set_fact: abc=httpd
- name: install Apache
  yum: name=httpd state=present
```

two.yml

```yaml
---
- hosts: dev
  become: yes
  tasks:
    - include: one.yml
    - name: install git
      service: name='{{abc}}' state=restarted
```

9. WRITE A PLAYBOOK TO ADD MULTIPLE USERS:

```yaml
- # LOOPS
hosts: remo
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
```

10. WRITE A PLAYBOOK USING HANDLERS:

```
--- # 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
~
```

11. WRITE A PLAYBOOK USING CONDITIONS:

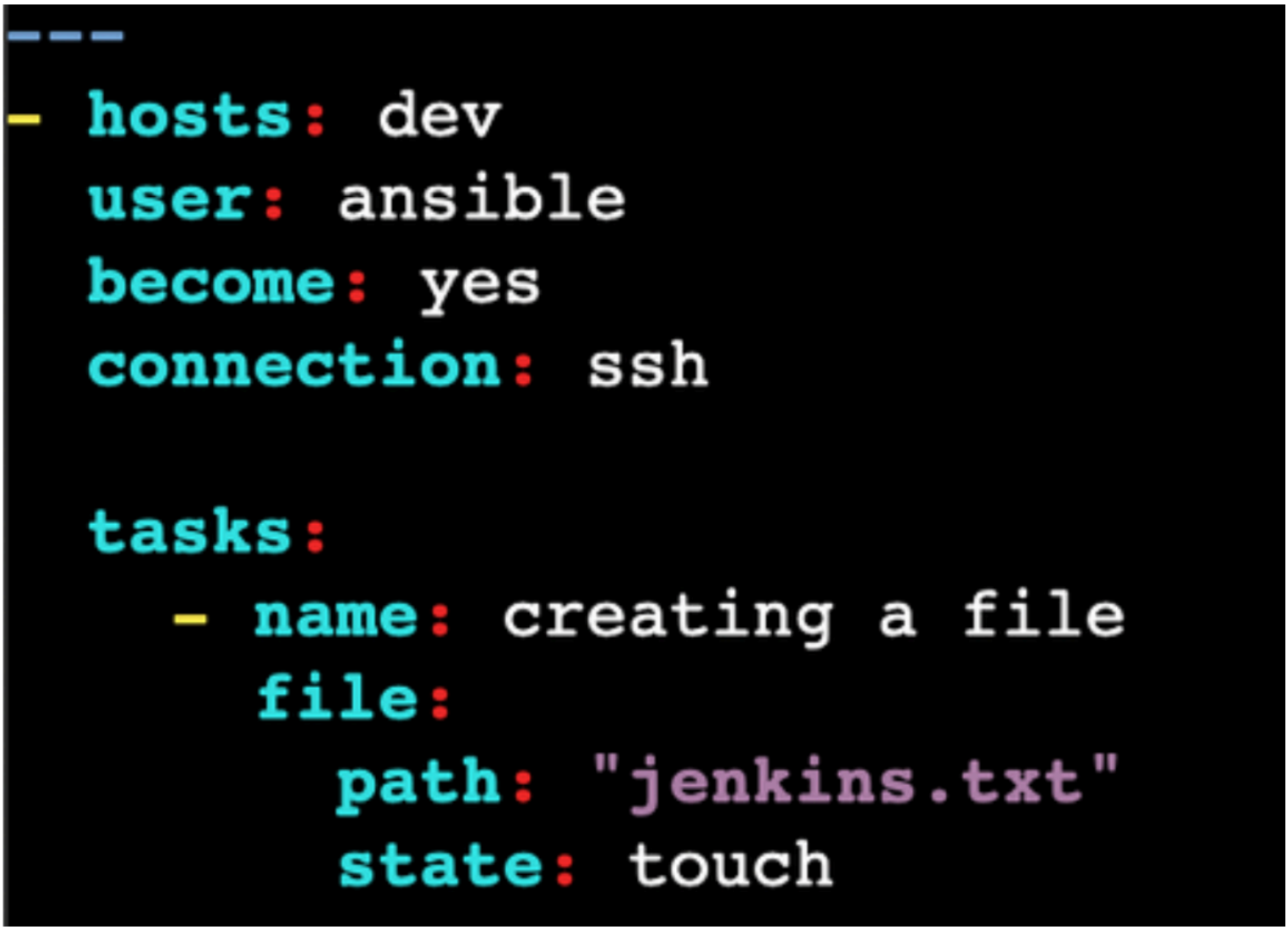
```
--- # CONDITIONS
- hosts: remo
  user: ansible
  become: yes
  connection: ssh
  tasks:
          - name: Install apache server for debian family
            command: apt-get-y install apache2
            when: ansible_os_family== "Debian"
          - name: install apache server for redhat family
            command: yum install httpd -y
            when: ansible_os_family== "RedHat"
```

12. WRITE A PLAYBOOK USING 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”

13. WRITE A PLAYBOOK FOR CREATING A FILE:



14. WRITE A PLAYBOOK FOR CREATING A FILE:

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

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

15. WRITE A 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
```

16. WRITE A 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
```

17. WRITE A 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>
```

18. WRITE A PLAYBOOK TO SETUP JENKINS:

```yaml
---
- hosts: localhost
  connection: ssh


  tasks:
    - name: getting links from jenkins.io
      get_url:
        url: https://pkg.jenkins.io/redhat-stable/jenkins.repo
        dest: /etc/yum.repos.d/jenkins.repo

    - name: import key from jenkins.io
      ansible.builtin.rpm_key:
        state: present
        key: https://pkg.jenkins.io/redhat-stable/jenkins.io.key

    - name: install java-11
      command: amazon-linux-extras install java-openjdk11 -y

    - name: install jenkins
      action: yum name=jenkins state=present

    - name: restart jenkins
      service: name=jenkins state=restarted
```

19. WRITE A PLAYBOOK TO SETUP TOMCAT:

```yaml
---
- hosts: ops
  connection: ssh

  tasks:
    - name: gettling link
      get_url:
        url: https://dlcdn.apache.org/tomcat/tomcat-9/v9.0.71/bin/apache-tomcat-9.0.71.tar.gz
        dest: "/root/"

    - name: untar file
      command: tar -zxvf apache-tomcat-9.0.71.tar.gz

    - name: rename the file
      command: mv apache-tomcat-9.0.71 tomcat

    - name: java11
      command: amazon-linux-extras install java-openjdk11 -y

    - name: context.xml file
      template:
        src: context.xml
        dest: "/root/tomcat/webapps/manager/META-INF/context.xml"
```

```yaml
    - name: add credits
      template:
        src: tomcat-users.xml
        dest: "/root/tomcat/conf/tomcat-users.xml"


    - name: start tomcat
      shell: nohup ./tomcat/bin/startup.sh
```

20. WRITE A PLAYBOOK TO COPY A FILE:

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

  tasks:
    - name: copy files from master to slave
      copy:
        src: jenkins.yml
        dest: jenkins.yml
```

20. WRITE A PLAYBOOK TO GET A CODE FROM GITHUB(PUBLIC-REPO):

```yaml
---
- hosts: localhost
  become: yes
  tasks:
    - name: getting code from git
      git:
        repo: "https://github.com/devops0014/pubg.git"
        dest: "/home/mycode"
```

21. WRITE A 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

22. WRITE A 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}}"
```

23. WRITE A PLAYBOOK TO SEE LIST OF USERS:

```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:

Ansible roles are a way to organize and structure your Ansible playbooks in a more modular and reusable manner. They provide a means to group related tasks, variables, and files together, making your playbooks more organized and easier to manage. Roles can be thought of as a collection of tasks, templates, and variables that are designed for a specific purpose or function, such as setting up a web server, configuring a database, or managing a specific application.

1. Create the role directory structure:

You can create a role using the ansible-galaxy command or by manually creating the directory structure. Let's create the directory structure manually:

roles/

├── webserver/

│ ├── tasks/

│ │ └── main.yml

│ ├── handlers/

│ │ └── main.yml

│ ├── templates/

│ │ └── index.html.j2

│ ├── vars/

```
    │   └── main.yml
    ├── defaults/
    │   └── main.yml
    └── meta/
        └── main.yml
```

2. Define the role tasks in **roles/webserver/tasks/main.yml:**

```
---
- name: Install Apache web server
  yum: name=httpd state=present

- name: Ensure Apache service is running
  service: name=httpd state=started
```

3. Define role variables in **roles/webserver/vars/main.yml:**

```
---
apache_port: 80
```

4. Create a handler in **roles/webserver/handlers/main.yml** (optional) to restart the Apache service if needed:

```
---
- name: Restart Apache
  service: name=httpd state=restarted
```

5. Create a template for the index page in **roles/webserver/templates/index.html.j2** (optional):

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Welcome to My Website</title>
</head>
<body>
    <h1>Welcome to my web server!</h1>
</body>
</html>
```

6. Specify any necessary metadata for the role in **roles/webserver/meta/main.yml:**

```
---
dependencies: []
```

7. With this role structure in place, you can now use the webserver **role** in your Ansible playbook by specifying it in the roles section. For example:

```
---
- name: Configure Web Server
  hosts: web_servers
  become: yes

  roles:
    - webserver
```

ANSIBLE SETUP MODULES:

ansible_os_family

os name like RedHat, Debian, Ubuntu etc..

ansible_processor_cores

No of CPU cores

ansible_kernel

Based on the kernel version

ansible_devices

connected devices information

ansible_default_ipv4

IP Mac address, Gateway

ansible_architecture

64 Bit or 32 Bit

After executing a playbook, if you want to see the output in json format

ansible -m setup private_ip

if you want to apply a see particular output, you can apply filter.

- ansible -m setup -a "filter=ansible_os_family" private_ip
- ansible -m setup -a "filter=ansible_devices" private_ip
- ansible -m setup -a "filter=ansible_kernel" private_ip

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 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.

- 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 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.