**ANSIBLE:**

**1) Ansible Architecture:**

Ansible follows a simple architecture:

Control Node: Machine where Ansible is installed and commands are executed.

Managed Nodes: Machines managed by Ansible.

Inventory: Configuration file defining managed nodes.

Playbooks: Files containing Ansible automation tasks written in YAML.

**2) Difference Between Normal YAML File and Ansible Playbook:**

Normal YAML File: Contains structured data following YAML syntax.

Ansible Playbook: YAML file containing a set of tasks, plays, roles, and configurations for automating infrastructure tasks using Ansible.

**3) Ansible Playbook and its Terms:**

Playbook: YAML file containing a collection of plays, where each play defines a set of tasks to be executed on hosts.

Terms:

hosts: Defines target hosts where tasks will run.

tasks: Specifies actions or commands to be executed.

roles: Organizational units for grouping tasks, handlers, variables, etc.

vars: Variables used in the playbook.

**4) Handlers and Notifiers in Ansible:**

Handlers: Special tasks triggered only if another task in the play changes something. For example, restart a service if the configuration file changes.

Notifiers: There's no specific "notifier" concept in Ansible. Handlers are commonly used for such purposes.

**5) Registers in Ansible:**

Registers: Used to capture the output of a task and save it to a variable for later use within the playbook.

**6) Various Ansible Modules:**

Modules in Ansible are the building blocks of playbooks. Common modules include apt, yum, file, service, command, shell, template, copy, etc.

**7) Using when Condition and Loops in Ansible:**

when Condition: Used to execute tasks conditionally based on defined criteria.

Loops: Achieved using with\_items or loop keyword to perform tasks iteratively.

**8) Playbook for Service Operations:**

Here's an example playbook:

yaml

Copy code

---

- name: Manage Service

hosts: your\_hosts

tasks:

- name: Start service

service:

name: your\_service\_name

state: started

- name: Stop service

service:

name: your\_service\_name

state: stopped

- name: Check service health

shell: /usr/bin/check\_health\_command

**9) Ansible Ad-hoc Commands:**

Ad-hoc commands are quick one-liners to perform tasks on remote hosts without writing a playbook.

Example: ansible all -i hosts -m ping (ping all hosts).

**10) Ansible Roles and Folder Structure:**

Roles: Organizational units for organizing tasks, variables, files, etc., in a structured manner.

Folder Structure:

roles/

my\_role/

tasks/

handlers/

files/

templates/

vars/

defaults/

**11) Ansible Vault and Storing Secret Files:**

Ansible Vault: Tool to encrypt sensitive data or files like passwords, keys, etc., in Ansible playbooks.

Secret files are stored encrypted and decrypted when needed using vault password.

**12) SSH Connectivity in Ansible:**

Ansible uses SSH to connect to managed nodes. It uses SSH keys or passwords configured in inventory files or Ansible configuration.

**Ansible Playbook Showing Handlers:**

An example of an Ansible playbook with handlers:

yaml

Copy code

---

- hosts: servers

tasks:

- name: Ensure Apache is installed

package:

name: apache2

state: present

notify: restart apache

handlers:

- name: restart apache

service:

name: apache2

state: restarted

**4. Ansible Variable Precedence:**

Ansible variable precedence follows a specific order:

Extra Variables: Passed through the command line.

Variables defined in Inventory: Defined within inventory files.

Variables in Playbooks: Defined within playbooks.

Role Defaults: Default variables set within roles.

Variables defined in Role Vars: Variables explicitly defined for a role.

Set\_facts task results: Variable set by set\_fact tasks.

Host facts: Information gathered about hosts.

**7. Ansible Roles and Their Advantages:**

Roles: In Ansible, roles are a way to organize tasks, handlers, variables, and other files in a structured format, making it easier to reuse and share.

Advantages: Roles provide modularity, reusability, and organization to playbooks. They allow you to encapsulate related tasks, making it easier to manage and maintain complex configurations. Roles can be shared and reused across different playbooks, promoting best practices and consistency in deployments.

**2. Different Groupings of Ansible Inventory Files:**

Static Inventory: Contains a list of hosts grouped manually.

Dynamic Inventory: Generated dynamically using external scripts or tools.

INI Format: Simple textual representation with groups and hosts.

YAML Format: Hierarchical representation using YAML syntax.

**6. Write an Ansible playbook to deploy an application and check its URL:**

*---*

*- hosts: your\_hosts*

*tasks:*

*- name: Deploy application*

*# Your deployment tasks here*

*- name: Check if app URL is up*

*uri:*

*url: http://your\_app\_url*

*return\_content: yes*

*register: response*

*- name: Print response*

*debug:*

*msg: "Application URL is up with status code {{ response.status }}"*

*Replace your\_hosts, your\_app\_url, and the deployment tasks with your specific values and tasks.*

Step 1:- Launch two ec2 machines named master and client

Step 2:-Create user tomcat in both the ec2 machines master and client

Step 3:-add passwd to user tomcat in both the ec2 machines master and client

Step 4:-in master and client change the sudo file/ add user tomcat to sudo file

Step 5:-in master and client change the sshd config file / make password auth yes

Step 6:-sudo systemctl restart sshd

Step 7:-generate ssh-keygen in master

step 8:-copy public key from master to client/ ssh-copy-id 172.x.x.x

step 9:-ansible --version

step 10:-go to host file in master and add client ip

step 11:-go to client authorized\_keys and change permission to chmod 600 authorized\_keys

step 12:- go to master and login as tomcat user and give ansible all -m ping

#create 2 ec2 master and client

#login to both master and client and perform below

sudo su

sudo amazon-linux-extras install ansible2 -y

useradd ansible

passwd ansible

visudo

add user ansible

cd /etc/ssh/

vi sshd\_config

enable authontication yes

**sudo systemctl restart sshd.service**

login as ansible user in master

ssh-keygen

ssh-copy-id 172.31.38.237

123456

cd /etc/ansible/

sudo su

vi hosts

/home/ansible/.ssh chmod 600 authorized\_keys

ansible all -m ping

—--------------------------------------

ansible all -m ping

To check syntax error:-ansible-playbook -i hosts k8.yml --syntax-check

To run playbook:-ansible-playbook -i hosts k8.yml -vv

To create a user:-ansible -i hosts all -m user -a "name=test state=present" --become

#here if we add become it will run as root user

above are all ad-hoc commands

#sample playbook -1

- hosts: all

become: true

tasks:

- name: delete old deployment

command: kubectl delete -f /home/ubuntu/deploy/.yml

- name: delete old service

command: kubectl delete -f /home/ubuntu/service.yml

- name: create new deployment

command: kubectl apply -f /home/ubuntu/deploy.yml

- name: create new service

command: kubectl apply -f /home/ubuntu/service.yml

—--------------------------------------------------------------------------------------------

**What are Ansible facts?**

Ansible facts are pieces of information about remote systems that Ansible gathers automatically when it connects to them. These facts include details about the system's hardware, network configuration, OS type and version, etc. They are gathered using the setup module and can be used within playbooks to make decisions based on the state of the target machine.

**How to manage an order of executions in a playbook?**

In an Ansible playbook, tasks are executed in the order they are written. You can control the order of task execution by simply ordering the tasks in the desired sequence within the playbook. Additionally, you can use tags, dependencies, and conditionals to fine-tune execution order.

**How is Ansible different from other configuration management tools?**

Ansible is agentless, meaning it does not require any software to be installed on the target machines. It uses SSH for communication, making it lightweight and easy to set up. Ansible also emphasizes simplicity and ease of use with its YAML-based playbooks, in contrast to more complex domain-specific languages used by some other tools.

**How does Ansible ensure security for targeted hosts?**

Ansible uses SSH for secure communication between the control node and managed nodes. Additionally, sensitive data can be encrypted using Ansible Vault, ensuring that passwords and other sensitive information are protected.

**What are Ansible modules?**

Ansible modules are discrete units of code that can perform specific tasks. They are the building blocks for Ansible playbooks and can manage system resources, execute commands, manage files, install packages, etc. Examples include the copy, yum, and service modules.

**What is the use of ad-hoc commands?**

Ad-hoc commands in Ansible are used for quick, one-off tasks without the need to write a full playbook. They are useful for executing simple commands across multiple machines, such as restarting a service or copying a file.

**How to handle errors or executions in playbooks?**

Errors in playbooks can be handled using the failed\_when and ignore\_errors directives. The failed\_when directive allows you to define conditions under which a task is considered to have failed, while ignore\_errors: yes will allow the playbook to continue executing subsequent tasks even if the current task fails.

**There is something called ignore error. What does it stand for?**

The ignore\_errors directive, when set to yes, tells Ansible to ignore failures of the task and continue with the next tasks in the playbook. This can be useful in scenarios where a failure is acceptable and should not stop the entire execution.

**Managing sensitive information in Ansible?**

Sensitive information can be managed in Ansible using Ansible Vault, which allows you to encrypt entire files or variable values within a playbook. This ensures that sensitive data such as passwords, API keys, and other secrets are not exposed in plain text.

**What is ignore and what does it stand for?**

The term "ignore" in the context of Ansible usually refers to the ignore\_errors directive, which allows tasks to continue execution even if a previous task has failed.

**Why do we have the serial keyword in Ansible?**

The serial keyword in Ansible playbooks allows you to control the number of hosts on which tasks are executed concurrently. This can be useful for rolling updates, where you want to update a few servers at a time rather than all at once.

**How to organize and structure a playbook?**

A well-structured playbook can be organized using roles, which group related tasks, files, templates, and variables. Additionally, playbooks can be broken into multiple files and include other playbooks using the include or import\_playbook directives. This helps keep playbooks modular and maintainable.

**What is dynamic inventory in Ansible?**

Dynamic inventory in Ansible allows you to pull inventory information from external sources, such as cloud providers (AWS, Azure, GCP), databases, or other services. This is useful for environments where the infrastructure changes frequently.

**How does Ansible handle Windows systems?**

Ansible can manage Windows systems using the WinRM protocol for communication. Ansible provides specific modules for managing Windows, such as win\_feature, win\_service, and win\_shell. Configuration may require setting up WinRM on the Windows machines to allow Ansible to connect and manage them.

—------------------------------------------------------------------------------------------------------------------------