

Ansible

Ansible is a powerful, open-source automation tool designed to streamline IT tasks such as configuration management, application deployment, and orchestration. Its core strengths are its simplicity, scalability, and ease of use. Automation tasks in Ansible are defined using human-readable YAML files called Playbooks.

Key Features of Ansible

Agentless Operation

Ansible does not require additional software or agents on managed systems. It connects using SSH for Linux/Unix or WinRM for Windows.

Push-based Model

The control node sends commands directly to the managed nodes, ensuring efficient task execution.

Platform Independence

Ansible works seamlessly across diverse environments—cloud, on-premises, or hybrid.

Security by Design

Uses OpenSSH for secure communication and avoids storing sensitive information by default.

Benefits of Using Ansible

- Automates repetitive IT tasks, saving time and effort.
 - Reduces human error in configuration and system setup.
 - Supports multiple operating systems and environments.
 - Simple to learn, making it accessible even for teams new to automation.
-

Installation Steps

Ubuntu

```
sudo apt update
```

```
sudo apt install ansible
```

CentOS/RHEL

```
sudo yum install epel-release
```

```
sudo yum install ansible -y
```

macOS (using Homebrew)

```
brew install ansible
```

Ansible Installation steps

Step 1: Launch EC2 Instances

1. **Log in to AWS Management Console** and go to the **EC2 Dashboard**.
2. **Launch four EC2 instances:**
 - **1 Master Server** (Ansible control node).
 - **2 Web Servers**.
 - **1 Database Server**.
3. **Configure instance details:**

- **Amazon Machine Image (AMI):** Select **Ubuntu 20.04 LTS** or a newer version.
 - **Instance Type:** Use t2.micro (free tier eligible) or a larger type if needed.
 - **VPC and Subnet:** Ensure all instances are in the same **VPC and subnet**.
 - **Key Pair:** Create or select an existing key pair for SSH access. Download the .pem file if creating a new one.
4. **Configure security group:**
- Open port **22 (SSH)** for your local machine and Master server's IP for internal communication.
5. **Launch Instances** and note their **Private and Public IPs**.
-

Step 2: SSH into the Master Server

Connect to the Master server using the downloaded .pem file:

```
ssh -i <key.pem> ubuntu@<master_server_public_ip>
```

Update the system and install dependencies:

```
sudo apt update
```

```
sudo apt install python3 python3-pip -y
```

Install Ansible:

```
sudo apt install ansible -y
```

Step 3: Configure SSH Key Authentication

Generate SSH key pair on the Master server:

```
ssh-keygen
```

- Save the key in the default location: ~/.ssh/id_rsa.
- Leave the passphrase empty.

Copy the public key to all managed nodes (Web servers and Database server):

```
ssh-copy-id -i ~/.ssh/id_rsa.pub ubuntu@<target_private_ip>
```

2. Replace <target_private_ip> with the private IP of each managed node.

3. **Manually Add Public Key (if ssh-copy-id is unavailable):**

On the Master server:

```
cat ~/.ssh/id_rsa.pub
```

SSH into the managed node:

```
ssh -i <key.pem> ubuntu@<target_private_ip>
```

Open or create the ~/.ssh/authorized_keys file:

```
nano ~/.ssh/authorized_keys
```

- Paste the public key into this file, save, and exit.

Set permissions on the managed node:

```
chmod 700 ~/.ssh
```

```
chmod 600 ~/.ssh/authorized_keys
```

Test passwordless SSH login:

```
ssh ubuntu@<target_private_ip>
```

Step 4: Create the Ansible Inventory

Create an inventory file on the Master server:

```
nano ~/inventory
```

Add private IPs of managed nodes under appropriate groups:

```
ini
```

```
[web_servers]
```

```
192.168.1.101
```

```
192.168.1.102
```

```
[db_server]
```

```
192.168.1.103
```

Step 5: Test Ansible Connectivity

Ping all nodes to verify connectivity:

```
ansible all -i ~/inventory -m ping
```

Expected Output:

```
plaintext
```

```
192.168.1.101 | SUCCESS => {
```

```
  "changed": false,
```

```
  "ping": "pong"
```

```
}
```

```
192.168.1.102 | SUCCESS => {
```

```
  "changed": false,
```

```
  "ping": "pong"
```

```
}
```

```
192.168.1.103 | SUCCESS => {
```

```
"changed": false,  
"ping": "pong"  
}
```

Writing and Running Playbooks

Example: Install and Start Nginx

yaml

- name: Install and configure Nginx

hosts: web_servers

become: yes

tasks:

- name: Install Nginx

apt:

name: nginx

state: present

- name: Start Nginx

service:

name: nginx

state: started

Ansible Playbooks Example

1. Simple Playbook to Install a Package

yaml

- name: Install Nginx web server

hosts: webservers

become: yes

tasks:

- name: Install nginx package

apt:

name: nginx

state: present

In this example, the playbook installs the Nginx web server on a group of hosts called webservers.

2. Playbook to Start and Enable a Service

yaml

- name: Ensure Nginx is running

hosts: webservers

become: yes

tasks:

- name: Start Nginx service

service:

name: nginx

state: started

enabled: yes

This playbook ensures the Nginx service is started and enabled to start on boot.

3. Playbook to Configure a File

yaml

- name: Configure Nginx

hosts: webservers

become: yes

tasks:

- name: Copy the Nginx configuration file

template:

src: /path/to/nginx.conf.j2

dest: /etc/nginx/nginx.conf

notify:

- restart nginx

handlers:

- name: restart nginx

service:

name: nginx

state: restarted

This playbook copies a customized Nginx configuration file using the template module and restarts the Nginx service if the configuration changes.

4. Playbook to Install Multiple Packages

yaml

- name: Install common utilities

hosts: all

become: yes

tasks:

- name: Install packages

apt:

name:

- curl

- vim

- git

state: present

This playbook installs multiple packages (curl, vim, and git) on all specified hosts.

5. Playbook to Install Apache Web Server and Configure Virtual Host

yaml

- name: Install Apache Web Server and Configure Virtual Host

hosts: webservers

become: yes

tasks:

- name: Install Apache

apt:

name: apache2

state: present

- name: Copy virtual host configuration file

template:

src: /path/to/vhost.conf.j2

dest: /etc/apache2/sites-available/vhost.conf

notify:

- restart apache

- name: Enable the virtual host

command: a2ensite vhost.conf

notify:

- restart apache

handlers:

- name: restart apache

service:

name: apache2

state: restarted

This playbook installs Apache, configures a virtual host using a template, and enables the virtual host configuration.

6. Playbook to Create Users and Set Permissions

yaml

- name: Create Users and Set Permissions

hosts: all

become: yes

tasks:

- name: Create a new user

user:

name: "{{ item.username }}"

state: present

shell: /bin/

loop:

- { username: "user1" }

- { username: "user2" }

- name: Create directory and set permissions

file:

```
path: "/home/{{ item.username }}/shared"
```

```
state: directory
```

```
mode: '0775'
```

```
owner: "{{ item.username }}"
```

```
group: "{{ item.username }}"
```

```
loop:
```

```
- { username: "user1" }
```

```
- { username: "user2" }
```

This playbook creates two users (user1 and user2), and it also sets up a shared directory for each user with appropriate permissions.

7. Playbook to Backup a Directory

```
yaml
```

```
---
```

```
- name: Backup Directory
```

```
hosts: all
```

```
become: yes
```

```
tasks:
```

```
- name: Create backup directory
```

file:

path: "/backup"

state: directory

- name: Copy files to backup directory

copy:

src: "/path/to/important_data"

dest: "/backup/"

remote_src: yes

mode: '0644'

- name: Verify backup

stat:

path: "/backup/important_data"

register: backup_status

- name: Display backup status

debug:

msg: "Backup created successfully"

when: backup_status.stat.exists

This playbook creates a backup of a directory (/path/to/important_data) to a /backup folder and verifies the backup.

8. Playbook to Set Up a MySQL Database

yaml

- name: Set up MySQL Database and User

hosts: dbservers

become: yes

vars:

db_name: mydatabase

db_user: dbuser

db_password: "secret"

tasks:

- name: Install MySQL server

apt:

name: mysql-server

state: present

- name: Start MySQL service

service:

name: mysql

state: started

enabled: yes

- name: Create a database

mysql_db:

name: "{{ db_name }}"

state: present

- name: Create a user and grant privileges

mysql_user:

name: "{{ db_user }}"

password: "{{ db_password }}"

priv: "{{ db_name }}.*:ALL"

state: present

This playbook installs MySQL, starts the MySQL service, creates a database (mydatabase), and creates a user with privileges on the database.

9. Playbook to Update System Packages

yaml

- name: Update System Packages

 - hosts: all

 - become: yes

 - tasks:

 - name: Update apt cache

 - apt:

 - update_cache: yes

 - name: Upgrade all packages

 - apt:

 - upgrade: dist

This playbook updates the system's package cache and performs a full system upgrade.

10. Playbook to Manage a Firewall

yaml

- name: Manage Firewall Rules

 - hosts: all

 - become: yes

 - tasks:

- name: Ensure ufw is installed

apt:

name: ufw

state: present

- name: Allow SSH through firewall

ufw:

rule: allow

name: 'OpenSSH'

- name: Deny all incoming traffic

ufw:

default: deny

direction: incoming

- name: Allow outgoing traffic

ufw:

default: allow

direction: outgoing

- name: Enable the firewall

ufw:

state: enabled

This playbook installs the ufw firewall, configures firewall rules to allow SSH and deny incoming traffic by default while allowing outgoing traffic.

11. Playbook to Set Up a Node.js Application

yaml

- name: Set up Node.js Application

hosts: webservers

become: yes

tasks:

- name: Install Node.js

apt:

name: nodejs

state: present

- name: Install npm

apt:

name: npm

state: present

- name: Deploy application files

copy:

src: /path/to/app/

dest: /var/www/myapp/

mode: '0755'

- name: Install dependencies

npm:

path: /var/www/myapp/

state: present

- name: Start the application

shell: nohup node /var/www/myapp/app.js &

This playbook installs Node.js and npm, deploys the application files, installs dependencies, and starts the Node.js application.

Advanced Features in Playbooks:

- **Conditionals:** Run tasks based on specific conditions, using when.
- **Loops:** Repeat tasks using loop.
- **Includes and Imports:** Organize playbooks using include or import to reuse common tasks.

- **Templates:** Use Jinja2 templates to dynamically generate configuration files.

12. Playbook to Deploy a Web Application (Using Git)

yaml

- name: Deploy a Web Application from Git

hosts: webservers

become: yes

vars:

app_repo: "https://github.com/username/repository.git"

app_dest: "/var/www/myapp"

tasks:

- name: Ensure git is installed

apt:

name: git

state: present

- name: Clone the web application from GitHub

git:

```
repo: "{{ app_repo }}"
```

```
dest: "{{ app_dest }}"
```

```
clone: yes
```

```
update: yes
```

```
- name: Install dependencies using npm
```

```
npm:
```

```
path: "{{ app_dest }}"
```

```
state: present
```

```
- name: Start the web application
```

```
shell: nohup node {{ app_dest }}/app.js &
```

This playbook clones a web application from a Git repository, installs dependencies using npm, and starts the application.

13. Playbook to Install Docker and Run a Container

yaml

```
- name: Install Docker and Run a Container
```

hosts: all

become: yes

tasks:

- name: Install Docker

apt:

name: docker.io

state: present

- name: Start Docker service

service:

name: docker

state: started

enabled: yes

- name: Pull the Docker image

docker_image:

name: nginx

source: pull

- name: Run the Docker container

docker_container:

name: nginx_container

image: nginx

state: started

published_ports:

- "8080:80"

This playbook installs Docker, pulls an Nginx image, and runs it in a container, exposing port 8080 on the host machine.

14. Playbook to Set Up a Python Virtual Environment

yaml

- name: Set up Python Virtual Environment

hosts: all

become: yes

tasks:

- name: Install python3 and pip

apt:

name:

- python3

- python3-pip

state: present

- name: Install virtualenv package

pip:

name: virtualenv

state: present

- name: Create a virtual environment

command:

cmd: python3 -m venv /home/{{ ansible_user }}/myenv

creates: /home/{{ ansible_user }}/myenv

- name: Install dependencies in the virtual environment

pip:

requirements: /path/to/requirements.txt

virtualenv: /home/{{ ansible_user }}/myenv

This playbook installs Python, pip, and the virtualenv package, creates a virtual environment, and installs dependencies from a requirements.txt file.

ansible-playbook -i inventory playbook.yml

Advanced Ansible Features

Using Variables

Variables enhance playbook flexibility.

yaml

vars:

package_name: nginx

tasks:

- name: Install a package

apt:

name: "{{ package_name }}"

state: present

Templates with Jinja2

Use templates for dynamic configurations.

yaml

- name: Apply Nginx configuration

template:

src: nginx.conf.j2

dest: /etc/nginx/nginx.conf

Handlers for Notifications

Trigger actions like service restarts when configurations change.

Roles for Reusability

Organize tasks into reusable roles using:

```
ansible-galaxy init <role_name>
```

Summary

Ansible simplifies IT automation with its agentless architecture, human-readable Playbooks, and robust features for managing tasks across diverse environments. Whether you're deploying applications, configuring servers, or orchestrating complex workflows, Ansible is a versatile solution for all.

Basic Structure of a Playbook

A typical Ansible playbook contains:

- **Hosts:** Specifies the target systems (can be a group or individual machine).
 - **Tasks:** Defines the actions to be performed, such as installing a package, copying files, or restarting services.
 - **Variables:** Used to manage configurations dynamically.
 - **Handlers:** Special tasks that are triggered only when notified by other tasks (e.g., restart a service if a configuration file is changed).
-

15. Playbook to Secure a Server (SSH, Firewall, and Users)

```
yaml
```

```
---
```

- name: Secure the Server (SSH, Firewall, and Users)

hosts: all

become: yes

tasks:

- name: Disable root login over SSH

lineinfile:

path: /etc/ssh/sshd_config

regexp: '^#?PermitRootLogin'

line: 'PermitRootLogin no'

notify:

- restart ssh

- name: Configure UFW to allow SSH

ufw:

rule: allow

name: 'OpenSSH'

- name: Add a user for remote login

user:

name: "remoteuser"

state: present

shell: /bin/

- name: Disable password authentication in SSH

lineinfile:

path: /etc/ssh/sshd_config

regexp: '^#?PasswordAuthentication'

line: 'PasswordAuthentication no'

notify:

- restart ssh

handlers:

- name: restart ssh

service:

name: ssh

state: restarted

This playbook improves server security by:

- Disabling root login via SSH.
 - Setting up the UFW firewall to allow SSH traffic.
 - Adding a user for secure login.
 - Disabling password-based SSH authentication (requires SSH key authentication).
-

16. Playbook to Set Up Redis

yaml

- name: Install and Configure Redis

 - hosts: dbservers

 - become: yes

 - tasks:

 - name: Install Redis

 - apt:

 - name: redis-server

 - state: present

 - name: Start Redis service

 - service:

 - name: redis-server

 - state: started

 - enabled: yes

 - name: Configure Redis to bind to all IP addresses

 - lineinfile:

 - path: /etc/redis/redis.conf

regexp: '^#?bind 127.0.0.1'

line: 'bind 0.0.0.0'

notify:

- restart redis

handlers:

- name: restart redis

service:

name: redis-server

state: restarted

This playbook installs Redis, starts the service, and configures it to listen on all IP addresses by modifying the configuration file.

17. Playbook to Create a Backup of MySQL Database

yaml

- name: Backup MySQL Database

hosts: dbservers

become: yes

tasks:

- name: Backup MySQL database to a file

command:

cmd: mysqldump -u root -p{{ mysql_root_password }} {{ mysql_db }} >
/backup/{{ mysql_db }}_backup.sql

args:

chdir: /tmp

environment:

MYSQL_PWD: "{{ mysql_root_password }}"

- name: Verify backup file

stat:

path: "/backup/{{ mysql_db }}_backup.sql"

register: backup_status

- name: Display backup status

debug:

msg: "Backup created successfully"

when: backup_status.stat.exists

This playbook creates a backup of a MySQL database using mysqldump and stores it in a specified backup directory. It verifies the existence of the backup file and prints a message.

18. Playbook to Install and Configure Postfix Mail Server

yaml

- name: Install and Configure Postfix Mail Server

hosts: mailservers

become: yes

tasks:

- name: Install Postfix

apt:

name: postfix

state: present

- name: Configure Postfix to relay emails

lineinfile:

path: /etc/postfix/main.cf

regexp: '^#?relayhost'

line: 'relayhost = [smtp.example.com]:587'

notify:

- restart postfix

handlers:

- name: restart postfix

service:

name: postfix

state: restarted

This playbook installs the Postfix mail server, configures it to relay emails through a remote SMTP server, and restarts the service.

19. Playbook to Perform System Cleanup

yaml

- name: System Cleanup

hosts: all

become: yes

tasks:

- name: Clean the apt cache

apt:

autoclean: yes

- name: Remove unused packages

apt:

autoremove: yes

- name: Clean old log files

file:

path: "/var/log/{{ item }}"

state: absent

loop:

- auth.log

- syslog

- dpkg.log

This playbook cleans up the system by:

- Autocleaning the APT cache.
- Removing unused packages.
- Deleting old log files.

20. Playbook to Set Up a Jenkins Server

yaml

- name: Set up Jenkins Server

hosts: jenkins_servers

become: yes

tasks:

- name: Install Java

apt:

name: openjdk-11-jdk

state: present

- name: Add Jenkins repository

apt_repository:

repo: "deb http://pkg.jenkins.io/debian/ stable main"

state: present

- name: Install Jenkins

apt:

name: jenkins

state: present

- name: Start Jenkins service

service:

name: jenkins

state: started

enabled: yes

This playbook installs Java, adds the Jenkins repository, installs Jenkins, and starts the Jenkins service.

Advanced Playbook Features:

- **Conditionals with when:** Used to run tasks based on conditions.
 - **Loops:** Helps repeat a task multiple times with different variables.
 - **Templates:** Use Jinja2 templating for dynamic file generation.
 - **Error Handling:** Implement block, rescue, and always to manage error scenarios.
-

21. Playbook to Configure NTP (Network Time Protocol)

yaml

- name: Configure NTP on Servers

hosts: all

become: yes

tasks:

- name: Install ntp package

apt:

name: ntp

state: present

- name: Ensure NTP service is started

service:

name: ntp

state: started

enabled: yes

- name: Configure NTP server

lineinfile:

path: /etc/ntp.conf

regexp: '^server'

line: 'server time.nist.gov iburst'

notify:

- restart ntp

handlers:

- name: restart ntp

service:

name: ntp

state: restarted

This playbook installs the ntp package, configures the NTP server to synchronize with an NTP source (e.g., time.nist.gov), and ensures the NTP service is running.

22. Playbook to Install and Configure a Web Server (Apache)

yaml

- name: Install and Configure Apache Web Server

hosts: webservers

become: yes

tasks:

- name: Install Apache

- apt:

- name: apache2

- state: present

- name: Ensure Apache is running

- service:

- name: apache2

- state: started

- enabled: yes

- name: Copy the website files

copy:

src: /local/path/to/website/

dest: /var/www/html/

owner: www-data

group: www-data

mode: '0755'

- name: Configure Apache virtual host

template:

src: /local/path/to/vhost.conf.j2

dest: /etc/apache2/sites-available/000-default.conf

notify:

- restart apache

handlers:

- name: restart apache

service:

name: apache2

state: restarted

This playbook installs Apache, starts the service, copies website files to the web server's document root, and configures a virtual host using a template file.

23. Playbook to Install and Configure MariaDB

yaml

- name: Install and Configure MariaDB

 - hosts: dbservers

 - become: yes

 - tasks:

 - name: Install MariaDB

 - apt:

 - name: mariadb-server

 - state: present

 - name: Start MariaDB service

 - service:

 - name: mariadb

 - state: started

 - enabled: yes

 - name: Secure MariaDB installation

```
mysql_secure_installation:

  login_user: root

  login_password: "{{ mysql_root_password }}"

  root_password: "{{ mysql_root_password }}"

  set_root_password: yes

  remove_anonymous_users: yes

  disallow_root_login_remotely: yes

  remove_test_db: yes

  state: present
```

This playbook installs MariaDB, ensures the service is running, and secures the installation by setting the root password and disabling remote root login.

24. Playbook to Install and Configure Redis Cluster

```
yaml
---
- name: Install and Configure Redis Cluster

  hosts: redis_nodes

  become: yes

  tasks:
```

- name: Install Redis

apt:

name: redis-server

state: present

- name: Configure Redis for clustering

lineinfile:

path: /etc/redis/redis.conf

regexp: '^#?cluster-enabled'

line: 'cluster-enabled yes'

notify:

- restart redis

- name: Open Redis cluster ports

ufw:

rule: allow

name: 'Redis Cluster'

port: '6379:6380'

handlers:

- name: restart redis

service:

name: redis-server

state: restarted

This playbook installs Redis, configures it for clustering, and opens the required ports on the firewall.

25. Playbook to Install and Configure a Filebeat Agent (for Log Shipping)

yaml

- name: Install and Configure Filebeat

hosts: all

become: yes

tasks:

- name: Install Filebeat package

apt:

name: filebeat

state: present

- name: Configure Filebeat to ship logs

template:

src: /local/path/to/filebeat.yml.j2

dest: /etc/filebeat/filebeat.yml

notify:

- restart filebeat

- name: Enable Filebeat service

service:

name: filebeat

state: started

enabled: yes

handlers:

- name: restart filebeat

service:

name: filebeat

state: restarted

This playbook installs Filebeat, configures it using a template file to ship logs to a centralized log management system, and ensures the Filebeat service is started.

26. Playbook to Install and Configure Jenkins Agent

yaml

- name: Install and Configure Jenkins Agent

hosts: jenkins_agents

become: yes

tasks:

- name: Install Java

apt:

name: openjdk-11-jdk

state: present

- name: Download Jenkins agent JAR

get_url:

url: "https://<jenkins-url>/jnlpJars/agent.jar"

dest: /opt/jenkins/agent.jar

- name: Run Jenkins agent

shell: java -jar /opt/jenkins/agent.jar -jnlpUrl <jenkins-url>/computer/{{
inventory_hostname }}/slave-agent.jnlp

async: 60

poll: 0

This playbook installs Java, downloads the Jenkins agent JAR, and runs the agent as a background process to connect to the Jenkins master.

27. Playbook to Set Up a Vault Server (HashiCorp Vault)

yaml

- name: Install and Configure Vault Server

hosts: vault_servers

become: yes

tasks:

- name: Install Vault

apt:

name: vault

state: present

- name: Enable Vault service

systemd:

name: vault

enabled: yes

state: started

- name: Initialize Vault (first-time setup)

- command: vault operator init -key-shares=1 -key-threshold=1

- register: vault_init

- when: vault_init.rc != 0

- name: Unseal Vault

- command: vault operator unseal "{{ vault_init.stdout_lines[0] }}"

- when: vault_init.rc == 0

This playbook installs HashiCorp Vault, starts the service, and initializes and unseals Vault on the first run.

28. Playbook to Install and Configure Kubernetes Worker Nodes

yaml

- name: Configure Kubernetes Worker Nodes

- hosts: kubernetes_workers

- become: yes

- tasks:

- name: Install kubelet, kubeadm, and kubectl

apt:

name:

- kubelet
- kubeadm
- kubectl

state: present

- name: Join the Kubernetes cluster

command: kubeadm join {{ master_ip }}:6443 --token {{ token }}
--discovery-token-ca-cert-hash sha256:{{ ca_hash }}

when: inventory_hostname != "master"

- name: Ensure kubelet is running

service:

name: kubelet

state: started

enabled: yes

This playbook installs the necessary Kubernetes packages on worker nodes, joins the node to the Kubernetes cluster, and ensures the kubelet service is running.

29. Playbook to Set Up Docker Swarm Cluster

yaml

- name: Configure Docker Swarm Cluster

 - hosts: swarm_masters

 - become: yes

 - tasks:

 - name: Install Docker

 - apt:

 - name: docker.io

 - state: present

 - name: Initialize Docker Swarm on Master Node

 - command: docker swarm init

 - when: inventory_hostname == "master"

 - name: Join Swarm cluster as worker node

 - command: docker swarm join --token {{ swarm_token }} {{ master_ip }}:2377

 - when: inventory_hostname != "master"

This playbook configures a Docker Swarm cluster with master and worker nodes by initializing the swarm on the master and joining worker nodes.

30. Playbook to Install and Configure Prometheus Server

yaml

- name: Install and Configure Prometheus Server

hosts: prometheus_servers

become: yes

tasks:

- name: Install Prometheus

apt:

name: prometheus

state: present

- name: Configure Prometheus server

template:

src: /local/path/to/prometheus.yml.j2

dest: /etc/prometheus/prometheus.yml

notify:

- restart prometheus

- name: Ensure Prometheus service is started

service:

name: prometheus

state: started

enabled: yes

handlers:

- name: restart prometheus

service:

name: prometheus

state: restarted

This playbook installs Prometheus, configures it with a template file, and ensures the Prometheus service is started and enabled.

31. Playbook to Set Up an Nginx Reverse Proxy

yaml

- name: Set Up Nginx Reverse Proxy

hosts: webservers

become: yes

tasks:

- name: Install Nginx

apt:

name: nginx

state: present

- name: Configure Nginx as reverse proxy

template:

src: /local/path/to/nginx_reverse_proxy.conf.j2

dest: /etc/nginx/sites-available/default

notify:

- restart nginx

- name: Ensure Nginx is started and enabled

service:

name: nginx

state: started

enabled: yes

handlers:

- name: restart nginx

service:

name: nginx

state: restarted

This playbook installs Nginx, configures it as a reverse proxy using a template file, and ensures that the service is running and enabled on the system.

32. Playbook to Configure System Security (Firewall)

yaml

- name: Configure Firewall (UFW)

hosts: all

become: yes

tasks:

- name: Install UFW (Uncomplicated Firewall)

apt:

name: ufw

state: present

- name: Allow SSH traffic

ufw:

rule: allow

name: OpenSSH

- name: Allow HTTP traffic

- ufw:

- rule: allow

- name: 'Apache Full'

- name: Enable UFW

- ufw:

- state: enabled

- default: deny

This playbook installs and configures the Uncomplicated Firewall (UFW) to allow SSH and HTTP traffic while denying other inbound connections.

33. Playbook to Install and Configure Prometheus Node Exporter

yaml

- name: Install and Configure Prometheus Node Exporter

- hosts: all

- become: yes

- tasks:

- name: Download Prometheus Node Exporter

get_url:

url:

https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_exporter-1.3.1.linux-amd64.tar.gz

dest: /tmp/node_exporter.tar.gz

- name: Extract Node Exporter

unarchive:

src: /tmp/node_exporter.tar.gz

dest: /opt/

remote_src: yes

- name: Create systemd service for Node Exporter

copy:

content: |

[Unit]

Description=Prometheus Node Exporter

After=network.target

[Service]

User=nobody

ExecStart=/opt/node_exporter-1.3.1.linux-amd64/node_exporter

[Install]

WantedBy=multi-user.target

dest: /etc/systemd/system/node_exporter.service

notify:

- reload systemd

- name: Start Node Exporter

service:

name: node_exporter

state: started

enabled: yes

handlers:

- name: reload systemd

systemd:

daemon_reload: yes

This playbook installs the Prometheus Node Exporter, creates a systemd service for it, and ensures the service is started and enabled.

34. Playbook to Set Up a MySQL Database Backup

yaml

- name: Set Up MySQL Database Backup

hosts: dbservers

become: yes

tasks:

- name: Install MySQL client and cron

apt:

name:

- mysql-client

- cron

state: present

- name: Create backup directory

file:

path: /var/backups/mysql

state: directory

mode: '0755'

- name: Create cron job for backup

cron:

name: "Daily MySQL Backup"

minute: "0"

hour: "2"

job: "/usr/bin/mysqldump -u root -p{{ mysql_root_password }}
--all-databases > /var/backups/mysql/backup_\$(date +%F).sql"

state: present

This playbook installs the MySQL client and cron service, creates a backup directory, and sets up a cron job to back up the MySQL databases every day at 2 AM.

35. Playbook to Set Up a Jenkins Master Node

yaml

- name: Install and Configure Jenkins Master Node

hosts: jenkins_masters

become: yes

tasks:

- name: Install Java OpenJDK 11

apt:

name: openjdk-11-jdk

state: present

- name: Add Jenkins repository key

apt_key:

url: https://pkg.jenkins.io/jenkins.io.key

- name: Add Jenkins repository

apt_repository:

repo: deb http://pkg.jenkins.io/debian/ stable main

- name: Install Jenkins

apt:

name: jenkins

state: present

- name: Start Jenkins service

service:

name: jenkins

state: started

enabled: yes

This playbook installs Java (required by Jenkins), adds the Jenkins repository, and installs Jenkins on the master node, ensuring the service is started and enabled.

36. Playbook to Install and Configure Elasticsearch

yaml

- name: Install and Configure Elasticsearch

 - hosts: elasticsearch_servers

 - become: yes

 - tasks:

 - name: Install Java (required for Elasticsearch)

 - apt:

 - name: openjdk-11-jdk

 - state: present

 - name: Add Elasticsearch GPG key

 - apt_key:

 - url: <https://artifacts.elastic.co/GPG-KEY-elasticsearch>

 - name: Add Elasticsearch APT repository

apt_repository:

repo: "deb https://artifacts.elastic.co/packages/7.x/apt stable main"

- name: Install Elasticsearch

apt:

name: elasticsearch

state: present

- name: Start Elasticsearch service

service:

name: elasticsearch

state: started

enabled: yes

This playbook installs Elasticsearch on a server, ensures Java is present, and configures Elasticsearch to start automatically.

37. Playbook to Set Up Docker Registry

yaml

- name: Set Up Docker Registry

hosts: registry_servers

become: yes

tasks:

- name: Install Docker

apt:

name: docker.io

state: present

- name: Create Docker Registry directory

file:

path: /var/lib/registry

state: directory

- name: Run Docker Registry container

docker_container:

name: registry

image: registry:2

state: started

ports:

- "5000:5000"

volumes:

- /var/lib/registry:/var/lib/registry

This playbook installs Docker, creates a directory for storing Docker images, and runs the Docker Registry container.

38. Playbook to Install and Configure GitLab CI/CD Runner

yaml

- name: Install and Configure GitLab CI/CD Runner

 - hosts: ci_cd_servers

 - become: yes

 - tasks:

 - name: Install GitLab Runner

 - apt:

 - name: gitlab-runner

 - state: present

 - name: Register GitLab Runner

 - command: gitlab-runner register --url https://gitlab.com/ --registration-token {{ gitlab_runner_token }} --executor shell --description "{{ inventory_hostname }}"

 - when: ansible_facts['distribution'] == 'Ubuntu'

- name: Start GitLab Runner

- service:

- name: gitlab-runner

- state: started

- enabled: yes

This playbook installs the GitLab CI/CD runner, registers it with the GitLab instance using a registration token, and ensures the service is started.

39. Playbook to Set Up a Redis Sentinel Cluster

yaml

- name: Set Up Redis Sentinel Cluster

- hosts: redis_sentinels

- become: yes

- tasks:

- name: Install Redis

- apt:

- name: redis-server

- state: present

- name: Configure Redis Sentinel

template:

src: /local/path/to/sentinel.conf.j2

dest: /etc/redis/sentinel.conf

notify:

- restart redis sentinel

- name: Start Redis Sentinel service

service:

name: redis-sentinel

state: started

enabled: yes

handlers:

- name: restart redis sentinel

service:

name: redis-sentinel

state: restarted

This playbook installs Redis and sets up Redis Sentinel to provide high availability and failover for a Redis cluster.

40. Playbook to Set Up a Kubernetes Dashboard

yaml

- name: Install and Configure Kubernetes Dashboard

hosts: master_nodes

become: yes

tasks:

- name: Deploy Kubernetes Dashboard

kubernetes:

name: kubernetes-dashboard

state: present

api_version: apps/v1

kind: Deployment

namespace: kube-system

definition:

apiVersion: apps/v1

kind: Deployment

metadata:

name: kubernetes-dashboard

namespace: kube-system

```
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: kubernetes-dashboard
  template:
    metadata:
      labels:
        k8s-app: kubernetes-dashboard
    spec:
      containers:
        - name: kubernetes-dashboard
          image: kubernetesui/dashboard:v2.0.0
          ports:
            - containerPort: 9090
```

This playbook deploys the Kubernetes Dashboard on the master nodes of a Kubernetes cluster.

41. Playbook to Install and Configure Docker

yaml

```
- name: Install and Configure Docker
```

hosts: all

become: yes

tasks:

- name: Update apt repository

apt:

update_cache: yes

- name: Install dependencies for Docker

apt:

name:

- apt-transport-https
- ca-certificates
- curl
- software-properties-common

state: present

- name: Add Docker GPG key

apt_key:

url: <https://download.docker.com/linux/ubuntu/gpg>

- name: Add Docker repository

apt_repository:

repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu
\$(lsb_release -cs) stable

- name: Install Docker

apt:

name: docker-ce

state: present

- name: Start and enable Docker service

service:

name: docker

state: started

enabled: yes

This playbook installs Docker on an Ubuntu machine, adds the official Docker repository, and ensures the service is started and enabled.

42. Playbook to Configure NTP (Network Time Protocol)

yaml

- name: Install and Configure NTP

hosts: all

become: yes

tasks:

- name: Install NTP service

apt:

name: ntp

state: present

- name: Start and enable NTP service

service:

name: ntp

state: started

enabled: yes

- name: Configure NTP servers

lineinfile:

path: /etc/ntp.conf

regexp: '^server'

line: 'server time.google.com iburst'

notify:

- restart ntp

handlers:

- name: restart ntp

service:

- name: ntp

- state: restarted

This playbook installs the NTP service, configures a specific NTP server, and ensures the service is running and enabled.

43. Playbook to Install and Configure MySQL Server

yaml

- name: Install and Configure MySQL Server

- hosts: dbservers

- become: yes

- tasks:

- name: Install MySQL server

- apt:

- name: mysql-server

state: present

- name: Ensure MySQL service is started

service:

name: mysql

state: started

enabled: yes

- name: Create a database

mysql_db:

name: example_db

state: present

- name: Create a MySQL user

mysql_user:

name: example_user

password: "{{ mysql_user_password }}"

state: present

priv: "example_db.*:ALL"

This playbook installs MySQL, creates a new database and user, and grants the necessary privileges.

44. Playbook to Install and Configure Apache Kafka

yaml

- name: Install and Configure Apache Kafka

hosts: kafka_nodes

become: yes

tasks:

- name: Install Java (required for Kafka)

apt:

name: openjdk-11-jdk

state: present

- name: Download Kafka

get_url:

url: https://downloads.apache.org/kafka/2.8.0/kafka_2.13-2.8.0.tgz

dest: /tmp/kafka.tgz

- name: Extract Kafka

unarchive:

src: /tmp/kafka.tgz

dest: /opt/

remote_src: yes

- name: Create a Kafka systemd service

copy:

content: |

[Unit]

Description=Apache Kafka

After=network.target

[Service]

User=nobody

ExecStart=/opt/kafka_2.13-2.8.0/bin/kafka-server-start.sh
/opt/kafka_2.13-2.8.0/config/server.properties

[Install]

WantedBy=multi-user.target

dest: /etc/systemd/system/kafka.service

notify:

- reload systemd

- name: Start Kafka service

service:

name: kafka

state: started

enabled: yes

handlers:

- name: reload systemd

systemd:

daemon_reload: yes

This playbook installs Apache Kafka, configures a systemd service for it, and ensures the service is running.

45. Playbook to Configure AWS EC2 Instances Using Ansible

yaml

- name: Configure AWS EC2 Instances

hosts: localhost

gather_facts: no

tasks:

- name: Launch an EC2 instance

ec2_instance:

```
key_name: "{{ aws_key_name }}"
id: "{{ aws_instance_id }}"
instance_type: t2.micro
region: "{{ aws_region }}"
image_id: ami-0c55b159cbfafa1f0
wait: yes
count: 1
security_group: "{{ aws_security_group }}"
subnet_id: "{{ aws_subnet_id }}"
instance_tags:
    Name: "MyEC2Instance"
assign_public_ip: yes
register: ec2_instances
```

- name: Output instance details

```
debug:
```

```
var: ec2_instances.instances
```

This playbook launches an EC2 instance in AWS, waits for it to be ready, and outputs the instance details.

46. Playbook to Install and Configure Elasticsearch and Kibana

yaml

- name: Install and Configure Elasticsearch and Kibana

 - hosts: all

 - become: yes

 - tasks:

 - name: Install Elasticsearch

 - apt:

 - name: elasticsearch

 - state: present

 - name: Start Elasticsearch service

 - service:

 - name: elasticsearch

 - state: started

 - enabled: yes

 - name: Install Kibana

 - apt:

 - name: kibana

state: present

- name: Start Kibana service

service:

name: kibana

state: started

enabled: yes

This playbook installs and starts both Elasticsearch and Kibana services on the server.

47. Playbook to Install and Configure Nginx as a Load Balancer

yaml

- name: Install and Configure Nginx as a Load Balancer

hosts: load_balancer

become: yes

tasks:

- name: Install Nginx

apt:

name: nginx

state: present

- name: Configure Nginx for load balancing

template:

src: /local/path/to/nginx_load_balancer.conf.j2

dest: /etc/nginx/nginx.conf

notify:

- restart nginx

- name: Start Nginx service

service:

name: nginx

state: started

enabled: yes

handlers:

- name: restart nginx

service:

name: nginx

state: restarted

This playbook installs Nginx and configures it as a load balancer using a template file for the configuration.

48. Playbook to Install and Configure Redis

yaml

- name: Install and Configure Redis

 - hosts: all

 - become: yes

 - tasks:

 - name: Install Redis

 - apt:

 - name: redis-server

 - state: present

 - name: Ensure Redis is running

 - service:

 - name: redis-server

 - state: started

 - enabled: yes

- name: Configure Redis to listen on all IPs

lineinfile:

path: /etc/redis/redis.conf

regexp: '^bind 127.0.0.1'

line: 'bind 0.0.0.0'

notify:

- restart redis

handlers:

- name: restart redis

service:

name: redis-server

state: restarted

This playbook installs Redis, configures it to listen on all IPs, and ensures the service is running.

49. Playbook to Install and Configure Prometheus

yaml

- name: Install and Configure Prometheus

 - hosts: all

 - become: yes

 - tasks:

 - name: Install Prometheus

 - apt:

 - name: prometheus

 - state: present

 - name: Ensure Prometheus is started

 - service:

 - name: prometheus

 - state: started

 - enabled: yes

This playbook installs and ensures Prometheus is running.

50. Playbook to Set Up Docker Swarm Cluster

yaml

- name: Set Up Docker Swarm Cluster

 - hosts: swarm_masters

become: yes

tasks:

- name: Initialize Docker Swarm

shell: docker swarm init

when: inventory_hostname == groups['swarm_masters'][0]

- name: Join Docker Swarm cluster

shell: docker swarm join --token {{ swarm_token }} {{ groups['swarm_masters'][0] }}:2377

when: inventory_hostname != groups['swarm_masters'][0]

Ansible Project

LAMP Stack Setup and Web Application Deployment with Ansible

This project demonstrates automating the setup of a LAMP stack (Linux, Apache, MySQL, PHP) and deploying a simple PHP-based web application. Tasks are modularized into separate playbooks for clarity and flexibility.

Project Overview

Goal: Automate the installation of a LAMP stack and deploy a PHP-based web application.

Technologies: Ansible, Apache, MySQL, PHP.

Environment: Ubuntu 20.04 (adaptable for other distributions).

Project Structure

plaintext

lamp_project/

├── ansible/

| ├── inventory.ini # Defines hosts and groups

| ├── lamp_playbook.yml # Main playbook for running all tasks

| ├── vars.yml # Common variables

| └── files/

| └── index.php # Simple PHP application

└── README.md

└── playbooks/

├── install_apache.yml # Playbook for Apache installation

├── install_php.yml # Playbook for PHP installation

├── install_mysql.yml # Playbook for MySQL installation

└── deploy_application.yml # Playbook for application deployment

Configuration Files

Inventory File (inventory.ini)

Defines the target hosts for playbook execution:

ini

```
[web_servers]
```

```
webserver1 ansible_host=192.168.1.10
```

```
webserver2 ansible_host=192.168.1.11
```

```
[db_servers]
```

```
dbserver1 ansible_host=192.168.1.20
```

Variables File (vars.yml)

Stores common configuration variables:

yaml

```
apache_package: apache2
```

```
mysql_root_password: "rootpassword"
```

```
php_packages:
```

```
- php
```

```
- libapache2-mod-php
```

- php-mysql

webapp_name: "myapp"

Playbooks

1. Install Apache (install_apache.yml)

Installs and configures the Apache web server:

yaml

- name: Install Apache web server

hosts: web_servers

become: yes

tasks:

- name: Install Apache

apt:

name: "{{ apache_package }}"

state: present

- name: Ensure Apache is running and enabled

service:

name: apache2

state: started

enabled: yes

2. Install PHP (install_php.yml)

Installs PHP and its required modules:

yaml

- name: Install PHP

hosts: web_servers

become: yes

tasks:

- name: Install PHP packages

apt:

name: "{{ item }}"

state: present

loop: "{{ php_packages }}"

- name: Restart Apache to apply PHP configurations

service:

name: apache2

state: restarted

3. Install MySQL (install_mysql.yml)

Sets up the MySQL database server:

yaml

- name: Install MySQL database server

hosts: db_servers

become: yes

tasks:

- name: Install MySQL server

apt:

name: mysql-server

state: present

- name: Set MySQL root password

mysql_user:

name: root

password: "{{ mysql_root_password }}"

host: "localhost"

state: present

- name: Ensure MySQL is started and enabled

service:

name: mysql

state: started

enabled: yes

4. Deploy PHP Application (deploy_application.yml)

Deploys the PHP application on web servers:

yml

- name: Deploy PHP application

hosts: web_servers

become: yes

tasks:

- name: Copy the PHP application file

copy:

src: files/index.php

dest: /var/www/html/index.php

- name: Set ownership and permissions

file:

path: /var/www/html/index.php

owner: www-data

group: www-data

mode: '0644'

- name: Restart Apache to apply changes

service:

name: apache2

state: restarted

5. Main Playbook (lamp_playbook.yml)

Coordinates all the playbooks:

yaml

- name: Set up LAMP stack and deploy web app

hosts: all

become: yes

tasks:

- name: Include Apache installation

```
import_playbook: playbooks/install_apache.yml
```

- name: Include PHP installation

```
import_playbook: playbooks/install_php.yml
```

- name: Include MySQL installation

```
import_playbook: playbooks/install_mysql.yml
```

- name: Deploy PHP application

```
import_playbook: playbooks/deploy_application.yml
```

Application File (files/index.php)

A simple PHP web application:

```
php
```

```
<?php
```

```
echo "Hello, World! This is your web application running on the LAMP stack!";
```

```
?>
```

How to Run the Project

Prerequisites

- Ensure Ansible 2.9+ is installed on your control node.
- Configure the inventory.ini file with the target server IPs.
- Adjust vars.yml to match your desired settings.

Run the Main Playbook

```
ansible-playbook -i ansible/inventory.ini ansible/lamp_playbook.yml
```

Verifying the Setup

After execution, open a web browser and navigate to the IP address of any web server (e.g., <http://192.168.1.10>).

You should see the following message displayed:

Hello, World! This is your web application running on the LAMP stack!

Summary

This project simplifies the deployment of a LAMP stack using Ansible. Its modular structure allows for:

- Adding virtual hosts for Apache.
- Configuring advanced MySQL settings.

- Deploying more complex web applications.

This Ansible solution is designed for efficiency, scalability, and ease of use, making it ideal for small to medium-sized projects.

Interview Questions And Answers

Basic Questions

What is Ansible, and why is it used?

Ansible is an open-source automation tool used for configuration management, application deployment, and task automation. It uses YAML-based Playbooks and operates without agents, connecting to managed nodes via SSH or WinRM.

What are the main components of Ansible?

- **Control Node:** The machine where Ansible is installed.
- **Managed Nodes:** The servers or devices Ansible manages.
- **Inventory:** A file listing managed nodes.
- **Modules:** Predefined tools Ansible uses to execute tasks.
- **Playbooks:** YAML files describing automation tasks.
- **Plugins:** Extensions for additional functionality.

What makes Ansible different from other automation tools?

- Agentless (uses SSH/WinRM).
- Uses a simple push-based model.

- YAML syntax is easier to read and write compared to other tools like Puppet or Chef.

What is an Ansible Playbook?

A Playbook is a YAML file containing a set of tasks to automate configurations or workflows across managed nodes.

What is the purpose of an Inventory file?

The Inventory file defines the managed nodes and organizes them into groups. It can be static or dynamically generated from sources like AWS or databases.

Intermediate Questions

What is Ansible Vault, and how is it used?

Ansible Vault encrypts sensitive data like passwords or secrets. Commands include:

- `ansible-vault create secrets.yml`
- `ansible-vault encrypt secrets.yml`
- `ansible-vault decrypt secrets.yml`

How do you use Handlers in Ansible?

Handlers are triggered only when notified by a task.

yaml

tasks:

- name: Update config

template:

src: config.j2

dest: /etc/myapp/config

notify: Restart service

handlers:

- name: Restart service

service:

name: myapp

state: restarted

What is a Dynamic Inventory?

Dynamic Inventory retrieves host information from external sources at runtime (e.g., AWS, Azure). It's defined by custom scripts or plugins.

What is `gather_facts`?

`gather_facts` collects system information about managed nodes (e.g., OS, IP). It's enabled by default but can be turned off:

yaml

`gather_facts: no`

How do you loop tasks in Ansible?

Using `with_items`:

yaml

tasks:

- name: Install packages

apt:

name: "{{ item }}"

with_items:

- nginx
- git

How do you manage dependencies in Ansible Roles?

Define dependencies in meta/main.yml:

yaml

dependencies:

- role: common
- role: webserver

Advanced Questions

What is delegate_to, and how is it used?

delegate_to executes a task on a different host.

yaml

tasks:

- name: Run command on another server

command: uptime

delegate_to: 192.168.1.100

How do you ensure idempotency in Ansible?

Ansible modules are designed to make changes only when required. For example, installing a package that's already installed won't perform redundant actions.

What are lookup plugins?

Lookup plugins fetch data from external sources during execution.

yaml

tasks:

- name: Retrieve file content

debug:

msg: "{{ lookup('file', '/path/to/file.txt') }}"

What are the differences between vars, vars_files, and vars_prompt?

- vars: Inline variable declaration.
- vars_files: External variable files.
- vars_prompt: Prompts for user input at runtime.

How do you debug Ansible Playbooks?

- Use -v, -vv, or -vvv for verbose output.
- Add the debug module to print variables.

yaml

tasks:

- debug:

var: my_variable

What is the purpose of block, rescue, and always?

They handle error scenarios gracefully:

yaml

tasks:

- block:

- name: Try something

command: /bin/true

rescue:

- name: Handle failure

debug:

msg: "Something went wrong"

always:

- name: Cleanup

debug:

msg: "Cleanup actions"

Scenario-Based Questions

Scenario: Install a specific version of a package on some hosts and remove it on others.

yaml

tasks:

- name: Install nginx

apt:

name: nginx=1.18.0

state: present

when: "'install_nginx' in group_names"

- name: Remove nginx

apt:

name: nginx

state: absent

when: "'remove_nginx' in group_names"

Scenario: How do you manage different environments like dev, staging, and production?

- Use group-specific variables in group_vars/.
- Separate inventory files: inventory_dev, inventory_staging.

Pass environment variables:

```
ansible-playbook site.yml -e "env=staging"
```

Scenario: Ensure a file exists with specific content and permissions.

yaml

tasks:

- name: Create a file

copy:

dest: /tmp/example.txt

content: "Hello, World!"

owner: root

group: root

mode: '0644'

Troubleshooting and Optimization

What do you do if tasks take too long?

- Increase forks in ansible.cfg.
- Use async and poll for background execution.
- Disable facts gathering if unnecessary: gather_facts: no.

How do you handle SSH authentication issues?

- Use key-based SSH authentication.

- Ensure managed nodes are reachable with `ansible all -m ping`.

How do you test a Playbook without applying changes?

Use the `--check` flag for a dry run and `--diff` to see changes.

Basic to Intermediate Questions

What is the difference between a Task and a Play in Ansible?

- **Task:** A single unit of work in a Playbook. Example: installing a package.
- **Play:** A collection of tasks applied to a group of hosts.

How do you manage Ansible configurations?

Configurations are managed via `ansible.cfg`, which can be located in:

1. The current directory.
2. The user's home directory (`~/.ansible.cfg`).
3. `/etc/ansible/ansible.cfg` (global).

Key settings include inventory location, privilege escalation, retries, and connection parameters.

What is the difference between static and dynamic inventory?

- **Static Inventory:** Manually lists hosts and groups in an inventory file.
- **Dynamic Inventory:** Uses scripts or plugins to fetch real-time host data from external sources like cloud providers.

What are Facts in Ansible, and how do you use them?

Facts are system properties gathered automatically by Ansible (e.g., OS type, IP address). They are accessible as variables. Example:

yaml

tasks:

- debug:

msg: "The system IP is {{ ansible_default_ipv4.address }}"

How does Ansible handle parallelism?

Ansible runs tasks in parallel on multiple hosts. The degree of parallelism is controlled by the forks parameter in ansible.cfg (default is 5).

What is the ansible-pull command?

ansible-pull allows nodes to pull configurations from a centralized Git repository, enabling a pull-based model.

What are some commonly used Ansible Modules?

- **File Management:** copy, template, fetch, lineinfile.
 - **Package Management:** apt, yum, dnf.
 - **System Management:** user, service, cron.
 - **Networking:** uri, firewalld, iptables.
-

Advanced Questions

What is the difference between include_tasks and import_tasks?

- **include_tasks:** Dynamically includes tasks during runtime. Variables can be evaluated dynamically.
- **import_tasks:** Statically includes tasks during Playbook parsing. Variables are evaluated at parse time.

What are Filters in Ansible, and how do you use them?

Filters modify variables or results in a Playbook. Example:

yaml

tasks:

- debug:

msg: "{{ mylist | join(', ') }}"

How do you optimize Ansible Playbooks?

- Avoid running tasks on all hosts unnecessarily. Use when conditions.
- Use async for long-running tasks.
- Use tags to run specific parts of Playbooks.
- Disable gather_facts if not needed.

What are Custom Modules in Ansible?

Custom modules are Python scripts written to extend Ansible functionality. They follow a specific structure and use AnsibleModule for interaction.

What are Callback Plugins?

Callback plugins alter or extend Ansible's output. Example: Adding custom logging, notifications, or real-time updates.

How do you use the register keyword in Ansible?

register stores the output of a task into a variable.

yaml

tasks:

- name: Check free disk space

 - command: df -h

 - register: disk_space

- debug:

 - var: disk_space.stdout

What is the purpose of become, and how do you use it?

become enables privilege escalation (e.g., sudo).

yaml

tasks:

- name: Install nginx

 - apt:

 - name: nginx

 - state: present

 - become: yes

Scenario-Based Questions

Scenario: Deploy a web server and ensure it is running.

yaml

- hosts: webservers

tasks:

- name: Install Apache

apt:

name: apache2

state: present

- name: Start and enable Apache

service:

name: apache2

state: started

enabled: yes

Scenario: Copy a file to multiple hosts only if it doesn't already exist.

yaml

tasks:

- name: Copy file if not present

copy:

src: /local/path/to/file

dest: /remote/path/to/file

remote_src: yes

creates: /remote/path/to/file

Scenario: Perform a health check after deploying an application.

yaml

tasks:

- name: Check application health

uri:

url: http://localhost:8080/health

status_code: 200

register: health_check

- name: Fail if health check fails

fail:

msg: "Application is not healthy"

when: health_check.status != 200

Troubleshooting Questions

What do you do if ansible-playbook fails to connect to hosts?

- Check SSH connectivity using `ssh user@host`.
- Verify inventory file syntax.
- Test with `ansible all -m ping`.
- Confirm Ansible's control machine can resolve hostnames.

How do you handle undefined variables?

Use the default filter to avoid errors:

yaml

```
{{ variable_name | default('default_value') }}
```

-
- Set `DEFAULT_UNDEFINED_VAR_BEHAVIOR` in `ansible.cfg`.

What if Ansible is too slow during execution?

- Use `pipelining = True` in `ansible.cfg`.
- Disable fact gathering if unnecessary.
- Increase parallelism with `-f` or `forks`.

Miscellaneous Questions

How do you handle secrets securely in Ansible?

- Use Ansible Vault to encrypt sensitive data.
- Combine with tools like HashiCorp Vault for additional secret management.

What is ansible-galaxy, and how is it used?

ansible-galaxy is a command-line tool to manage roles and collections.

```
ansible-galaxy install geerlingguy.nginx
```

What is the purpose of `roles_path` in `ansible.cfg`?

`roles_path` specifies the directory where Ansible looks for roles. This is useful for organizing large projects.

