ANSIBLE CORE CONCEPTS & PLAYBOOK ESSENTIALS

**Ansible Core Concepts:**

**1. Configuration Management Principles**

Think of configuration management like a remote control that can turn on, off, or configure many devices (servers) with one command.

* Ensures consistent system configuration across multiple servers.
* Prevents configuration drift.

**2. Controller vs Target Node Functionality**

**-** **Controller Node (Ansible Control Node)**

Functionality:

* This is **where Ansible is installed**.
* It’s responsible for:
  + Running **playbooks**, **modules**, and **roles**.
  + **Managing SSH connections** to the target nodes.
  + Rendering **Jinja2 templates** and processing variables.
  + Collecting and displaying **execution results**.

What it does:

* Parses YAML playbooks.
* Executes tasks **remotely** on the target nodes using:
  + **SSH** (Linux)
  + **WinRM** (Windows)
* **Does NOT require agents** on the target nodes.
* Manages **inventory** of hosts (typically in /etc/ansible/hosts or dynamic inventory).

**-Target Node (Managed Node)**

Functionality:

* These are the **remote systems** where Ansible performs tasks (e.g., web servers, databases, routers).
* They do **not need Ansible installed**.
* Must be:
  + Reachable via **SSH (Linux/macOS)** or **WinRM (Windows)**.
  + Properly **authenticated** (usually via SSH keys).

What it does:

* Executes **commands sent from the controller node**.
* Performs tasks like:
  + Installing packages.
  + Configuring files/services.
  + Managing users, permissions, etc.

### Controller Analogy: Think of it as the ****director**** who has the script (playbook) and gives instructions to all the actors (target nodes).

### Target Analogy****:**** Think of it as the ****worker**** who follows instructions from the controller.

**How They Work Together (Flow)**

1. You write a **playbook** on the controller.
2. Ansible uses **SSH/WinRM** to connect to target nodes.
3. Tasks defined in the playbook are executed **on the target nodes**.
4. Output is sent back to the controller and shown on the terminal.

### 3. Apache HTTPD Setup via Ansible

**Steps:**

* Check Apache installation: rpm -q httpd
* Install Ansible: yum install ansible-core:
  + Check version – ansible –version
  + List all hosts – ansible all –list-hosts
* Create inventory: vim /etc/ansible/hosts:
  + Add target node IP address in file /etc/ansible/hosts:
  + [webservers]
    - 1. ansible\_user=root ansible\_ssh\_pass=<changed passwd> ansible\_become=yes ansible\_become\_method=sudo
      2. ansible\_user=root ansible\_ssh\_pass=<changed passwd> ansible\_become=yes ansible\_become\_method=sudo
  + Check hosts – ansible all –list-hosts
* Connecting target nodes via ssh (Authentication Troubleshooting):
  + Ssh <IP\_ADDRESS\_OF\_TARGET\_NODE>
  + Go to the target node and change permission in /etc/ssh/sshd\_config (if ssh connectivity error).
* **Change “PermitRootLogin No to Yes” & “PasswordAuthentication from no to yes”:**
  + Restart sshd services: systemctl restart sshd.
* Change password of root user:
  + CMD - Passwd root : <add your new password>
* Remote login & connecting to managed/target nodes:
  + Changing permissions for connecting hosts:
    - Vim /etc/ansible/ansible.cnf
    - Edit “host\_key\_checking=True to False”(save and exit).
    - Check connectivity – ansible all -m command -a date
    - Test Connection: ansible all -m ping
* Installing and configuring HTTPD(Apache on target/managed nodes):
  + Check httpd package in target nodes:
    - Rpm -q httpd
* Installing httpd package in all target nodes:
  + “ ansible all -m package -a “name=httpd state =present”
* **Idempotency:** The ansible gives output in orange color; it means it has changed the state. If you try to install the software again, the ansible detects that the software is already installed, it gives output in green color – it means what we are looking for is available in the OS, the software that we are trying to install is already in the system.

This concept is called as Idempotent. If the current state is present as the desired state. This concept is idempotent. It will saves performance and resource.

* Output in **orange** = Changed
* Output in **green** = Already Present
* Create a webpage using apache web server on the controller Node and it will automatically copy these to all the managed/target nodes:
  + Create html file:
    - <body bgcolor=’blue’ >
    - Hii from Harshwardhan
    - </body> or take a code from internet.
  + Sharing file via “Synosis Concept”:
    - Ansible all –m copy –a “src=/pathOfFile dest=/var/www/html/filename.html”
    - File = harsh.html upload/copy -> index.html
    - Controller --------------------------🡪 Managed/Target nodes
* Starting Apache(HTTPD) services:
  + Ansible all –m service –a “name=httpd state=started”
    - This will start httpd service of all target/managed nodes.

## Playbook Essentials

### 1.Running Ad hoc Commands:

ansible [what computers] -m [what to do] -a "[how to do it]"

* target\_hosts: Inventory group or host(s)
* m: Module name (e.g., ping, shell, yum, copy, etc.)
* a: Module arguments.

# 1. Ping all managed hosts: ansible all -m ping

# 2. Check disk usage: ansible all -m shell -a "df -h"

# 3. Check system uptime: ansible all -m command -a "uptime"

# 4. Install httpd using yum (RHEL-based): ansible webservers -m yum -a "name=httpd state=present"

# 5. Start httpd service: ansible webservers -m service -a "name=httpd state=started"

# 6. Copy file from control node to target: ansible webservers -m copy -a "src=/tmp/file.txt dest=/tmp/file.txt"

# 7. Create a directory: ansible all -m file -a "path=/tmp/mydir state=directory"

# 8. Change file permission: ansible all -m file -a "path=/tmp/file.txt mode=0644"

# 9. Reboot remote machines: ansible all -m reboot

# 10. Check free memory: ansible all -m shell -a "free -m"

# 11. Update apt packages (Debian-based) with sudo: ansible all -m shell -a "apt update" --become

### 2. Writing Basic Playbooks:

**Example:** Install docker

Yaml for docker\_install.yaml:

---

- name: Install Docker

hosts: all

become: yes

tasks:

- name: Install Docker

package:

name: docker

state: present

### Run:

### ansible-playbook apache\_install.yaml

### 3. YAML Syntax and Structure

* YAML = **YAML Ain’t Markup Language**
* Indentation is **critical** (spaces, not tabs)
* Key-Value structure

key: value

list:

- item1

- item2

### 4. vi or vim Editor for YAML

* Open file: vi playbook.yaml
* Insert mode: Press i
* Save & Exit:
  + Esc, then :wq or Shift+ZZ

### 5. Using ansible-playbook Command

ansible-playbook <playbook\_name>.yaml

* Use -v, -vv for verbose
* Use --check for dry-run

**6. Accessing & Understanding Module Documentation**

### What is a Module in Ansible?

In **Ansible**, a **module** is a **reusable, standalone script** that does one specific task—like installing packages, managing files, creating users, restarting services, etc.

# General syntax: ansible-doc <module\_name>

# Example: View detailed documentation for the 'package' module: ansible-doc package

# Example: View detailed documentation for the 'copy' module: ansible-doc copy

# Example: View all available modules: ansible-doc -l

# Example: Search for a module by keyword: ansible-doc -s package

**Q&A**

**1. What is the role of an inventory file in Ansible?**  
Answer:  
An inventory file in Ansible defines the managed nodes (hosts) and organizes them into groups. It can be a static file listing IP addresses or hostnames, or a dynamic inventory that pulls information from external sources like cloud providers. This structure allows Ansible to target specific groups of hosts for task execution (like production, dev, etc.).

+--------------------+

| Inventory File |

+--------------------+

|

+-----------------+------------------+

| |

+---------------+ +----------------+

| [webservers] | | [dbservers] |

| 192.168.1.10 | | 192.168.1.20 |

| 192.168.1.11 | +----------------+

+---------------+

|

|--> Ansible connects via SSH (default)

|--> Runs playbooks/tasks (install httpd, etc.)

# /etc/ansible/hosts

[webservers]

192.168.1.10

192.168.1.11

[dbservers]

192.168.1.20

**2. What is the difference between ad-hoc commands and playbooks in Ansible?**  
Answer:  
Ad-hoc commands are one-time commands executed directly from the command line for quick tasks. Playbooks are YAML files that define a series of tasks to be executed on managed nodes, allowing for complex, reusable configurations.

+------------------+ +------------------+

| Ad-hoc Command | | Playbook |

+------------------+ +------------------+

| ansible all -m | | YAML file with |

| shell -a "date" | | multiple tasks |

| | | and automation |

+--------+---------+ +--------+---------+

| |

| |

v v

[ Managed Nodes ] [ Managed Nodes ]

Quick Task Complex Workflow

| **Ad-hoc** | **Playbook** |
| --- | --- |
| Like ordering 1 pizza on the phone | Subscribing to a meal plan that delivers every week |
| Temporary, now-only action | Structured, repeatable process |

Eg: K8s – Pods/Jobs Eg: K8s – Deployment/Cronjobs

**Thank you**