Ansible

Configuration Management Tools

What is Ansible

Ansible is an open-source **IT automation tool** that enables **configuration management, application deployment, and infrastructure automation** without requiring agents on remote systems.

Key Features:

✔ **Agentless** – Uses SSH (Linux) & WinRM (Windows) for communication.

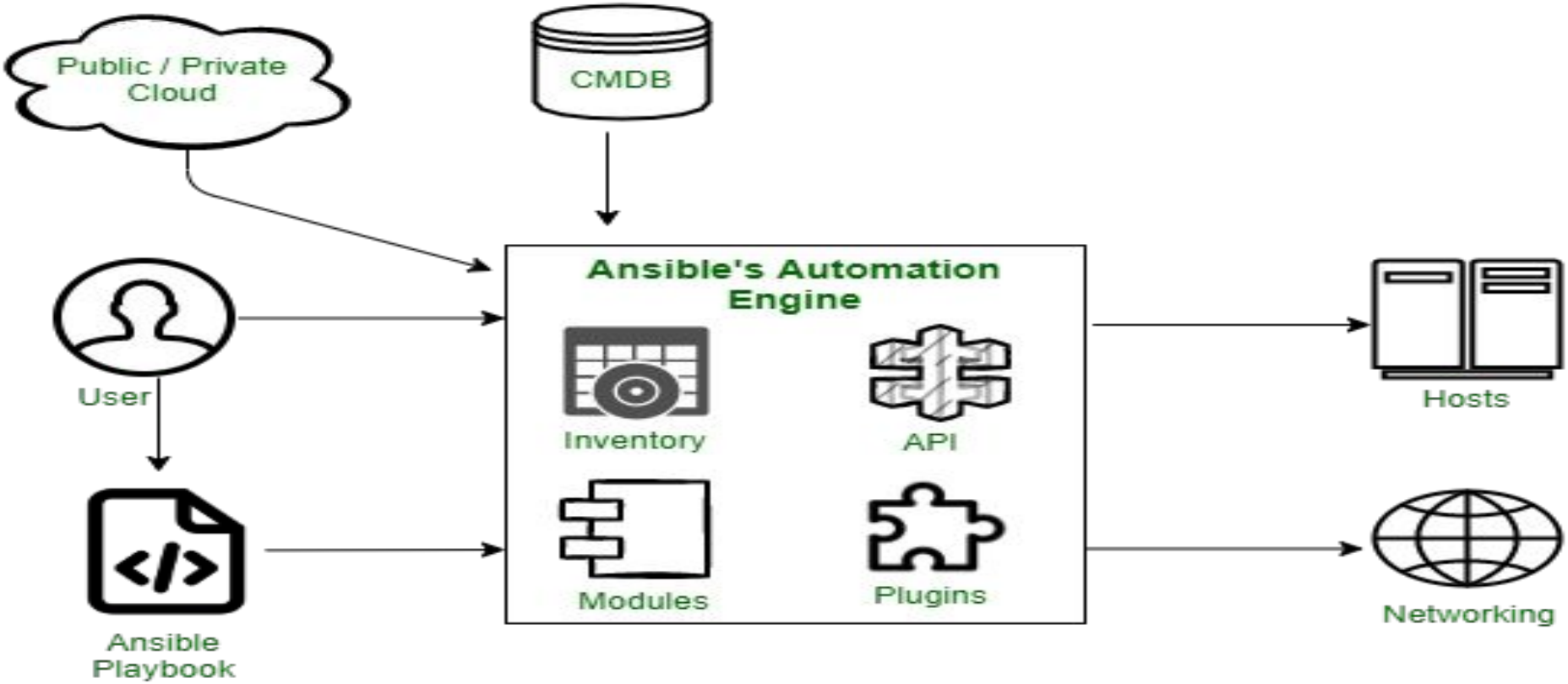
✔ **Declarative & Idempotent** – Ensures the desired state without unnecessary changes.

✔ **YAML-based Playbooks** – Easy-to-read automation scripts.

✔ **Extensible** – Supports cloud platforms, containers, and DevOps tools.

✔ **Enterprise-Ready** – Used for large-scale automation and security compliance.

Ansible Architecture-



Components of Ansible

📌 **Control Node** – The machine where Ansible is installed and executed.

📌 **Managed Nodes** – Target systems where Ansible applies configurations.

📌 **Inventory** – A file listing hosts or groups of hosts.

📌 **Modules** – Reusable scripts that perform automation tasks.

📌 **Playbooks** – YAML files containing automation instructions.

📌 **Plugins** – Extend Ansible functionalities.

Ansible Workflow

1. Install Ansible on a control node.
2. Define target systems in the **Inventory** file.
3. Write automation scripts using **Playbooks (YAML format)**.
4. Run playbooks using the `ansible-playbook` command.
5. Ansible executes tasks on remote systems via SSH/WinRM.

Ansible Playbook Example

```yaml
- name: Install Apache Web Server
  hosts: webservers
  become: yes
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: present

    - name: Start Apache Service
      service:
        name: apache2
        state: started
```

What Are Ansible Roles

Ansible **Roles** help organize playbooks into reusable, structured directories, making automation modular and scalable.

Why Use Roles?

✔ **Reusability** – Avoid repetition by creating reusable role components.
✔ **Maintainability** – Keep automation clean and structured.
✔ **Scalability** – Easily manage large projects.
✔ **Separation of Concerns** – Divide tasks into logical components.

Ansible Role Directory Structure

📌 When you create a role using `ansible-galaxy init <role_name>`, it generates the following structure:

```
my_role/
├── defaults/      # Default variables
├── files/         # Static files to copy
├── handlers/      # Handlers (e.g., restart services)
├── meta/          # Role metadata
├── tasks/         # Main task definitions
├── templates/     # Jinja2 templates
├── vars/          # Role-specific variables
```

Best Practices for Ansible Roles

✔ **Keep Roles Modular** – Each role should perform a single function.

✔ **Use Variables & Defaults** – Keep roles configurable and reusable.

✔ **Use Handlers Efficiently** – Restart services only when needed.

✔ **Follow Directory Structure** – Maintain a clean and structured format.

✔ **Use Role Dependencies** – Define dependencies in `meta/main.yml`.

Repo

https://github.com/ankit20000/Ansible