
🧰 What is Ansible?

Ansible is an **open-source automation tool** used for:

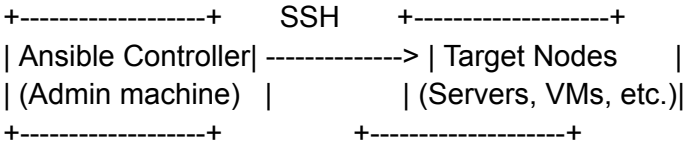
- **Configuration Management**
- **Application Deployment**
- **Provisioning infrastructure**
- **Continuous Delivery**
- **Orchestration**

It allows you to **automate repetitive tasks** and manage multiple servers with simple, human-readable **YAML-based configuration files** (called **Playbooks**), without needing agents.

🚀 Key Features

- ✅ Agentless (uses SSH or WinRM)
- ✅ Idempotent (runs repeatedly with same result)
- ✅ Declarative (you define the desired state)
- ✅ Simple syntax using YAML
- ✅ Large number of pre-built modules

🏗️ Ansible Architecture



- **Control Node (Controller):**
  Where Ansible is installed and playbooks are executed

- **Managed Nodes (Clients):**
  Servers that are being configured or automated (no Ansible installation needed)

- **Inventory:**
  A list of managed hosts (in `.ini`, `.yaml`, or dynamic format)

📦 Key Ansible Components

Component	Description
Inventory	File listing target machines (IP/DNS)
Playbook	YAML file with instructions (tasks/roles)
Task	A single unit of work (e.g., install nginx)
Module	Code executed by Ansible to perform tasks
Role	Collection of tasks, templates, files, etc., used for reuse
Handler	Runs only when notified (e.g., restart a service)
Facts	Collected system info from managed hosts
Variable	Dynamic values used inside playbooks
Templates	Jinja2-based dynamic configuration files

🔧 Example Workflow

1. **Write an Inventory file**

```
[web]
web1.example.com
```

web2.example.com

[db]
db1.example.com

2. **Create a Playbook**

```
- name: Install and start nginx
  hosts: web
  become: true
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present

    - name: Ensure nginx is running
      service:
        name: nginx
        state: started
```

3. **Run the Playbook**

```
ansible-playbook -i inventory.ini webserver.yaml
```

🔍 Ansible Command Examples

Command	Purpose
`ansible --version`	Check version
`ansible all -m ping -i inventory.ini`	Ping all hosts
`ansible-playbook playbook.yaml -i inventory.ini`	Run playbook
`ansible web -a "uptime"`	Run ad-hoc command

🧰 Use Cases

Use Case	Example
Configuration Management	Install packages, manage users, deploy apps
Provisioning	Create EC2 instances (with `boto3`)
App Deployment	Deploy Django, Node.js apps
Security Automation	Patch updates, firewall settings
CI/CD Pipelines	Integrate with Jenkins, GitLab
Cloud Automation	AWS, Azure, GCP using dynamic inventory & modules

---

📚 Advantages of Ansible

- ✅ **No agent** installation
- ✅ Simple to learn (YAML syntax)
- ✅ Scales easily
- ✅ Integrates with all major cloud providers
- ✅ Actively maintained and supported (Red Hat)

⚙️ Real-World Scenarios

Scenario	Solution with Ansible
Setup 10 servers with Apache	Use a playbook to install, configure, and start Apache
Roll out firewall rules	Define rules as tasks and apply across fleet
Rotate SSH keys	Automate key replacement across instances
Provision EC2 + install app	Use Ansible with AWS modules to create and configure

Manage Kubernetes nodes	Configure kubelet, kubeadm, kube-proxy, etc.

📁 Directory Structure (Best Practice)

```
my-ansible-project/
├── inventory.ini
├── site.yaml
├── roles/
│   └── webserver/
│       ├── tasks/
│       │   └── main.yaml
│       ├── templates/
│       └── handlers/
```

🧪 Suggested Lab Exercise

- Install Ansible on a control node (Ubuntu or Amazon Linux)
- Configure a basic inventory with 2 EC2 instances
- Write a playbook to:
    - Install Apache
    - Create a web page
    - Start the service
- Verify using browser or `curl`

📚 Learning Resources

- 🔗 Official Docs
- 📘 *Ansible for DevOps* by Jeff Geerling

- 📺 YouTube: Red Hat and community tutorials
- 🔧 GitHub: Awesome Ansible Repo
