Date: 15.07.2025



Copyright @ 2024 PibyThree.com All Rights Reserved

Ansible

Contents

[1. Introduction 4](#_Toc203481204)

[a. What is Ansible? 4](#_Toc203481205)

[b. Key Features 4](#_Toc203481206)

[c. Ansible vs Its Alternatives 4](#_Toc203481207)

[2. Core Concepts 5](#_Toc203481208)

[3. Installing and Setting up Ansible 5](#_Toc203481209)

[a. Installation 5](#_Toc203481210)

[b. Setting up SSH Access from control node to managed nodes 6](#_Toc203481211)

[c. Configuring Inventory file 6](#_Toc203481212)

[d. ansible.cfg Configuration File 6](#_Toc203481213)

[4. Basic Usage 6](#_Toc203481214)

[a. Running Ad-Hoc Commands 6](#_Toc203481215)

[b. Running Playbooks 6](#_Toc203481216)

[c. Using Variables 7](#_Toc203481217)

[d. Conditionals and Loops 7](#_Toc203481218)

[e. Tags and Includes 7](#_Toc203481219)

[5. Playbook Structure and Syntax 8](#_Toc203481220)

[a. Basic Playbook Structure 8](#_Toc203481221)

[b. Common Modules and Syntax 8](#_Toc203481222)

[c. Other Concepts 8](#_Toc203481223)

[d. Best Practices 9](#_Toc203481224)

[6. Roles and Reusability 9](#_Toc203481225)

[a. What Is a Role? 9](#_Toc203481226)

[b. Role Directory Structure 9](#_Toc203481227)

[c. Creating a Role 9](#_Toc203481228)

[d. Using Roles in a Playbook 10](#_Toc203481229)

[e. Sharing and Downloading Roles (Ansible Galaxy) 10](#_Toc203481230)

[f. include\_role vs roles 10](#_Toc203481231)

[g. Benefits of Using Roles 10](#_Toc203481232)

[7. Advanced Features 10](#_Toc203481233)

[a. Ansible Vault 10](#_Toc203481234)

[b. Templates with Jinja2 10](#_Toc203481235)

[c. Dynamic Inventories 11](#_Toc203481236)

[d. Callback Plugins 11](#_Toc203481237)

[e. Custom Modules 11](#_Toc203481238)

[f. Collections 11](#_Toc203481239)

[g. Asynchronous Tasks 11](#_Toc203481240)

[8. Best Practices 12](#_Toc203481241)

[a. Directory Structure 12](#_Toc203481242)

[b. Use Roles and Collections 12](#_Toc203481243)

[c. Keep Playbooks Idempotent 12](#_Toc203481244)

[d. Protect Sensitive Data 12](#_Toc203481245)

[e. Test Changes Locally First 12](#_Toc203481246)

[f. Use Tags Wisely 13](#_Toc203481247)

[g. Track and Log Outputs 13](#_Toc203481248)

[h. Use Variables and Defaults 13](#_Toc203481249)

[i. Reuse and Share Code 13](#_Toc203481250)

[j. Lint and Validate 13](#_Toc203481251)

[9. Troubleshooting and Debugging 13](#_Toc203481252)

[a. Use Verbose Mode 13](#_Toc203481253)

[b. Use the debug Module 13](#_Toc203481254)

[c. Dry Run with --check 14](#_Toc203481255)

[d. Validate Syntax Before Running 14](#_Toc203481256)

[e. Register and Inspect Results 14](#_Toc203481257)

[f. Examine Logs (if using callback plugins) 14](#_Toc203481258)

[g. Common Errors and Fixes 14](#_Toc203481259)

[h. Use ansible-lint 14](#_Toc203481260)

[i. Use --start-at-task for Focused Debugging 15](#_Toc203481261)

[j. Use --step to Confirm Before Each Task 15](#_Toc203481262)

[10. Real-World Examples 15](#_Toc203481263)

[a. Example 1: Web Server Deployment (Nginx) 15](#_Toc203481264)

[b. Example 2: Managing Encrypted Secrets 16](#_Toc203481265)

[c. Example 3: Provision AWS EC2 Instances (with amazon.aws collection) 16](#_Toc203481266)

[d. Example 4: Use of Roles in a Complete Playbook 17](#_Toc203481267)

[e. Example 5: CI/CD Integration 18](#_Toc203481268)

# 1. Introduction

## What is Ansible?

Ansible is an open-source automation tool primarily used for **configuration management**, **application deployment**, **task automation**, and **orchestration**. It helps automate repetitive IT tasks, making system administration easier, more consistent, and scalable.

Ansible uses a simple, human-readable language based on YAML (Yet Another Markup Language) called **playbooks** to describe automation jobs.

## Key Features

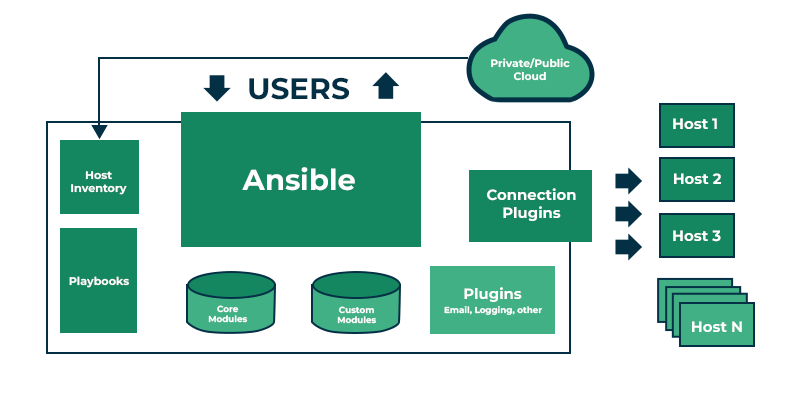
|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Agentless Architecture** | Does not require any agent software on target machines; communicates over SSH (Linux/Unix) or WinRM (Windows). |
| **Simple and Easy to Learn** | Playbooks are written in YAML, which is straightforward and readable, even for non-programmers. |
| **Idempotency** | Tasks are idempotent, so they can run multiple times without causing unintended changes. |
| **Extensible with Modules** | Includes a large library of built-in modules and supports custom modules for managing systems and apps. |
| **Agentless and Secure** | Uses SSH keys for secure communication, eliminating the need for extra software on managed nodes. |
| **Powerful Orchestration** | Supports complex workflows with dependencies and conditional execution across multiple systems. |
| **Wide Platform Support** | Compatible with Linux, Unix, Windows, network devices, cloud platforms, containers, and more. |

## Ansible vs Its Alternatives

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Ansible** | **Puppet** | **Chef** | **SaltStack** |
| **Agent Required?** | No | Yes | Yes | Optional |
| **Language** | YAML (Playbooks) | Domain-Specific Language | Ruby DSL | YAML / Python |
| **Learning Curve** | Low | Moderate | Moderate to High | Moderate |
| **Orchestration** | Strong | Moderate | Moderate | Strong |
| **Configuration** | Push-based | Pull-based | Pull-based | Both |
| **Extensibility** | Extensive modules, plugins | Modules and manifests | Cookbooks | Modules and states |

# 2. Core Concepts

|  |  |  |
| --- | --- | --- |
| **Concept** | **Description** | **Notes** |
| **Control Node** | The machine where Ansible is installed and run | Your laptop or server controlling other machines |
| **Managed Nodes** | Remote machines Ansible manages | Servers, cloud instances, network devices |
| **Inventory** | List of managed nodes (hosts) | Static file or dynamic script listing servers |
| **Modules** | Pre-built tools to perform tasks on managed nodes | apt for package install, service for services |
| **Playbooks** | YAML files describing automation workflows | Define tasks to run on hosts |
| **Roles** | Reusable, organized collections of tasks and files | Standard directory structure for easier sharing |
| **Variables** | Values that customize tasks and playbooks | OS type, usernames, paths |
| **Facts** | Automatically collected system info from managed nodes | IP address, OS version |
| **Handlers** | Tasks triggered by other tasks (usually for restarts) | Restart service after config changes |
| **Templates** | Jinja2 files used to create dynamic configuration files | Config files with variables inside |



# 3. Installing and Setting up Ansible

## Installation

Refer this for Ansible Installation: [Installation Guide — Ansible Documentation](https://docs.ansible.com/ansible/latest/installation_guide/index.html)

## Setting up SSH Access from control node to managed nodes

1. Generate SSH key on control node - `ssh-keygen`

2. Copy public key to managed nodes - `ssh-copy-id user@<managed-node-ip>`

|  |
| --- |
| [webservers]  web1.example.com  web2.example.com  [dbservers]  db1.example.com |

## Configuring Inventory file

The inventory file lists hosts managed by Ansible.

Default location: `/etc/ansible/hosts`

You can group hosts and refer to groups in playbooks.

## ansible.cfg Configuration File

|  |
| --- |
| [defaults]  inventory = ./hosts  remote\_user = ubuntu  private\_key\_file = ~/.ssh/id\_rsa |

Controls Ansible settings such as:

* + Default inventory location
  + Remote user
  + SSH options
  + Privilege escalation settings (sudo)

Place `ansible.cfg` in your project directory or home directory.

# 4. Basic Usage

## Running Ad-Hoc Commands

Ad-hoc commands let you execute simple tasks directly from the command line without writing a playbook.

**Syntax: `**ansible <host-pattern> -m <module> -a "<module-arguments>"`

**Examples:**

* Ping all hosts in inventory: `ansible all -m ping`
* Check uptime on webservers: `ansible webservers -m command -a "uptime"`
* Install a package using apt: `ansible all -m apt -a "name=htop state=present" --become`

## Running Playbooks

A playbook is a YAML file containing tasks to run on specified hosts.

|  |  |
| --- | --- |
| **install\_apache.yml** | **Run it** |
| - name: Install Apache web server  hosts: webservers  become: true  tasks:  - name: Ensure Apache is installed  apt:  name: apache2  state: present  - name: Ensure Apache is running  service:  name: apache2  state: started  enabled: true | `ansible-playbook install\_apache.yml` |

## Using Variables

Variables let you customize your playbooks without hardcoding value

|  |
| --- |
| - name: Install a package with a variable  hosts: all  vars:  package\_name: htop  tasks:  - name: Install package  apt:  name: "{{ package\_name }}"  state: present |

## Conditionals and Loops

|  |  |
| --- | --- |
| **Conditionals** | **Loops** |
| - name: Install Nginx only on Debian  apt:  name: nginx  state: present  when: ansible\_facts['os\_family'] == "Debian" | - name: Install multiple packages  apt:  name: "{{ item }}"  state: present  loop:  - git  - curl  - htop |

## Tags and Includes

* + **Tags** allow you to run specific parts of a playbook
  + **Includes** break playbooks into smaller files

|  |  |
| --- | --- |
| **Tags** | **Includes** |
| - name: Install and configure  hosts: all  tasks:  - name: Install nginx  apt:  name: nginx  state: present  tags: install  - name: Configure nginx  template:  src: nginx.conf.j2  dest: /etc/nginx/nginx.conf  tags: config | - name: Load common tasks  import\_tasks: common.yml |

Run only the install task: `ansible-playbook site.yml --tags install`

# 5. Playbook Structure and Syntax

## Basic Playbook Structure

|  |  |
| --- | --- |
| **General Description** | **Example** |
| - name: Description of the play  hosts: group\_or\_host  become: true # for privilege escalation  vars:  some\_var: value  tasks:  - name: Description of task 1  module\_name:  option1: value1  option2: value2  - name: Description of task 2  module\_name:  ... | - name: Install and start Apache  hosts: webservers  become: true  tasks:  - name: Install Apache  apt:  name: apache2  state: present  update\_cache: yes  - name: Start Apache  service:  name: apache2  state: started  enabled: true |

## Common Modules and Syntax

|  |  |  |
| --- | --- | --- |
| **Module** | **Purpose** | **Example** |
| apt, yum | Package management | name=nginx state=present |
| service | Manage services | name=nginx state=started enabled=true |
| copy | Copy files to remote hosts | src=local.conf dest=/etc/config.conf |
| template | Use Jinja2 templates | src=file.j2 dest=/etc/file.conf |
| command | Run raw shell commands | command: uptime |
| file | Set file attributes | path=/etc/file mode=0644 state=touch |
| user | Manage users | name=john state=present shell=/bin/bash |

## Other Concepts

|  |  |
| --- | --- |
| **Error Handling and Retries** | **Debugging** |
| - name: Try a command with retries  command: /path/to/sometimes\_fails.sh  register: result  retries: 3  delay: 5  until: result.rc == 0 | - name: Show variable value  debug:  var: ansible\_facts['distribution']  Or for custom messages:  - debug:  msg: "This is a debug message" |

## Best Practices

* **Use descriptive names**: Makes tasks easier to understand.
* **Group related tasks**: Keeps your work organized and readable.
* **Use variables**: Avoids hardcoding and makes reuse easier.
* **Separate logic into roles**: Helps make playbooks modular and reusable.
* **Use version control**: Lets you track and manage changes with tools like Git.

# 6. Roles and Reusability

## What Is a Role?

A **role** is a structured directory layout that contains everything needed to automate a specific task or feature, such as installing a web server or setting up a database.

Each role can contain:

* Tasks
* Handlers
* Variables
* Templates
* Files
* Defaults
* Metadata

## Role Directory Structure

Only tasks/main.yml is **required**; the rest are optional.

|  |
| --- |
| my\_role/  |-- defaults/  |-- main.yml # Default variables  |-- files/  |-- example.conf # Static files to copy  |-- handlers/  |-- main.yml # Handlers triggered by tasks  |-- meta/  |-- main.yml # Role metadata (dependencies, etc.)  |-- tasks/  |-- main.yml # Main list of tasks  |-- templates/  |-- config.j2 # Jinja2 templates  |-- vars/  |-- main.yml # Variables with higher priority |

## Creating a Role

To create a role: `ansible-galaxy init my\_role`

This automatically generates the folder structure above.

## Using Roles in a Playbook

|  |  |
| --- | --- |
| **Basic Role Usage** | **Role with Variables** |
| - name: Apply webserver role  hosts: webservers  become: true  roles:  - webserver | roles:  - role: nginx  vars:  listen\_port: 8080 |

## Sharing and Downloading Roles (Ansible Galaxy)

[Ansible Galaxy](https://galaxy.ansible.com/) is a hub where you can find and share roles.

|  |  |
| --- | --- |
| **Installing a role** | **Using installed role** |
| `ansible-galaxy install geerlingguy.nginx` | roles:  - geerlingguy.nginx |

## include\_role vs roles

* roles (in a play): used in the `roles:` section of a play.
* include\_role (in a task): dynamically include a role inside a task.

## Benefits of Using Roles

* **Reusability**: Use the same role across different playbooks or projects.
* **Organization**: Keeps tasks, variables, and templates well-structured.
* **Modularity**: Makes it easier to develop and test small parts of automation.
* **Shareability**: Roles can be shared through Ansible Galaxy or Git.

# 7. Advanced Features

## Ansible Vault

Ansible Vault is used to **encrypt sensitive data** like passwords, keys, or secrets.

|  |  |
| --- | --- |
| Encrypt a file | ansible-vault encrypt secrets.yml |
| Decrypt a file | ansible-vault decrypt secrets.yml |
| Edit encrypted file | ansible-vault edit secrets.yml |
| Run playbook with vault | ansible-playbook playbook.yml --ask-vault-pass |

You can also use vault IDs or key files for non-interactive decryption.

## Templates with Jinja2

Templates let you create dynamic configuration files using **Jinja2** syntax and Ansible variables.

|  |  |
| --- | --- |
| **Example template (nginx.conf.j2)** | **In playbook** |
| server {  listen {{ port }};  server\_name {{ domain }};  root {{ web\_root }};  } | - name: Deploy nginx config  template:  src: nginx.conf.j2  dest: /etc/nginx/nginx.conf |

## Dynamic Inventories

Instead of static inventory files, you can use **scripts or cloud plugins** to dynamically generate host lists (e.g., AWS, GCP, Azure).

**`**ansible-inventory -i aws\_ec2.yml --graph`

To use:

* Install cloud collection (e.g., amazon.aws)
* Create inventory plugin config (aws\_ec2.yml)
* Reference with -i aws\_ec2.yml

## Callback Plugins

Callback plugins change **how Ansible displays output** or reports status.

|  |
| --- |
| Enable in ansible.cfg |
| [defaults]  stdout\_callback = yaml |

Common use: pretty output, JSON logs, sending results to Slack.

More examples:

* json: machine-readable output
* minimal: less noisy output
* slack: send playbook results to Slack (custom plugin)

## Custom Modules

You can create your own Ansible modules in Python.

|  |  |
| --- | --- |
| **Structure** | **Use with** |
| #!/usr/bin/python  from ansible.module\_utils.basic import AnsibleModule  def main():  module = AnsibleModule(argument\_spec=dict(name=dict(type='str', required=True)))  name = module.params['name']  module.exit\_json(changed=False, msg=f"Hello {name}")  if \_\_name\_\_ == '\_\_main\_\_':  main() | - name: Run custom module  my\_module:  name: world |

Put your module in library/ or set library = ./library in ansible.cfg.

## Collections

Collections package multiple roles, plugins, and modules together.

|  |  |
| --- | --- |
| **Install from Ansible Galaxy** | **Use in playbook** |
| `ansible-galaxy collection install community.general` | - name: Use community.general collection  community.general.handy\_module:  option: value |

|  |
| --- |
| - name: Run a script asynchronously  command: /usr/bin/long\_script.sh  async: 300  poll: 0 |

## Asynchronous Tasks

Run long tasks **in the background**:

You can later check job status with polling.

# 8. Best Practices

## Directory Structure

|  |  |
| --- | --- |
| project/  |-- ansible.cfg  |-- inventory/  | |-- hosts.yml  | |-- group\_vars/  | |-- all.yml  |-- playbooks/  | |-- webserver.yml  | |-- database.yml  |-- roles/  | |-- common/  | |-- nginx/  |-- templates/ | Use roles/ to modularize tasks, and group your inventories and variables for better organization. |

## Use Roles and Collections

* Break large playbooks into reusable roles.
* Share roles using **Ansible Galaxy** or **collections**.
* Name roles clearly and consistently (e.g., webserver, db\_setup).

## Keep Playbooks Idempotent

* Tasks should be **safe to run multiple times** without causing unintended side effects.
* Use modules like apt, copy, file instead of raw commands to ensure idempotency.

|  |  |
| --- | --- |
| **Good** | **Bad** |
| yaml  - name: Install nginx  apt:  name: nginx  state: present | yaml  - name: Install nginx (non-idempotent)  command: apt install nginx |

## Protect Sensitive Data

* Use **Ansible Vault** for passwords, secrets, API keys.
* Avoid storing plaintext credentials in playbooks or Git.

`ansible-vault encrypt secrets.yml`

## Test Changes Locally First

* Use **Vagrant**, **Docker**, or a test environment before applying to production.
* Use --check (dry-run) mode: `ansible-playbook site.yml --check`

## Use Tags Wisely

|  |  |
| --- | --- |
| Tag tasks to run only specific parts of a playbook  Run only restart tasks:  `ansible-playbook site.yml --tags restart` | - name: Restart nginx  service:  name: nginx  state: restarted  tags: restart |

## Track and Log Outputs

* Use verbose mode (-v, -vvv) when debugging: `ansible-playbook site.yml -vv`
* Use callback plugins for structured output (e.g., JSON, YAML).

## Use Variables and Defaults

* Centralize values in group\_vars/ or defaults/main.yml.
* Avoid hardcoding values across multiple playbooks.

## Reuse and Share Code

* Create **roles**, not repetitive tasks.
* Use **includes**, **import\_tasks**, and **handlers** for modularity.
* Share reusable components with your team or publish to Ansible Galaxy.

## Lint and Validate

* Use **ansible-lint** to catch common errors: `ansible-lint playbook.yml`
* Validate syntax before running: `ansible-playbook playbook.yml --syntax-check`

# 9. Troubleshooting and Debugging

## Use Verbose Mode

|  |  |
| --- | --- |
| **Command** | **Description** |
| ansible-playbook site.yml -v | Shows basic info about tasks and results |
| ansible-playbook site.yml -vv | Adds more output (e.g., variables, changes) |
| ansible-playbook site.yml -vvv | Shows SSH details, task arguments, and full debug |
| ansible-playbook site.yml -vvvv | Includes connection debugging and internal data (rarely needed) |

## Use the debug Module

|  |  |
| --- | --- |
| Use the debug module in playbooks to print variables or custom messages | - name: Show variable  debug:  var: ansible\_facts['distribution']  - name: Print custom message  debug:  msg: "Web root is set to {{ web\_root }}" |

## Dry Run with --check

The --check flag runs the playbook in **"dry run"** mode: `ansible-playbook site.yml --check`

* Shows what would change without making any changes.
* Great for testing logic or reviewing pending updates.

## Validate Syntax Before Running

Catch errors early with: `ansible-playbook playbook.yml --syntax-check`

This checks for YAML formatting and module errors.

## Register and Inspect Results

|  |  |
| --- | --- |
| Use register to capture output from a task and analyze it in later tasks: | - name: Run shell command  shell: uptime  register: result  - debug:  var: result.stdout |

## Examine Logs (if using callback plugins)

|  |  |
| --- | --- |
| Enable logging in ansible.cfg: | This stores detailed logs that can be reviewed after execution. |
| [defaults]  log\_path = ./ansible.log |

## Common Errors and Fixes

|  |  |  |
| --- | --- | --- |
| **Error** | **Possible Cause** | **Fix** |
| UNREACHABLE! | SSH error or host is down | Check SSH config, network, hostnames |
| FAILED! => {"msg": "Module not found"} | Missing or misspelled module | Check spelling or module availability |
| Permissions denied (sudo error) | Wrong become settings or user rights | Ensure correct become: true and user configs |
| YAML syntax error | Incorrect spacing or indentation | Validate with --syntax-check, use 2 spaces |
| Variable undefined | Variable not set or misnamed | Set in vars, defaults, or group\_vars |

## Use ansible-lint

Catch common mistakes with this linter tool: `ansible-lint playbook.yml`

It checks for:

* Deprecated module usage
* Bad practices
* Missing handlers or variables
* Improper formatting

## Use --start-at-task for Focused Debugging

Resume a playbook at a specific task by name: `ansible-playbook site.yml --start-at-task="Install packages"`

Helps when debugging failures late in a playbook.

## Use --step to Confirm Before Each Task

This interactive mode asks you before running each task: `ansible-playbook site.yml --step`

Useful when trying to isolate where a problem starts.

# 10. Real-World Examples

## Example 1: Web Server Deployment (Nginx)

**Objective:** Install and configure Nginx on a group of webservers.

|  |  |
| --- | --- |
| Directory Structure | playbook.yml |
| nginx\_deploy/  |-- playbook.yml  |-- templates/  | |-- nginx.conf.j2  |-- inventory/  |-- hosts | - name: Deploy Nginx on web servers  hosts: webservers  become: true  vars:  nginx\_port: 80  tasks:  - name: Install Nginx  apt:  name: nginx  state: present  update\_cache: yes  - name: Configure Nginx  template:  src: templates/nginx.conf.j2  dest: /etc/nginx/nginx.conf  - name: Start and enable Nginx  service:  name: nginx  state: started  enabled: true |
| inventory/hosts |
| [webservers]  192.168.1.10 |
| templates/nginx.config.j2 |
| server {  listen {{ nginx\_port }};  server\_name localhost;  location / {  root /var/www/html;  index index.html;  }  } |

`ansible-playbook nginx\_deploy/playbook.yml -i nginx\_deploy/inventory/hosts`

## Example 2: Managing Encrypted Secrets

**Objective:** Store a database password securely.

|  |  |
| --- | --- |
| Directory Structure | playbook.yml |
| vault\_example/  |-- playbook.yml  |-- templates/  | |-- db\_config.j2  |-- inventory/  | |-- hosts  |-- group\_vars/  |-- dbservers/  |-- vault.yml | - name: Configure database  hosts: dbservers  become: true  tasks:  - name: Set DB password  template:  src: templates/db\_config.j2  dest: /etc/db.conf |
| inventory/hosts |
| [dbservers]  192.168.1.20 |
| group\_vars/dbservers/vault.yml | templates/db\_config.j2 |
| db\_password: "SuperSecret123"  // `ansible-vault create vault.yml` | DB\_PASSWORD={{ db\_password }} |

`ansible-playbook vault\_example/playbook.yml -i vault\_example/inventory/hosts --ask-vault-pass`

## Example 3: Provision AWS EC2 Instances (with amazon.aws collection)

|  |  |
| --- | --- |
| Directory Structure | playbook.yml |
| aws\_ec2/  |-- playbook.yml | - name: Launch EC2 instance  hosts: localhost  gather\_facts: false  vars:  key\_name: my-key  region: us-east-1  tasks:  - name: Launch EC2 instance  amazon.aws.ec2\_instance:  key\_name: "{{ key\_name }}"  region: "{{ region }}"  instance\_type: t2.micro  image\_id: ami-0abcdef1234567890  wait: yes  count: 1  register: ec2 |
| Install Collection |
| ansible-galaxy collection install amazon.aws |

`ansible-playbook aws\_ec2/playbook.yml -i localhost,` [ `-i localhost,` 🡪 Interpreted as a literal host ]

## Example 4: Use of Roles in a Complete Playbook

|  |  |
| --- | --- |
| Directory Structure | roles/apache/tasks/main.yml |
| project/  |-- playbook.yml  |-- roles/  |-- apache/  |-- tasks/  | |-- main.yml  |-- templates/  |-- httpd.conf.j2 | - name: Install Apache  apt:  name: apache2  state: present  - name: Deploy config  template:  src: httpd.conf.j2  dest: /etc/apache2/apache2.conf  - name: Start Apache  service:  name: apache2  state: started  enabled: true |
| playbook.yml |
| - name: Setup Apache  hosts: webservers  become: true  roles:  - apache |
| roles/apache/templates/httpd.conf.j2 |
| ServerName localhost  <Directory /var/www/html>  Options Indexes FollowSymLinks  AllowOverride None  Require all granted  </Directory> |

`ansible-playbook project/playbook.yml -i project/inventory/hosts`

## Example 5: CI/CD Integration

|  |  |
| --- | --- |
| Directory Structure | .github/workflows/deploy.yml |
| cicd/  |-- .github/  | |-- workflows/  | |-- deploy.yml  |-- deploy.yml  |-- inventory/  | |-- hosts | jobs:  deploy:  runs-on: ubuntu-latest  steps:  - name: Checkout code  uses: actions/checkout@v3  - name: Set up Ansible  run: |  sudo apt update  sudo apt install ansible -y  - name: Run Ansible playbook  run: |  ansible-playbook deploy.yml -i inventory/hosts --key-file ~/.ssh/id\_rsa |
| inventory/hosts |
| [webservers]  myserver.example.com |
| deploy.yml (Example Ansible playbook) |
| - name: CI Deployed Webserver  hosts: webservers  become: true  tasks:  - name: Ensure Nginx is installed  apt:  name: nginx  state: present |

**`**ansible-playbook cicd/deploy.yml -i cicd/inventory/hosts --key-file ~/.ssh/id\_rsa**`**

(For local testing)