🔥 Ansible Zero to Hero: The Ultimate Hands-On Guide

What is Ansible?

Ansible is an **open-source IT automation tool** developed by Red Hat. It helps automate tasks like:

- **Configuration Management** and **Application Deployment**
- **Infrastructure Provisioning** and **Orchestration**
- **Security Automation** and **Continuous Delivery**

It's **agentless**, uses **SSH (Linux/Unix)** or **WinRM (Windows)** to connect-remote systems.

Why Use Ansible?

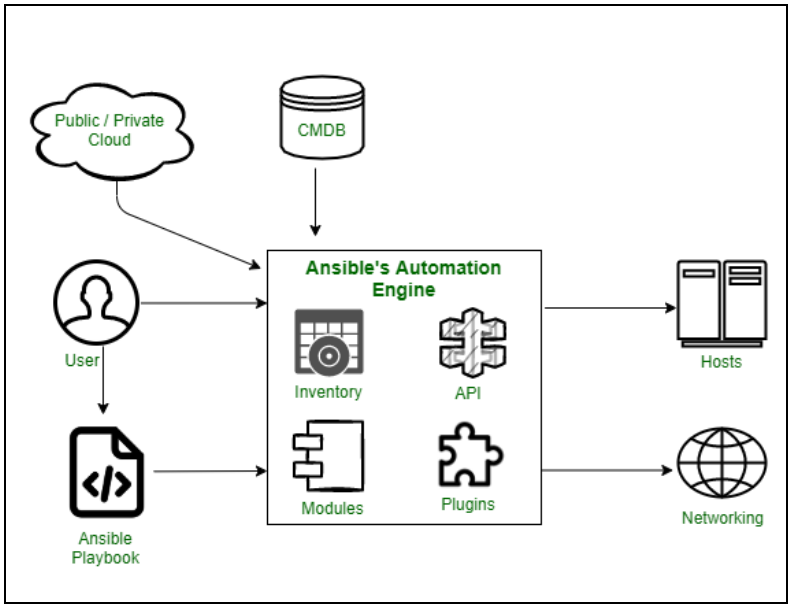
- ☑ **Simple**: Uses **YAML** for writing automation scripts (playbooks)
- ☑ **Agentless**: No need to install anything on remote nodes
- ☑ **Idempotent**: same playbook multiple times won't change, already in the desired state
- ☑ **Scalable**: Manages thousands of nodes easily
- ☑ **Cross-platform**: Works with Linux, Windows, cloud platforms, network devices

---

How Ansible Works

1. write **Playbooks** in YAML and Ansible reads the **Inventory** to find target systems.
2. It connects to those using **SSH or WinRM** and Executes **Modules** defined in the playbook.
3. Applies changes idempotently.

Ansible Architecture (Simplified)



Core Components of Ansible

Component	Description
Inventory	List of target hosts to manage (static or dynamic)
Modules	Reusable, standalone scripts (e.g., install package, copy file, create user)
Playbooks	YAML files defining automation tasks
Tasks	Steps inside a playbook
Roles	Structured way to organize playbooks and variables
Variables	Custom values for different environments
Facts	System information gathered automatically from hosts
Plugins	Extend core Ansible functionality
Handlers	Triggered only when a task reports a change
Templates	Jinja2 templating to dynamically generate config files

---

Ansible Vault for secrets.
Ansible Galaxy for community roles.
Ex: Install a Role from **Ansible Galaxy**: **ansible-galaxy install geerlingguy.nginx**
Ansible Tower (GUI/enterprise edition by Red Hat).

Ansible Ad-Hoc Commands

```
# 1. Ping all hosts
ansible all -m ping

# 2. List all inventory hosts
ansible all --list-hosts

# 3. Run command (uptime)
ansible all -a "uptime"

# 4. Install package using yum
ansible all -b -m yum -a "name=git state=present"

# 5. Remove package using yum
ansible all -b -m yum -a "name=git state=absent"

# 6. Start a service
ansible all -b -m service -a "name=nginx state=started"

# 7. Stop a service
ansible all -b -m service -a "name=nginx state=stopped"

# 8. Restart a service
ansible all -b -m service -a "name=nginx state=restarted"
```

```
# 9. Create a user
ansible all -b -m user -a "name=devops shell=/bin/bash"

# 10. Copy a file
ansible all -b -m copy -a "src=/tmp/file.txt dest=/tmp/file.txt"

# 11. Fetch a file
ansible all -b -m fetch -a "src=/var/log/messages dest=./logs/ flat=yes"

# 12. Check disk space
ansible all -a "df -h"

# 13. Reboot a host
ansible all -b -m reboot

# 14. Create a directory
ansible all -b -m file -a "path=/opt/demo state=directory mode=0755"

# 15. Change file permissions
ansible all -b -m file -a "path=/tmp/file.txt mode=0644"

# 16. Download a file from URL
ansible all -b -m get_url -a "url=https://example.com/file.txt
dest=/tmp/file.txt"

# 17. Create a symbolic link
ansible all -b -m file -a "src=/opt/app current dest=/opt/latest state=link"

# 18. Set cron job
ansible all -b -m cron -a "name='daily backup' job='/usr/bin/backup.sh'
minute=0 hour=2"

# 19. View system info (facts)
ansible all -m setup -a "filter=ansible_distribution"

# 20. Run shell with sudo
ansible all -b -a "systemctl restart nginx"
```

---

60 Ansible Commands with Source Code Examples

```
###################################
# 1. Ping All Hosts
###################################
- name: Ping all hosts
  hosts: all
  tasks:
```

```yaml
    - name: Ping
      ping:

####################################
# 2. Run Shell Command
####################################
- name: Run uptime on all hosts
  hosts: all
  tasks:
    - name: Uptime
      shell: uptime

####################################
# 3. Install NGINX
####################################
- name: Install nginx package
  hosts: all
  become: yes
  tasks:
    - name: Install nginx
      yum:
        name: nginx
        state: present

####################################
# 4. Create User
####################################
- name: Create devops user
  hosts: all
  become: yes
  tasks:
    - name: Add user
      user:
        name: devops
        shell: /bin/bash

####################################
# 5. Copy File
####################################
- name: Copy file to host
  hosts: all
  become: yes
  tasks:
    - name: Copy index.html
      copy:
        src: ./index.html
        dest: /var/www/html/index.html
```

```yaml
###################################
# 6. Restart Service
###################################
- name: Restart nginx
  hosts: all
  become: yes
  tasks:
    - name: Restart nginx
      service:
        name: nginx
        state: restarted

###################################
# 7. Update System
###################################
- name: Update system packages
  hosts: all
  become: yes
  tasks:
    - name: Update all packages
      yum:
        name: '*'
        state: latest

###################################
# 8. Download File
###################################
- name: Download file
  hosts: all
  become: yes
  tasks:
    - name: Download via get_url
      get_url:
        url: https://example.com/file.txt
        dest: /tmp/file.txt

###################################
# 9. Unarchive Zip File
###################################
- name: Unarchive zip file
  hosts: all
  become: yes
  tasks:
    - name: Extract zip
      unarchive:
        src: /tmp/archive.zip
        dest: /opt/
        remote_src: yes
```

```yaml
####################################
# 10. Set Hostname
####################################
- name: Set hostname
  hosts: all
  become: yes
  tasks:
    - name: Set system hostname
      hostname:
        name: app-server

####################################
# 11. Create Directory
####################################
- name: Create directory
  hosts: all
  become: yes
  tasks:
    - name: Create /opt/myapp
      file:
        path: /opt/myapp
        state: directory
        mode: '0755'

####################################
# 12. Create Cron Job
####################################
- name: Schedule cron job
  hosts: all
  become: yes
  tasks:
    - name: Add cron entry
      cron:
        name: "backup job"
        minute: "0"
        hour: "2"
        job: "/usr/local/bin/backup.sh"

####################################
# 13. Replace Line in File
####################################
- name: Replace line in config
  hosts: all
  become: yes
  tasks:
    - name: Ensure config is set
      lineinfile:
```

```
        path: /etc/myapp.conf
        regexp: '^enabled='
        line: 'enabled=true'

####################################
# 14. Wait for Port
####################################
- name: Wait for port 80 to be open
  hosts: all
  become: yes
  tasks:
    - name: Wait for HTTP
      wait_for:
        port: 80
        delay: 10
        timeout: 300

####################################
# 15. Reboot Host
####################################
- name: Reboot server
  hosts: all
  become: yes
  tasks:
    - name: Reboot system
      reboot:
        reboot_timeout: 300

####################################
# 16. Use Template (Jinja2)
####################################
- name: Use NGINX config template
  hosts: all
  become: yes
  tasks:
    - name: Deploy NGINX config
      template:
        src: templates/nginx.conf.j2
        dest: /etc/nginx/nginx.conf

####################################
# 17. Use Variables
####################################
- name: Use variables in tasks
  hosts: all
  vars:
    app_user: devops
  tasks:
```

```yaml
    - name: Create user with variable
      user:
        name: "{{ app_user }}"

###################################
# 18. Register Output
###################################
- name: Capture command output
  hosts: all
  tasks:
    - name: Get disk usage
      command: df -h
      register: disk_output

    - name: Show output
      debug:
        var: disk_output.stdout

###################################
# 19. Use Conditionals
###################################
- name: Install for RedHat only
  hosts: all
  tasks:
    - name: Install httpd
      yum:
        name: httpd
        state: present
      when: ansible_os_family == 'RedHat'

###################################
# 20. Use Loop (with_items)
###################################
- name: Install multiple packages
  hosts: all
  become: yes
  tasks:
    - name: Install common tools
      yum:
        name: "{{ item }}"
        state: present
      with_items:
        - git
        - unzip
        - curl

###################################
# 21. Use Handlers
```

```yaml
####################################
- name: Reload nginx using handler
  hosts: all
  become: yes
  tasks:
    - name: Update NGINX conf
      copy:
        src: nginx.conf
        dest: /etc/nginx/nginx.conf
      notify: restart nginx

  handlers:
    - name: restart nginx
      service:
        name: nginx
        state: restarted

####################################
# 22. Include Task File
####################################
- name: Include tasks from file
  hosts: all
  become: yes
  tasks:
    - include_tasks: tasks/setup.yml

####################################
# 23. Include Role
####################################
- name: Use webserver role
  hosts: all
  become: yes
  roles:
    - webserver

####################################
# 24. Set Fact
####################################
- name: Use set_fact
  hosts: all
  tasks:
    - name: Define custom fact
      set_fact:
        app_dir: /opt/app

    - name: Use custom fact
      file:
        path: "{{ app_dir }}"
```

```
        state: directory

###################################
# 25. Assert Statement
###################################
- name: Use assert to validate facts
  hosts: all
  tasks:
    - name: Check OS
      assert:
        that:
          - ansible_os_family == 'RedHat'
        fail_msg: "This playbook only supports RedHat family OS"

###################################
# 26. Manage Docker Container
###################################
- name: Run Docker container
  hosts: docker_hosts
  become: yes
  tasks:
    - name: Run nginx container
      community.docker.docker_container:
        name: nginx
        image: nginx:latest
        ports:
          - "80:80"

###################################
# 27. Git Clone Repository
###################################
- name: Clone from Git
  hosts: all
  become: yes
  tasks:
    - name: Clone repo
      git:
        repo: https://github.com/example/app.git
        dest: /opt/app

###################################
# 28. Download S3 File
###################################
- name: Get WAR file from S3
  hosts: all
  become: yes
  tasks:
    - name: Download WAR from S3
```

```
    aws_s3:
      bucket: app-deploy
      object: myapp.war
      dest: /opt/tomcat/webapps/myapp.war
      mode: get
```

Ansible Programmatic Commands and core concepts with Examples

```
###################################
# 1. Use 'when' Condition Based on Variable
###################################
- name: Install Apache only on production
  hosts: all
  vars:
    env: production
  tasks:
    - name: Install Apache if prod
      yum:
        name: httpd
        state: present
      when: env == 'production'

###################################
# 2. Loop With Dictionary (Using with_dict)
###################################
- name: Create users with home directories
  hosts: all
  become: yes
  tasks:
    - name: Add users with custom homes
      user:
        name: "{{ item.key }}"
        home: "{{ item.value }}"
      with_dict:
        alice: /home/alice
        bob: /home/bob

###################################
# 3. Loop with List of Dictionaries
###################################
- name: Add multiple users with properties
  hosts: all
  become: yes
  vars:
    users:
      - { name: alice, uid: 1001 }
```

```
      - { name: bob, uid: 1002 }
  tasks:
    - name: Add user
      user:
        name: "{{ item.name }}"
        uid: "{{ item.uid }}"
      loop: "{{ users }}"

###################################
# 4. Use Default Jinja2 Filter
###################################
- name: Use default variable fallback
  hosts: all
  tasks:
    - name: Set environment with default
      debug:
        msg: "Environment is {{ env | default('dev') }}"

###################################
# 5. Combine Lists
###################################
- name: Merge two lists
  hosts: all
  vars:
    list1: [1, 2]
    list2: [3, 4]
  tasks:
    - name: Show merged list
      debug:
        msg: "{{ list1 + list2 }}"

###################################
# 6. Iterate with Index
###################################
- name: Loop with index
  hosts: all
  tasks:
    - name: Print indexed values
      debug:
        msg: "Index {{ item.0 }}: {{ item.1 }}"
      loop: "{{ [0, 1, 2] | zip(['apple', 'banana', 'cherry']) }}"

###################################
# 7. Use Pause With Prompt
###################################
- name: Wait for confirmation
  hosts: localhost
  tasks:
```

```yaml
    - pause:
        prompt: "Press ENTER to continue deployment"

####################################
# 8. Run Command with changed_when
####################################
- name: Custom change detection
  hosts: all
  tasks:
    - name: Echo command
      command: echo "done"
      changed_when: false

####################################
# 9. Fail With Message
####################################
- name: Fail if condition not met
  hosts: all
  tasks:
    - name: Check disk space
      shell: df -h / | tail -1 | awk '{print $5}' | sed 's/%//'
      register: disk

    - name: Fail if disk > 80%
      fail:
        msg: "Disk usage is above threshold"
      when: disk.stdout|int > 80

####################################
# 10. Dynamic Includes with Conditional
####################################
- name: Conditionally include playbook
  hosts: all
  tasks:
    - include_tasks: redhat.yml
      when: ansible_os_family == 'RedHat'

####################################
# 11. Create Nested Loop
####################################
- name: Matrix loop
  hosts: all
  tasks:
    - name: Print matrix
      debug:
        msg: "{{ item.0 }} - {{ item.1 }}"
      loop: "{{ ['x', 'y'] | product(['1', '2']) }}"
```

```yaml
####################################
# 12. Random UUID Generation
####################################
- name: Generate UUID
  hosts: localhost
  tasks:
    - name: Show UUID
      debug:
        msg: "{{ lookup('community.general.uuid') }}"

####################################
# 13. Set Complex Fact
####################################
- name: Set dictionary fact
  hosts: all
  tasks:
    - set_fact:
        app_config:
          port: 8080
          debug: true

####################################
# 14. Filter List (selectattr, map)
####################################
- name: Filter enabled services
  hosts: localhost
  vars:
    services:
      - { name: nginx, enabled: true }
      - { name: mysql, enabled: false }
  tasks:
    - debug:
        msg: "{{ services | selectattr('enabled') | map(attribute='name') |
list }}"

####################################
# 15. Lookup File Content
####################################
- name: Read secret from file
  hosts: localhost
  tasks:
    - name: Read file
      debug:
        msg: "{{ lookup('file', 'secrets.txt') }}"
```

Ansible role-based examples with full **folder structures and source code** — ideal for real-world DevOps/Cloud automation.

☑ Example 1: Role to Install Apache (httpd)

📁 Folder Structure

```
apache-role-example/
├── inventory.ini
├── site.yml
└── roles/
    └── apache/
        ├── tasks/
        │   └── main.yml
        ├── handlers/
        │   └── main.yml
        ├── templates/
        │   └── index.html.j2
        ├── vars/
        │   └── main.yml
        └── defaults/
            └── main.yml
```

🔹 **inventory.ini**

```
◇  inventory.ini
[web]
192.168.1.10
```

🔹 **site.yml**

```
◇  site.yml
---
- name: Configure Apache web server
  hosts: web
  become: yes
  roles:
    - apache
```

🔹 **roles/apache/tasks/main.yml**

```
---
```

```
- name: Install httpd
  yum:
    name: httpd
    state: present

- name: Deploy homepage
  template:
    src: index.html.j2
    dest: /var/www/html/index.html
  notify: restart apache

- name: Start and enable Apache
  service:
    name: httpd
    state: started
    enabled: yes
```

- **roles/apache/handlers/main.yml**

```
---
- name: restart apache
  service:
    name: httpd
    state: restarted
```

- **roles/apache/templates/index.html.j2**

```
<!DOCTYPE html>
<html>
<head><title>Apache Server</title></head>
<body>
<h1>Welcome to {{ ansible_hostname }}</h1>
</body>
</html>
```

- **roles/apache/vars/main.yml**

```
---
# (Optional role variables can go here)
```

- **roles/apache/defaults/main.yml**

```
---
# Default variables can go here
```

🧪 **Run Either Role-Based Playbook**

```
ansible-playbook -i inventory.ini site.yml
```

☑ Example 2: Role to Create Multiple Users

📁 **Folder Structure**

```
user-role-example/
├── inventory.ini
├── site.yml
├── vars/
│   └── users.yml
└── roles/
    └── create_users/
        └── tasks/
            └── main.yml
```

🔹 **inventory.ini**

```
[all]
localhost
```

🔹 **site.yml**

```
---
- name: Create system users
  hosts: all
  become: yes
  vars_files:
    - vars/users.yml
  roles:
    - create_users
```

🔹 **vars/users.yml**

```
---
users:
  - name: dev1
    shell: /bin/bash
  - name: qa1
    shell: /bin/zsh
```

- **roles/create_users/tasks/main.yml**

```
---
- name: Add users from list
  user:
    name: "{{ item.name }}"
    shell: "{{ item.shell }}"
    state: present
```

Run Either Role-Based Playbook

```
ansible-playbook -i inventory.ini site.yml
```

Two existing roles (e.g., nginx and user-role) stored in an outer folder like roles/, you can call them easily in your playbook using the roles: directive.

☑ **Folder Structure Example**

```
my-ansible-project/
├── inventory.ini
├── site.yml                # <--- Main playbook
└── roles/                  # <--- Outer folder
    ├── nginx/              # Role 1
    │   └── tasks/main.yml
    └── user-role/          # Role 2
        └── tasks/main.yml
```

- **inventory.ini**

```
[web]
192.168.1.10

[all]
localhost
```

- **site.yml (Playbook Calling Both Roles)**

```
---
- name: Configure NGINX and create users
  hosts: all
```

```
  become: yes

  roles:
    - nginx        # Calls role 1
    - user-role    # Calls role 2
```

- **Notes:**

  - The roles/ directory should be **at the same level as the playbook**, or you can define it in ansible.cfg:

```
[defaults]
roles_path = ./roles
```

  - Role names (e.g., nginx, user-role) must match folder names inside the roles/ directory.
  - Each role should have a standard structure:

```
tasks/
handlers/
templates/
vars/
defaults/
```

  - ---

- **Run It**

```
ansible-playbook -i inventory.ini site.yml
```