Ansible Introduction

Ansible is a **tool to automate tasks** on servers (install software, configure files, restart services) **without agents**.

👉 You write instructions → Ansible runs them on many machines.

Understanding YAML

YAML is a **human-readable format** used by Ansible.

Rules:

- Indentation matters
- Use : for key/value
- Use for lists

```
name: Install nginx
state: present
packages:
- nginx
- git
```

Ansible Inventory

Inventory is a **list of servers** Ansible manages.

```
web1 ansible_host=192.168.1.10
web2 ansible_host=192.168.1.11
```

Grouping and Parent-Child Relationships

You can group servers.

```
[web]
web1
web2

[db]
db1

[prod:children]
web
db
```

👉 prod contains both web and db.

Ansible Variables

Variables store values you can reuse.

```
app_name: nginx
```

Use it:

```
name: Install {{ app_name }}
```

Variable Types

Common types:

```
string: "hello"
number: 80
boolean: true
list:
 - nginx
 - git
dict:
 port: 80
 user: admin
```

👉 ***Note:*** A **list** is an ordered collection of items where each item is identified by its position (index) starting at zero, Items start with a **dash and a space** (-).

A **dictionary** is an unordered collection of **key-value pairs,** Uses a **colon and a space** (`key: value`).

Registering Variables and Variable Precedence

Register saves task output into a variable.

```
- command: uptime
  register: uptime_result
```

Use it:

```
- debug:
    msg: "{{ uptime_result.stdout }}"
```

Precedence (simplified):

Extra vars (`-e`) > Play vars > Inventory vars > Role defaults

---

Variable Scoping

Where a variable is visible.

- **Global**: everywhere
- **Play-level**: inside a play
- **Task-level**: inside one task

```
vars:
  env: prod
```

Magic Variables

Built-in variables provided by Ansible.

Examples:

```
inventory_hostname
groups
hostvars
- debug:
    msg: "{{ inventory_hostname }}"
```

Ansible Facts

Facts are **system information** collected automatically.

```
- debug:
    msg: "{{ ansible_os_family }}"
```

Example facts:

- OS
- IP address
- Memory
- CPU

Ansible Playbooks

Playbooks define **what to run and where**.

```
- hosts: web
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
```

Verifying Playbooks

Check syntax:

```
ansible-playbook playbook.yml --syntax-check
```

Dry run (no changes):

```
ansible-playbook playbook.yml --check
```

Ansible-lint

Tool to **check best practices and errors**.

```
ansible-lint playbook.yml
```

It warns about:

- Bad naming
- Deprecated modules
- Unsafe tasks

Ansible Conditionals

Run tasks **only if a condition is true**.

```
- name: Install nginx on Ubuntu
  apt:
    name: nginx
  when: ansible_os_family == "Debian"
```

Conditionals Based on Facts, Variables, Reuse

Using facts:

```
when: ansible_memtotal_mb > 1024
```

Using variables:

```
when: install_nginx == true
```

Reusable logic:

```
when: env == "prod"
```

Ansible Loops

Run a task multiple times.

```
- name: Install packages
  apt:
    name: "{{ item }}"
  loop:
   - nginx
   - git
   - curl
```

Ansible Modules

Modules are **units of work** (install, copy, service).

Examples:

- apt
- yum
- copy
- service

```
- service:
  name: nginx
  state: started
```

Introduction to Handlers

Handlers run **only when notified,** Tasks that only run **if something changed**.

```
- name: Copy config
  copy:
    src: nginx.conf
    dest: /etc/nginx/nginx.conf
  notify: Restart nginx

handlers:
  - name: Restart nginx
    service:
      name: nginx
      state: restarted
```

Ansible Roles

Roles organize playbooks into folders.

Structure:

```
roles/
  web/
    tasks/
    handlers/
    templates/
    vars/
```

Use role:

```
- hosts: web
  roles:
   - web
```

Ansible Collections

Collections bundle **roles + modules + plugins**.

Install:

```
ansible-galaxy collection install community.general
```

Use:

```
community.general.htpasswd:
```

Templating

Templates create files dynamically using variables.

```
- template:
  src: app.conf.j2
  dest: /etc/app.conf
```

Jinja2 Templates for Dynamic Configs

Template file (`app.conf.j2`):

```
server_name {{ inventory_hostname }};
port {{ app_port }};
```

Variables:

```
app_port:8080
```

Where does this file live?

Example structure:

```
project/
├── playbook.yml
├── templates/
│   └── app.conf.j2
```

Playbook: how the template is "called"

This is the **important part you're missing** 👇

```
- hosts: web
  vars:
    app_port: 8080
  tasks:
    - name: Create config file
      template:
      src: app.conf.j2
      dest: /etc/app.conf
```

What actually happens (step-by-step)

1. Ansible opens `app.conf.j2`
2. It sees:
3. `{{inventory_hostname }}`
4. `{{app_port }}`
5. It looks for values:
   - `inventory_hostname` → built-in variable (like `web1`)
   - `app_port` → from `vars`
6. It **replaces them**
7. It writes a normal file (NO Jinja inside) to `/etc/app.conf`

Final file on the server (real output)

If host = `web1`, file becomes:

```
server_name web1
port 8080
```