
🟦 What Are Loops in Ansible?

A **loop** allows you to **repeat a task multiple times** with different input values without writing multiple tasks.

- Instead of writing the same task for Ubuntu and CentOS separately, you can **loop over a list of items**.
- Each loop iteration sets a variable (`item`) that can be used inside the task.

Loops help make your playbooks:

- **Shorter**
- **Cleaner**
- **Easier to maintain**
- **Scalable** (easy to add more OS types or packages in the future)

🟦 How Loops Work in Your Playbook

Loop Example from Your Playbook:

```
loop:
  - { package: "apache2", distribution: "Ubuntu" }
  - { package: "httpd",  distribution: "CentOS" }
```

- **First iteration (`item`)**:
    - `item.package = apache2`
    - `item.distribution = Ubuntu`
- **Second iteration (`item`)**:
    - `item.package = httpd`
    - `item.distribution = CentOS`

During each iteration, Ansible checks the `when` condition to decide if the task should run.

🟦 Why Loops Are Useful

1. **Reduce Repetition**
    - Instead of 6 tasks for install/start/enable, you write **3 tasks only**.
2. **Dynamic Tasks**
    - The same task works for multiple OS types and packages.
3. **Easy to Maintain**
    - To add another OS (like Fedora), you just add another dictionary in the loop:

```
- { package: "httpd", distribution: "Fedora" }
```

4. **Works With `when` Conditions**
    - Each loop item can have its own conditions to ensure tasks only run where appropriate.

🟦 Loops vs Traditional Tasks

Traditional Tasks	Loops
Separate tasks for Ubuntu/CentOS	Single task with loop over OS/packages
Harder to maintain and scale	Easy to maintain and scale
More lines in playbook	Fewer lines, cleaner syntax
Risk of typos in repeated tasks	Less repetitive, reduces errors

---

🟦 In Plain English

Loops are like saying:

> “Run this task for each item in this list, but only if the OS matches the item.”

For example:

- Ubuntu → install apache2
- CentOS → install httpd

Instead of writing two separate tasks, loops do it automatically.

✅ Playbook With Loops Added (Clean + Short Version)

```
---
- name: Update all servers
  hosts: all
  become: yes

  tasks:
    - name: Ensure web server is installed
      package:
        name: "{{ item.package }}"
        state: present
      when: ansible_distribution == item.distribution
      loop:
        - { package: "apache2", distribution: "Ubuntu" }
        - { package: "httpd",  distribution: "CentOS" }

    - name: Ensure web server is started
      service:
        name: "{{ item.service }}"
        state: started
      when: ansible_distribution == item.distribution
      loop:
        - { service: "apache2", distribution: "Ubuntu" }
        - { service: "httpd",  distribution: "CentOS" }

    - name: Ensure web server is enabled
      service:
        name: "{{ item.service }}"
        enabled: true
      when: ansible_distribution == item.distribution
      loop:
        - { service: "apache2", distribution: "Ubuntu" }
        - { service: "httpd",  distribution: "CentOS" }
```

🟦 Explanation of the Loop Version

Here is how the loop simplifies your original 6 tasks:

🟥 1. Why We Use Loops

In the original playbook, you wrote:

- 3 tasks for Ubuntu
- 3 tasks for CentOS

But the tasks are the same (install, start, enable) — only **package name** and **OS** change.

Loops allow us to **reuse one task** for both OS types.

🟩 2. How the Loop Works

Example loop:

```
loop:
  - { package: "apache2", distribution: "Ubuntu" }
  - { package: "httpd",  distribution: "CentOS" }
```

The loop runs the task **twice**, once per item.

On each loop item:

- `item.package` becomes either `apache2` or `httpd`
- `item.distribution` becomes either `Ubuntu` or `CentOS`

🟦 3. How the when Condition Works With Loops

This line:

when: ansible_distribution == item.distribution

Ensures:

✔ Ubuntu servers only run the apache2 loop item

✔ CentOS servers only run the httpd loop item

Ansible tests each loop item separately.

🟩 4. Example of What Happens on Ubuntu

Loop iteration 1:

- item.package = apache2
- item.distribution = Ubuntu
- Condition TRUE → task runs

Loop iteration 2:

- item.package = httpd
- item.distribution = CentOS
- Condition FALSE → task skipped

🟥 5. Example on CentOS

Loop iteration 1:

- apache2 for Ubuntu → SKIPPED

Loop iteration 2:

- httpd for CentOS → RUNS

🟦 6. Final Outcome

Using loops:

- The install logic is written once
- The start logic is written once
- The enable logic is written once

But they automatically adjust for:

- Different OS (Ubuntu/CentOS)
- Different packages (apache2/httpd)
- Different services
