Ansible Install

To install Ansible on Ubuntu 24.04, the recommended approach is to add the official Ansible PPA (Personal Package Archive) to get the most recent stable version and then install it using the apt package manager. This ensures access to the latest features compared to the default Ubuntu repositories.

1. Update your system:

```
$ sudo apt update
$ sudo apt upgrade -y
```

2. Install the prerequisite package `software-properties-common` which is required to manage PPAs:

```
$ sudo apt install software-properties-common -y
```

3. Add the official Ansible PPA repository:

```
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
```

4. Install Ansible:

```
sudo apt install ansible -y
```

5. Verify the installation by checking the Ansible version:

```
$ ansible --version
```

Ansible Configuration

1. Create two Ubuntu instances on AWS EC2. Tag Names - node1 and node2

```
terraform {
 required_providers {
   aws = {
     source  = "hashicorp/aws"
     version = "~> 4.16"
   }
 }

 required_version = ">= 1.2.0"
}
```

```
provider "aws" {
 region = "ap-south-1"
}

resource "aws_instance" "node1" {
 #ami           = "ami-0af9569868786b23a"
 ami = "ami-0f918f7e67a3323f0"
 instance_type = "t2.micro"
 key_name      = "yogeshpem"
 tags = {
  Name = "node1"
 }
}

resource "aws_instance" "node2" {
 ami           = "ami-0af9569868786b23a"
 instance_type = "t2.micro"
 key_name      = "yogeshpem"
 tags = {
  Name = "node2"
 }
```

2. For SSH authentication Password less

    1. Create ssh
       ```
       $ ssh-keygen -o
       ```
    2. Copy public key to node machines
       ```
       $ ssh-copy-id -f "-o IdentityFile yogeshpem.pem" ubuntu@ec2-13-201-128-219.ap-south-1.compute.amazonaws.com
       ```
    3. Login into machine
       ```
       $ ssh ubuntu@ec2-13-201-128-219.ap-south-1.compute.amazonaws.com
       ```

3. Ansible Connect

How Ansible Controller knows that it has to work with Node1 and Node2 machines.

Answer - From INVENTORY

INVENTORY is just a file that can be written in 2 formats.

1. inventory.ini [Most Popular Approach Followed]
2. inventory.yaml

Inventory.ini can be created in any directory

create a directory named ansible

1. $ mkdir ansible
2. $ cd ansible
3. sudo vim inventory.ini

Add these lines:

```
$ sudo vim inventory.ini
```

ubuntu@16.171.32.129
ubuntu@13.61.25.34

4. Let's run some command on both the managed node1 and node2 from ansible
   - Practical 1 : Check connection

```
$ ansible -i inventory.ini  -m ping all
```

   - Practical 2: Let's create a directory named "test" on both the nodes.

```
$ ansible  -i inventory.ini -m shell -a "mkdir test" all
```

   - Practical 3: we want to group nodes under group like APP and DB

```
$ sudo vim inventory.ini
[app]
ubuntu@16.171.32.129
[db]
ubuntu@13.61.25.34
$ ansible -i inventory.ini  -m ping db
```

Ansible Playbook

https://docs.ansible.com/ansible/latest/getting_started/get_started_playbook.html

Ansible Playbook is a list of plays written in YAML

- Practical 1: Running Ping and echo on Nodes

https://docs.ansible.com/ansible/latest/getting_started/get_started_playbook.html

- Practical 2 : Installing Apache and copying index.html on Nodes

Create 3 files : apacheplaybook.yaml index.html inventory.ini

inventory.ini

[app] ubuntu@13.127.188.134

[db] ubuntu@13.127.188.134

index.html

Copy index.html (shared by me in the current directory)

```
<!DOCTYPE html>
<html>
<head>
 <title>Welcome to My Website</title>
 <meta charset="UTF-8">
</head>
<body>
 <h1>Hello, World!</h1>
 <p>This is the homepage.</p>
</body>
</html>
```

apacheplaybook.yaml

```
- hosts: all
  become: true
  tasks:
    - name: Install apache httpd
      ansible.builtin.apt:
```

```
    name: apache2
    state: present
    update_cache: yes

- name: Copy file with owner and permissions
  ansible.builtin.copy:
    src: index.html
    dest: /var/www/html/index.html
    owner: root
    group: root
    mode: '0644'
```

Run the playbook command

```
$ ansible-playbook -i inventory.ini first-playbook.yaml
```

Go to EC2 Node on AWS

1. connect to the instance
2. Run these commands to check if apache is installed and index.html is copied

```
$ sudo ps -ef | grep apache2
$ sudo systemctl status apache2
$ ls /var/www/html
```

3. Add a security group to the Node HTTP   80
4. Send request to ec2 instance public ip from browser http://your-public-ip/

You should see the index.html page returned from apache2 server

Roles in Ansible

Roles in Ansible are organizational units that group related automation code—such as tasks, variables, handlers, templates, and files—into reusable, modular, and maintainable packages.

What are Roles?

- Roles are a method for structuring Ansible automation into self-contained bundles, designed to improve reuse, collaboration, and scalability across playbooks and projects.
- Each role follows a standardized directory structure, enabling separate management of configuration aspects (e.g., tasks, files, variables).

Purpose and Benefits

- **Modularity:** Roles split configuration into logical, reusable pieces, allowing small playbooks to be combined as needed.
- **Reusability:** Same role can be applied across multiple playbooks and projects, preventing duplication.
- **Maintainability:** Organized code is easier to manage, update, and share.
- **Team Collaboration:** Teams can work on separate roles independently, enhancing parallel development.
- **Scalability:** Large projects become easier to scale due to clear separation and organization.

Role Directory Structure

A typical Ansible role has this structure:

Directory	Purpose
tasks/	Core automation actions (main.yml)
handlers/	Restart/reload tasks triggered by other actions

templates/	Jinja2 dynamic configuration files
files/	Static files for deployment
vars/	Role-specific variables
defaults/	Default variable values
meta/	Metadata and dependencies [1][4]

Example Use Case

Suppose a deployment needs a **web server**, **database server**, and general settings. Separate roles (e.g., “web-server”, “database-server”, “common”) can handle configuration for each independently and be reused for different servers or environments.

Roles are referenced in playbooks and can be conditionally included or parameterized to meet varying automation requirements.

How to create role? using - ansible-galaxy command

```
$ ansible-galaxy role init test

$ cd test

$ ls -ltr
```

You will see the structure created by role.

Explore the folders in the structure.

Practical 1 – Convert apache2 playbook into role.

The original Apache2 playbook can be converted into an Ansible **role** by following the standard role directory structure and separating the tasks accordingly.

Step-by-Step Conversion

1. Role Initialization

Use the command:

```
$ ansible-galaxy init apache
```

This will create a directory named `apache` with subfolders like `tasks`, `handlers`, `files`, etc.

2. Define Tasks (`tasks/main.yml`)

Edit `apache/tasks/main.yml` to include the Apache install and file copy tasks:

```
- name: Install apache httpd
  ansible.builtin.apt:
    name: apache2
    state: present
    update_cache: yes

- name: Copy file with owner and permissions
  ansible.builtin.copy:
    src: index.html
    dest: /var/www/html/index.html
    owner: root
    group: root
    mode: '0644'
```

Place the `index.html` file inside the `apache/files/` directory for the role to access.

3. Directory Structure Example

After initialization and populating the files, the role's structure will look like:

```
roles/
 └── apache/
    ├── tasks/
    │    └── main.yml
    ├── files/
    │    └── index.html
    ├── handlers/
    │    └── main.yml
    ├── defaults/
    │    └── main.yml
    ├── vars/
    │    └── main.yml
    └── meta/
        └── main.yml
```

Only `tasks` and `files` are required for this simple use-case.

4. Using the Role in a Playbook

Reference the role in your main playbook:

```
---
- hosts: webservers
  become: yes
  roles:
    - apache
```

This imports and executes the tasks defined in the role.

5. Run the ansible-playbook command -

```
$ ansible-playbook -i inventory.ini apacheindex.yaml

$ ansible-galaxy role -h       //help command
```