📜 Ansible Setup and Playbook Execution — Step-by-Step Documentation

🖥️ Environment

- **Control Node (Master):** Ubuntu EC2 instance with Ansible installed
- **Managed Nodes (Workers):** Ubuntu EC2 instances (worker1 & worker2)
- **Goal:** Use Ansible to install **Nginx** on both worker nodes using **group-level variables**

⚙️ Step 1: Update and Install Ansible

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

✅ This installs Ansible and its dependencies.

To verify:

```
ansible --version
```

📂 Step 2: Create Ansible Directory Structure

Create the main Ansible directory and subfolders:

```
sudo mkdir -p /etc/ansible/inventory
cd /etc/ansible/inventory
```

Create supporting folders for variables:

```
mkdir group_vars host_vars
```

Place your private SSH key in `/etc/ansible` for secure access to target nodes:

```
sudo nano /etc/ansible/octkey.pem
sudo chmod 600 /etc/ansible/octkey.pem
```

🧩 Step 3: Create Inventory File

File: `/etc/ansible/inventory/host.yaml`

```yaml
all:
  vars:
    ansible_become: true

  children:
    frontend:
      hosts:
        worker1:
          ansible_host: 43.205.207.55
          ansible_user: ubuntu
          ansible_ssh_private_key_file: /etc/ansible/octkey.pem
          ansible_ssh_common_args: '-o StrictHostKeyChecking=no'

        worker2:
          ansible_host: 43.205.215.86
          ansible_user: ubuntu
          ansible_ssh_private_key_file: /etc/ansible/octkey.pem
          ansible_ssh_common_args: '-o StrictHostKeyChecking=no'
```

✅ This defines:

- A group named **frontend**
- Two hosts: **worker1** and **worker2**
- SSH details for connection
- `ansible_become: true` enables privilege escalation (sudo)

📃 Step 4: Create Group-Level Variable

File: `/etc/ansible/inventory/group_vars/frontend.yaml`

```yaml
app_name: nginx
```

✅ This variable (`app_name`) applies to all hosts in the **frontend** group.

📄 Step 5: (Optional) Host-Level Variables

If you want individual host-specific variables, you can add files in:

```
/etc/ansible/inventory/host_vars/
```

Example: `/etc/ansible/inventory/host_vars/worker1.yaml`

```yaml
app_user: ubuntu
```

📜 Step 6: Create Playbook

File: `/etc/ansible/inventory/playbook.yaml`

```yaml
---
- name: nginx install
  hosts: frontend
  become: yes

  tasks:

    - name: Update the apt package index (for Debian/Ubuntu)
      ansible.builtin.apt:
        update_cache: yes

    - name: Install Nginx on Debian/Ubuntu
      ansible.builtin.apt:
        name: "{{ app_name }}"
        state: present
      when: ansible_os_family == "Debian"

    - name: Show app user
      debug:
        msg: "app will run as {{ app_user | default('ubuntu') }}"
```

✅ Key points:

- Installs the package defined in `app_name` (from group_vars)
- Uses `become: yes` for sudo
- `app_user` is displayed if defined; otherwise defaults to `ubuntu`

▶️ Step 7: Run the Playbook

Execute the playbook with:

```
ansible-playbook -i /etc/ansible/inventory/host.yaml
/etc/ansible/inventory/playbook.yaml
```

Expected output:

```
TASK [Update the apt package index] ***
ok: [worker1]
ok: [worker2]

TASK [Install Nginx on Debian/Ubuntu] ***
changed: [worker1]
changed: [worker2]

TASK [Show app user] ***
ok: [worker1] => {
    "msg": "app will run as ubuntu"
}
ok: [worker2] => {
    "msg": "app will run as ubuntu"
}
```

📁 Final Directory Structure

```
/etc/ansible/
├── octkey.pem
└── inventory/
    ├── host.yaml
    ├── playbook.yaml
    ├── group_vars/
    │   └── frontend.yaml
    └── host_vars/
        └── worker1.yaml  (optional)
```

✅ Validation

Step	Description	Status
Ansible installation	Installed successfully	✅
Directory structure	Created under `/etc/ansible`	✅
Inventory file	YAML structure valid	✅
Variable files	Group/host vars read correctly	✅
Playbook execution	Installs Nginx on both servers	✅