**LAB4: Implementing variables in playbook using template and survey :**

* cat third.yml

---

- name: "Deploy {{ pkgname }} webserver"

hosts: m1

become: yes

vars:

pkgname: apache2

service: apache2

desired\_state: present # 'present' or 'absent' (via survey)

listen\_port: 81 # Port number (via survey)

tasks:

- name: Update apt package index

ansible.builtin.apt:

update\_cache: yes

- name: Ensure {{ pkgname }} is in desired state {{ desired\_state }}

ansible.builtin.apt:

name: "{{ pkgname }}"

state: "{{ desired\_state }}"

- name: Update Apache listening port (only if installing)

ansible.builtin.lineinfile:

path: /etc/apache2/ports.conf

regexp: "^Listen "

line: "Listen {{ listen\_port }}"

when: desired\_state == "present"

notify:

- restart\_apache

- name: Update virtual host port (only if installing)

ansible.builtin.lineinfile:

path: /etc/apache2/sites-available/000-default.conf

regexp: '^<VirtualHost \\*:'

line: "<VirtualHost \*:{{ listen\_port }}>"

when: desired\_state == "present"

notify:

- restart\_apache

- name: Deploy Apache homepage (only if installing)

ansible.builtin.copy:

dest: /var/www/html/index.html

content: "<h1>This is a demo from Raman Khanna!</h1>"

when: desired\_state == "present"

notify:

- restart\_apache

handlers:

- name: restart\_apache

ansible.builtin.service:

name: "{{ service }}"

enabled: yes

state: restarted

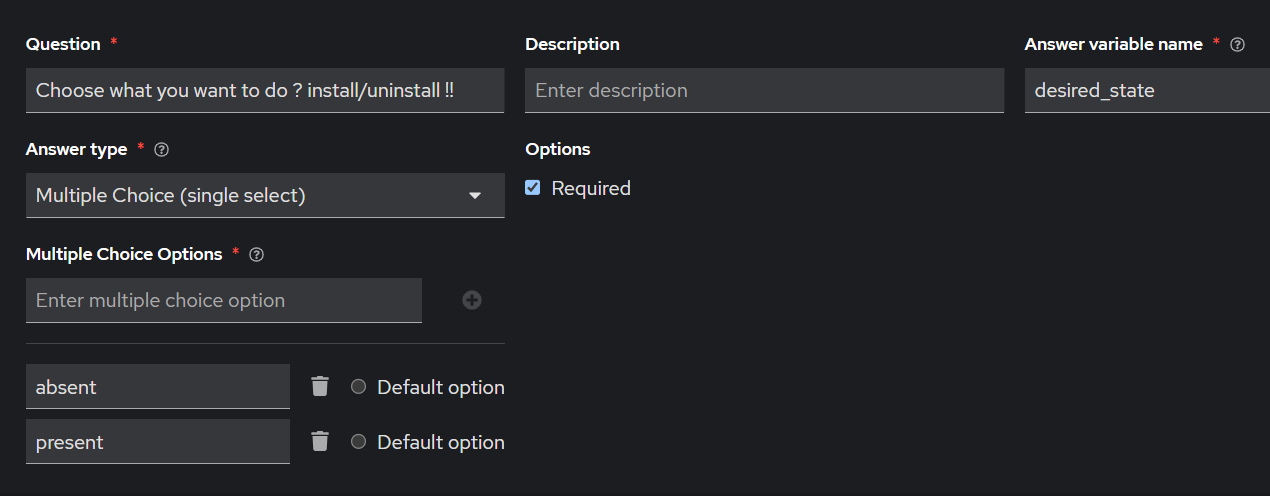
-- sync the project with the third.yml

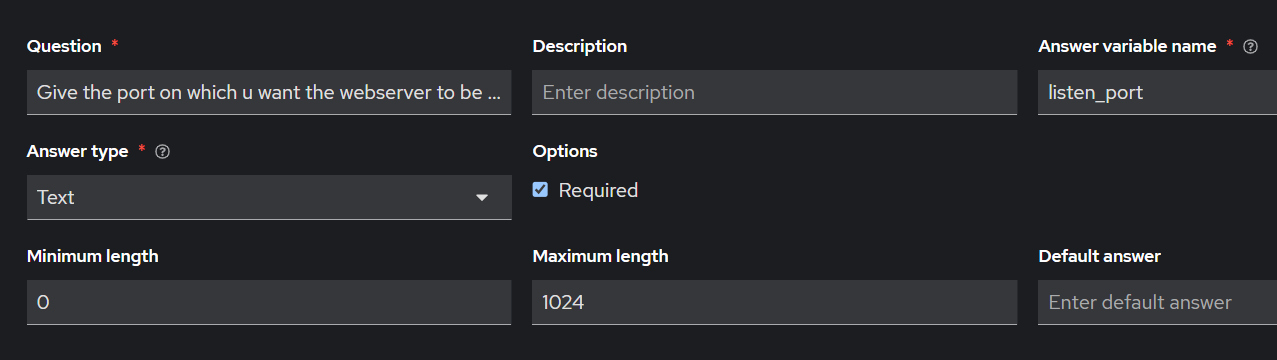
-- update ips on raman-static-inv

--- create a template :

    updating the necessry things : playbook third.yml

    --add the survey for port and state only...





    --- check it first

    -- run it

    -- browse the public ip of m1:82

**LAB5 : Grouping Hosts and Defining Variables**

**Step 1: Create a New Inventory**

1. Navigate to **Inventories** → **Add** → **Inventory**
2. Fill in:
   * **Name**: Ubuntu Inventory
   * **Organization**: <your-org>
   * Click **Save**

**Step 2: Create Groups in the Inventory**

1. Open the Ubuntu Inventory
2. Go to the **Groups** tab → **Add Group**
   * Group 1:
     + **Name**: web
   * Group 2:
     + **Name**: db

**Step 3: Add Hosts to Groups**

**Add Hosts to web group:**

1. Click on **web** group → **Hosts** tab → **Add Host**
   * **Host Name**: ubuntu-node1
   * add ansible\_host: 15.229.78.109 in variables )

**Add Hosts to db group:**

1. Click on **db** group → **Hosts** tab → **Add Host**
   * **Host Name**: ubuntu-node2
   * add ansible\_host: 192.168.1.102in variables )

**Step 4: Define Group Variables**

**For web group:**

1. Go to web group → **Variables** tab
2. Add the following in **YAML** format:

role: webserver

pkg\_name: nginx

service\_name: nginx

**For db group:**

1. Go to db group → **Variables** tab
2. Add:

role: database

pkg\_name: mysql-server

service\_name: mysql

**Step 5: Define Host-level Variable (Override Example)**

Let’s override the pkg\_name for ubuntu-node2:

1. Go to ubuntu-node2 under db group → **Variables** tab
2. Add:

pkg\_name: mariadb-server

**Step 6: Create a Project with a Git Repository**

1. Navigate to **Projects** → **Add**
2. Fill in:
   * **Name**: Grouping Lab Project
   * **SCM Type**: Git
   * **SCM URL**: https://github.com/<your-org>/aap-lab-grouping.git  
     *(You can host a repo with the below playbook)*
   * **SCM Branch**: main
   * **Execution Environment**: Default

**Step 7: Create the Playbook**

Create a file called install\_service.yml in your Git repository:

---

- name: Install and start services based on group vars

hosts: all

#hosts: web

#hosts:db

tasks:

- name: Install required package

apt:

name: "{{ pkg\_name }}"

state: present

update\_cache: yes

- name: Start and enable the service

service:

name: "{{ service\_name }}"

state: started

enabled: yes

- name: Print role info

debug:

msg: "This host is configured as a {{ role }}"

Push this playbook to GitHub.

**Step 8: Create a Credential for Ubuntu Nodes**

1. Go to **Credentials** → **Add**
2. Fill in:
   * **Name**: Ubuntu SSH Credential
   * **Credential Type**: Machine
   * **Username**: ubuntu
   * **Private Key**: Paste your SSH private key
   * **Privilege Escalation**: Checked (sudo)

**Step 9: Create a Job Template**

1. Go to **Job Templates** → **Add**
2. Fill in:
   * **Name**: Install Role Based Services
   * **Job Type**: Run
   * **Inventory**: Ubuntu Inventory
   * **Project**: Grouping Lab Project
   * **Playbook**: install\_service.yml
   * **Credentials**: Ubuntu SSH Credential
   * Click **Save**

**Step 10: Launch and Observe Output**

1. Launch the job template
2. Watch the output:
   * ubuntu-node1 installs **nginx** and sets role as webserver
   * ubuntu-node2 installs **mariadb-server** (host var overrides group var) and role as database

**🔍 Validation & Learning Points**

| **Feature** | **Demonstrated By** |
| --- | --- |
| Host grouping | Hosts assigned to web and db groups |
| Group variables | pkg\_name, role, service\_name at group level |
| Host variable override | pkg\_name: mariadb-server for ubuntu-node2 |
| Variable usage | Variables used in the playbook with {{ var }} syntax |

| **Precedence Level** | **Variable Source** | **Priority** |
| --- | --- | --- |
| 1 | defaults/ in roles | LOW |
| 2 | inventory variables |  |
| 3 | group\_vars/all |  |
| 4 | group\_vars/group\_name |  |
| 5 | host\_vars/hostname |  |
| 6 | vars: in playbook | ✅ HIGH |
| 7 | vars\_files: in playbook | ✅ HIGHER |
| 8 | include\_vars (inline vars loading) | ✅ HIGHEST (among manual includes) |
| 9 | CLI with -e (extra vars) | 🛑 HIGHEST of ALL |
|  |  |  |
|  |  |  |
|  |  |  |

**LAB 6 : VARIABLE DECLARATION USING VARS.YAML**

**- hosts: web,db**

**vars\_files:**

**- vars.yml**

**tasks:**

**- name: Update apt package index**

**apt:**

**update\_cache: yes**

**- name: Install {{ package }} package**

**package:**

**name: "{{ package }}"**

**state: "{{ desired\_state }}"**

**- name: Configure hostname and IP in webpage**

**template:**

**#copy:**

**src: index.html**

**dest: /var/www/html/index.html**

**when: desired\_state == "present"**

**notify: Restart {{ package }} service**

**handlers:**

**- name: Restart {{ package }} service**

**service:**

**name: "{{ package }}"**

**state: restarted**

* + - **vars.yml :**

**package: apache2**

**desired\_state: present**

**web\_content: |**

**<h1>This is a demo for Apache Webserver!</h1>**

**<p>Hostname: {{ ansible\_hostname }}</p>**

**<p>IP Address: {{ ansible\_default\_ipv4.address }}</p>**

**<p>OperatingSystem : {{ ansible\_os\_family }}</p>**

* + - **index.html :**

**{{ web\_content }}**

* + - **now add package: mariadb-server in groups and launch to checkif it overrides the default vars.yml**
    - **it will not override cz group/vars and host/vars have low priority than vars\_file**

| **Precedence Level** | **Variable Source** | **Priority** |
| --- | --- | --- |
| 1 | defaults/ in roles | LOW |
| 2 | inventory variables |  |
| 3 | group\_vars/all |  |
| 4 | group\_vars/group\_name |  |
| 5 | host\_vars/hostname |  |
| 6 | vars: in playbook | ✅ HIGH |
| 7 | vars\_files: in playbook | ✅ HIGHER |
| 8 | include\_vars (inline vars loading) | ✅ HIGHEST (among manual includes) |
| 9 | CLI with -e (extra vars) | 🛑 HIGHEST of ALL |
|  |  |  |
|  |  |  |

**Lab 7: Variable Precedence using group\_vars and host\_vars in AAP**

**🎯 Objective:**

Demonstrate how variables defined in group\_vars and host\_vars override each other, and how they affect playbook execution in an inventory synced from a GitHub project.

**🛠️ Lab Setup**

**1. Prepare GitHub Repo Structure**

Push the following structure to your GitHub repo (e.g., aap-lab4-demo):

\_13\_\_test\_project/

├── group\_vars/

│ ├── web

│ └── db

├── host\_vars/

│ ├── m1

│ └── m2

├── inv

├── var3.yml

├── index.html

└── README.md

**📁 File Details**

**inv (Inventory File)**

[web]

m1 ansible\_host=177.71.159.199

[web:vars]

web\_content=<h1>This is a demo</h1><p>Host: {{ ansible\_hostname }}</p>

[db]

m2 ansible\_host=52.67.104.173

**group\_vars/web**

pkg: apache2

**group\_vars/db**

pkg:

* + - mariadb-server
    - apache2

**host\_vars/m1**

location: Mumbai DC

role: WebServer

env: production

**host\_vars/m2**

location: São Paulo DC

role: DBServer

env: staging

**index.html**

<html>

<head><title>Demo Page</title></head>

<body>

<h1>This is a demo</h1>

<p>Host: {{ ansible\_hostname }}</p>

<p>Role: {{ role }}</p>

<p>Location: {{ location }}</p>

<p>Environment: {{ env }}</p>

</body>

</html>

**var3.yml (Playbook)**

- hosts: all

tasks:

- name: Update apt package index

apt:

update\_cache: yes

when: ansible\_os\_family == "Debian"

ignore\_errors: true

- name: Install {{ pkg }} package

package:

name: "{{ pkg }}"

state: present

ignore\_errors: true

- name: Configure hostname and IP in webpage

template:

src: index.html

dest: /var/www/html/index.html

notify: Restart service

ignore\_errors: true

- name: Show assigned role

debug:

msg: "Host {{ inventory\_hostname }} has role: {{ role }} and is located in {{ location }} (env: {{ env }})"

handlers:

- name: Restart service

service:

name: "{{ pkg }}"

state: restarted

**📦 AAP UI Configuration**

**1. Import GitHub Project**

* Go to **Resources → Projects**
* Create a new Project named aap-lab4-project
* Git URL: https://github.com/<your-user>/aap-lab4-demo.git
* Sync it.

**2. Create Inventory**

* Navigate to **Resources → Inventories**
* Create a new Inventory: lab4-inventory
* Add a Source:
  + Source: **Sourced from Project**
  + Project: aap-lab4-project
  + Inventory File: inv
  + Sync

**3. Create Template**

* Go to **Templates**
* Add a new Job Template:
  + Name: Install & Configure with group\_vars/host\_vars
  + Inventory: lab4-inventory
  + Project: aap-lab4-project
  + Playbook: var3.yml

**🚀 Execution & Verification**

1. **Launch the Job Template**
2. **Observe output:**
   * Correct packages are installed per host group (apache2, mariadb-server)
   * HTML file created using values from host\_vars
   * debug task outputs role, location, and env per host

**✅ Expected Output Snippets:**

* TASK [Install apache2 package] on m1
* TASK [Install mariadb-server package] on m2
* TASK [Show assigned role]:

"msg": "Host m1 has role: WebServer and is located in Mumbai DC (env: production)"

"msg": "Host m2 has role: DBServer and is located in São Paulo DC (env: staging)"

**🧪 Experiment Ideas**

* Override pkg in host\_vars/m1 and observe precedence.
* Add a variable with the same name in var3.yml using vars: and see how it overrides group/host variables.
* Create a new host and test fallback to group\_vars.