The **precedence of Ansible variables** determines which variable value is applied when variables with the same name are defined in multiple locations. Ansible follows a well-defined hierarchy, with variables closer to the task execution having higher precedence.

Here’s the variable precedence order, from **highest to lowest**:

**1. Extra Variables (-e)**

* Variables passed via the ansible-playbook command using the -e or --extra-vars option have the highest precedence.
* Example:
* **ansible-playbook playbook.yml -e "var1=value1"**

**2. Task Variables**

* Variables defined directly in a specific task within a playbook have the next highest precedence.
* Example:
* - name: Use a specific variable
* debug:
* msg: "The value is {{ var1 }}"
* vars:
* var1: "task-specific-value"

**3. Block Variables**

* Variables defined for an entire block of tasks.
* Example:
* - block:
* - name: Print block variable
* debug:
* msg: "Block variable value is {{ var1 }}"
* vars:
* var1: "block-value"

**4. Role Defaults**

* Variables defined in defaults/main.yml within a role. These are the least prioritized role-specific variables and are meant to provide default values that can be overridden.
* Example:
* # roles/role\_name/defaults/main.yml
* var1: "default-role-value"

**5. Play Variables**

* Variables defined at the play level within a playbook.
* Example:
* - hosts: all
* vars:
* var1: "play-level-value"

**6. Role Variables**

* Variables defined in vars/main.yml in a role. These variables are specific to the role and cannot be overridden by other sources except task and extra variables.
* Example:
* # roles/role\_name/vars/main.yml
* var1: "role-specific-value"

**7. Include Variables**

* Variables from files or tasks that are explicitly included in a playbook or role using include\_vars or vars\_files.
* Example:
* - include\_vars: vars.yml

**8. Set\_Facts Variables**

* Variables set dynamically during a playbook run using the set\_fact module. These are temporary variables available during the playbook's execution.
* Example:
* - name: Set a fact
* set\_fact:
* var1: "dynamic-value"

**9. Registered Variables**

* Variables that store the output of a task using the register keyword.
* Example:
* - name: Run a command
* command: whoami
* register: result
* - name: Print the result
* debug:
* var: result.stdout

**10. Host Variables**

* Variables defined for specific hosts in the host\_vars/ directory or inventory.
* Example in host\_vars/server1.yml:
* var1: "host-specific-value"

**11. Group Variables**

* Variables defined for groups of hosts in the group\_vars/ directory or inventory.
* Example in group\_vars/webservers.yml:
* var1: "group-specific-value"

**12. Inventory Variables**

* Variables defined directly in the inventory file (static or dynamic).
* Example in a static inventory file:
* [webservers]
* server1 ansible\_host=192.168.1.10 var1="inventory-value"

**13. Facts (Gathered Facts)**

* Facts are automatically gathered by Ansible using the setup module and include system information (e.g., IP address, OS family, etc.).
* Example:
* ansible\_facts['os\_family']

**14. Role Defaults**

* Variables defined in a role's defaults/main.yml file. These are the lowest-priority variables within a role, providing default values that are easy to override.
* Example:
* # roles/role\_name/defaults/main.yml
* var1: "default-value"

**Variable Precedence Summary Table**

| **Priority** | **Source** |
| --- | --- |
| 1 (Highest) | Extra Variables (-e) |
| 2 | Task Variables |
| 3 | Block Variables |
| 4 | Role Variables (vars/main.yml) |
| 5 | Play Variables |
| 6 | Include Variables |
| 7 | Set\_Facts Variables |
| 8 | Registered Variables |
| 9 | Host Variables (host\_vars/) |
| 10 | Group Variables (group\_vars/) |
| 11 | Inventory Variables |
| 12 | Gathered Facts |
| 13 (Lowest) | Role Defaults (defaults/main.yml) |

**Best Practices**

1. **Keep Variable Names Consistent**: Avoid duplicate variable names to reduce confusion and conflicts.
2. **Use Role Defaults for Defaults**: Place default values in defaults/main.yml and override them as needed.
3. **Limit Scope**: Use variables with the most appropriate scope (e.g., host-specific variables in host\_vars).
4. **Encrypt Sensitive Variables**: Use **Ansible Vault** for sensitive data.

By following the precedence hierarchy and best practices, you can structure your Ansible variables effectively and avoid unexpected overrides.

What is Roles

Example – is to install httpd

1. Dnf update
2. Epel-release
3. Install httpd
4. Update config file of httpd
5. Service – restart ,and then enable