**MANAGING ANSIBLE VARAIBLES**

Ansible supports variables that can be used to store values that can then be reused throughout files in an Ansible project.

**Valid and Invalid Variable Names**

* Web\_server web server
* Remote\_file remote.file
* File\_1 1file
* File1 1file
* Remote\_server\_1 remoteserver$1

**Variable Precedence**

* Global Scope: Variable set from the cmd line or Ansible Configuration
* Play Scope: Varaibles set in the play and related structures
* Host scope [ Inventory]: Varaible set on host groups and indicidual hosts by the inventory, fact gathering or registered tasks.

If the same variable name is defined at more than one level, the level with the highest precedence wins.

Variables defined by the inventory are overridden by variables defined by the playbook, which are overridden by varaibles defined on the command line.

**Defining Variables in Playbook**

It can be defined in multiple ways.

**Method 1 [INSIDE PLAYBOOK]**: One of the method is to place a variable in a **vars** block at the begining of a playbook



* name: variable defining inside the play

hosts: all

**vars:**

**user: venkat**

**home: /home/joe**

**package: httpd**

tasks:

**Method 2 [OUTSIDE PLAYBOOK]**:Defining variable in external file and called the file inside the playbook using **vars\_file**



* **hosts: all**

**vars\_file:**

**- vars.yml**

**Method 3 [DEFINE INSIDE INVENTORY]**: **HOST VARAIBLES &&& GROUP VARIABLES**

* Inventory varaibles that apply directly to hosts fall into 2 categories: host variables apply to a specific host, and group variables apply to all hosts in a hosts grp or in a group of hosts grps. Host varaibles takes precendence over group variable but varaibles defined by a playbook take precendence over both, which can override by parameter passing in cli.
* One way to define host variables and grp varaibles is to do it directly in the inventory file. This is an older approach not preferred, but u may still encoiunter it. Some disadv of this method is it makes inventory file more difficult to work with, it mixes information abot hosts and variables in the same file

**Host variable INI format**

localhost ansible\_connection=local

other1.example.com ansible\_connection=ssh ansible\_user=myuser

other2.example.com ansible\_connection=ssh ansible\_user=myotheruser

**Group variable INI format**

[atlanta]

host1

host2

[atlanta:vars]

ntp\_server=ntp.atlanta.example.com

proxy=proxy.atlanta.example.com

**Using Directories to populate host and group variables**

The preferred approach to defining varaibles for host and groups is to create twi dorectories, group\_vars and host\_vars, in the same working directory as the inventory file or directory. These directories contain files defining grp and host variables.

Project directory

**├── ansible.cfg**

**├── group\_vars ------------------------------directory**

**│   └── MERGE.yml ----------------------------varaible file for group MERGE**

**├── host\_vars ---------------------------------directory**

**│   └── servera.example.com ---------------------------varaible file for the server.example.com**

**├── inventory**

**├── play.yml**

**└── variable.yml**

**Overrinding varaibles from the command line**

**#ansible-playbook main.yml –e “package=httpd”**

**How to call the variable in playbook**

Variable are referenced by placing the variable name in double quote curly braces **{{}}**

Example :

- name: The {{ firewall\_pkg }} service sarted and enable ------**quotation not mandatory since it started in between**

- name: “ {{ firewall\_pkg }} service started and enabled ”---**quotation mandatory since it started started first**

service:

name: "{{ firewall\_pkg }}" -----------**quotation mandatory**

state: started

**Capturing Command output with REGISTERED varaibles**

Admin can use the register statement to capture the outpit of a command. The output is saved into a teporary variable that can be used in the playbook for either debugging purposes or to achieve something else, such as a particular configuration based on a commands’s output.

****