chapter : 04
Managing Variables and Facts

Variables can be used to store values that can then be reused throughout files/project.

Variables names must start with letter, and they can only contain letters, numbers and underscore.

variables defined in command line

↑

variables defined in playbook

↑

variables defined by inventory.

To mention variables in playbook :

```
- Hosts: all
  vars :
     user : joe
     home : /home/joe
```

To mention variables in files & mention file in playbook

```
- hosts : all
  vars_files :
      - file1
```

$ from file1

```
→ user : joe
  home : /home/joe
```

To call out / use the variable "{{ }}" is used.

$ Host variables & Group variable.

This is an older approach and not prefered.
Two category : 1) Host variables
2) Group variables.

ex: 1) Host variables.
in inventory.

```
[server_a]
node1        ansible_user = joe.
```

2) Group variables
in inventory.

```
[server_a]
node1
[server_b]
node2
[serverc : children]
 server_a
 server_b
[server_c : vars]
 ansible_user = joe
```

Preffered practice is to create two directories 'group_vars' 'host_vars' in /home/user/ansible dir.
& create files in them to mention variables.

```
$ cat /home/user/ansible/group_vars/server_c
→ package: httpd

$ cat /home/user/ansible/host_vars/node1
→ package: httpd
```

But these can be overriden by playbook vars & CLI

```
$ ansible-playbook package.yml -e "package=apache2"
```

To mention as array

```
users:
    bjones:
        first_name: Bob
```

Mention it as
users.bjones.first_name
⇒ Bob
OR
users['bjones']['first_name']
⇒ Bob

The register statement is used to capture the output of the task & store it in a variable which can be useful to give reference later.

```
- yum:
    name: httpd
    state: present
  register: result_of_yum
- debug: var= result_of_yum
    var: result_of_yum
- debug:
    msg: "Apache is installed"
  when: result_of_yum.rc == 0
```

Sensitive data such as passwords, API keys etc should not be stored in plain text for security measures.

- Ansible Vault is used to encrypt & decrypt any structured data file used by ansible.

```
$ ansible-vault
        create
        edit
        encrypt/ decrypt
        view
```

Files are encrypted used AES256

$ ansible-vault create new_file.yml
→ Creates new encryption file & opens 'vi' editor. by default (as env EDITOR = vi).
the default 'vi' can be changed by (export EDITOR = nano)

$ ansible-vault create --vault-password-file=pass new_file.yml
→ creates new encrypted file & uses password mentioned in 'pass' file in the directory.

$ ansible-vault view new-file.yml
→ view the file data after entering password

$ ansible-vault edit new_file.yml
→ allows to edit the file data after entering password (rewrites)

$ ansible-vault encrypt existing_file.yml
→ encrypts the already existing file.

$ ansible-vault encrypt old_file.yml --output=new_file.yml
→ saves the encrypted file with the new name

$ ansible-vault decrypt existing_file.yml
→ decrypts the file permanently.
(can use --output to save under different name)

$ ansible-vault rekey encrypted_file.yml
→ changes the password of the file.

$ ansible-vault rekey --new-vault-password-file=filename encryption_file.yml

If a Playbook is using data from vault file ; encryption password has to be provided while running a playbook

$ ansible-playbook --vault-id @prompt playbook.yml
$ ansible-playbook --ask-vault-pass playbook.yml
→ Asks for password before running the playbook.

$ ansible-playbook --vault-password-file=file playbook.yml
→ 'file' contains password for vault.

— ANSIBLE_VAULT_PASSWORD_FILE env = default location of pass-file.

Preferred way of variables;

```
├ansible.cfg
├ group_vars
│     └ webservers
│          └ ~~node1~~ vars
├ host_vars
      node1
        └ vars
        └ vault
```

— How to assign vault ID = ??.

Ansible facts

Ansible facts are variables that are automatically discovered by ansible on a managed node. It contains host-specific info.

Every play runs the setup module automatically before the first task = Gathering facts

```
- debug:
    var: ansible_facts
```

Ansible facts can be written as:
① ansible_facts['default_ipv4']['address]
② ansible_facts.default_ipv4.address

To call out ansible fact it should be mentioned in '{{ }}' similar to variable

$ ansible node1 -m setup
→ shows all ansible facts of 'node1' machine

If you do not want to gather facts
```
- hosts: all
  gather_facts: no
```

To gather facts at any time.
→ - setup:

Setup module loads facts from files & scripts in each managed host's '/etc/ansible/facts.d' directory all files in this directory should end with '.fact'

Custom.fact vim

```
→ [packages]          — INI style.
  web_package = httpd.
  [Users]
  user 1 = joe
  user 2 = jane.
```

This can calso be witten in JSON Format

To call out local facts

→ ansible_facts. ansible_local. custom. users. user 1

Magic vaiables

hostvars → variables for managed hosts,

group_names → groups of current managed host.

groups → All groups & hosts in inventory

inventory_hostname → Hostname of managed host as in inventory