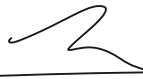
Bonjour Tout le Monde



Machine Physique / Virtuelle → Dev : Virtual Box, VMware Workst, Player, Hyper V

Cloud → Public : - AWS, - Azure, - GCP

Conteneurs → Virtualisation d'Applications → Prod : - Hyper V, - VMware ESXi

Mono host

Ansible →

Cluster

Ansible → Totalement

CAS NORMAL

Asuv Iaas → Infrastructure → VM → Vous gérez le contenu de vo VM.

Anib Paas → Platform → Hebergement - Web, - BAD, - Conteneurs

Saas → Software → Solution

Toute façon

Besoin d'une Infra → VM

conteneurs.

Infra

locale ⇐ VAGRANT (Hashicorp) "façon"

Distante ⇙ DataCenter ⇘ Cloud

Vagrant (local)

Vagrantfile
pb.yaml
} vagrant up

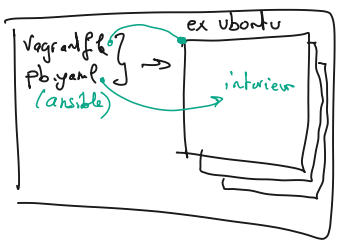
↓

① Créer la VM ("coque extérieure")

② Injecter l'exécution de Ansible pour "l'intérieur de la coque"

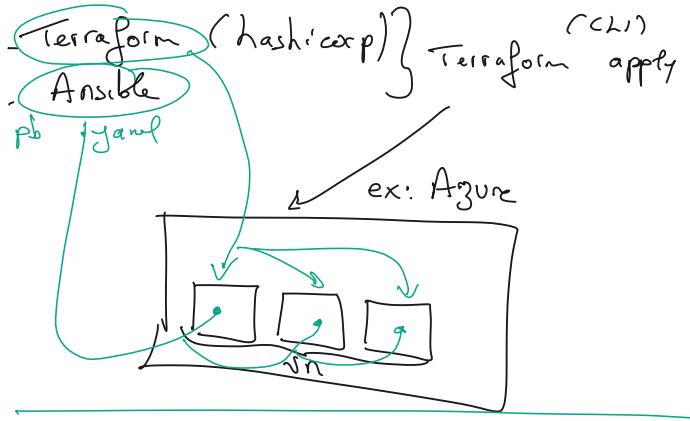
---

host



} n VMs en masse.

2) Pilotage de cluster ou DataCenter

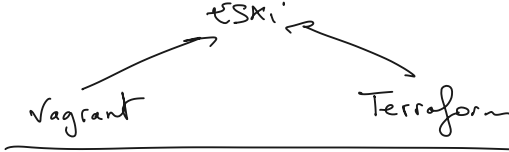


Azure
- <LI
- Portail
- Models déclaratifs

AWS ...
- CLI
- Portail
- Mod. déclaratifs

Terraform

- pb.yaml



Ansible → local
→ distant

Création d'infra : possible via Ansible

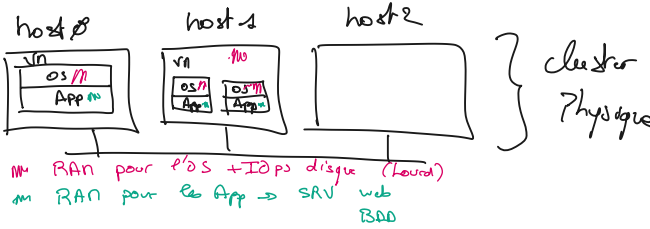
- exécution locale
  - soit encapsuler la CLI du fournisseur
    - AWS, Azure, ESXi
  - soit utiliser des plugins du fournisseur
    (par ex. VM AWS sous Ansible)
- Et enchaîner de l'Ansible distant pour l'intérieur

AZ → CLI Azure
ansible
pb_iaas.yaml → crée l'infra
- enchaîne sur
pb_os.yaml
→ Installer des paquets
Mettre de conf à jour
etc ...

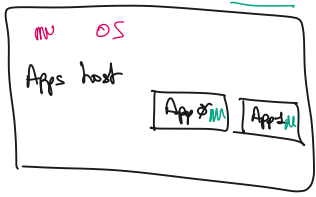
Virtualisation d'OS → VMWare ESX
- HyperV

↳ Lourde

Conteneurisation (Virtualisation d'Application)



Conteneurs → Recommandée



App0 } Conteneurs
App1 } - App Isolées

Docker → Mono Host

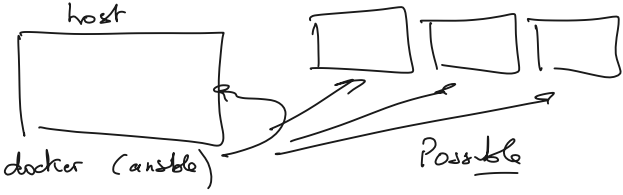


image par image → app par app

docker-compose → 1 groupe infra | Code

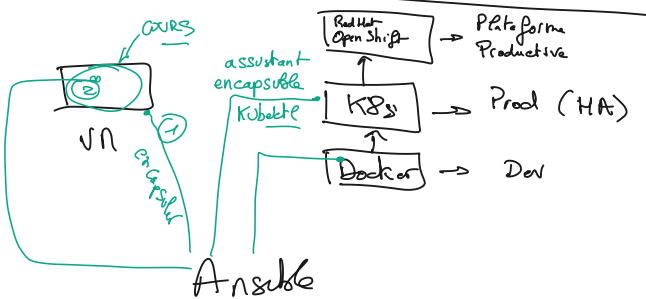
ex: front web + 1 BDD

Cluster → Prod

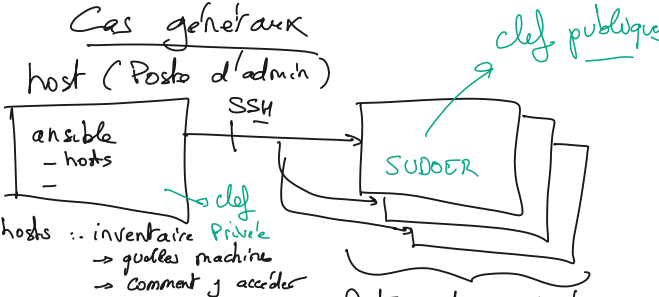
- Swarm → assez facile, accessible (limité en prod)

- Kubernetes (K8s) → + puissant utilisé en Prod
  Apprentissage + complexe à mettre en œuvre

- (Docker EE) → n'est pas utilisé



Cas généraux



hosts :. inventaire Privée
→ quelles machine
→ comment y accéder

- playbook
→ déclaratif des étapes

SECU : - fichier de clefs SSH
- port spécifique ?
- mots de passe

Modèle Impératif

- Scripting, ligne par ligne
→ Tests → Variables
→ Boucle → Exceptions

- Scripts -
Admin via Bash
↳ Impératif

⚠ → n Scripts pour le n Cas de figure
Pbm maintenance etc...

Modèle Déclaratif

- On écrit ~~un script~~ un fichier déclaratif format YAML -
- Colonné
- Recommandé : 2 espaces.
~~TAB~~ : interdit !

- Un orchestrateur (comme un Workflow) va Transformer l'état voulu en des étapes à effectuer

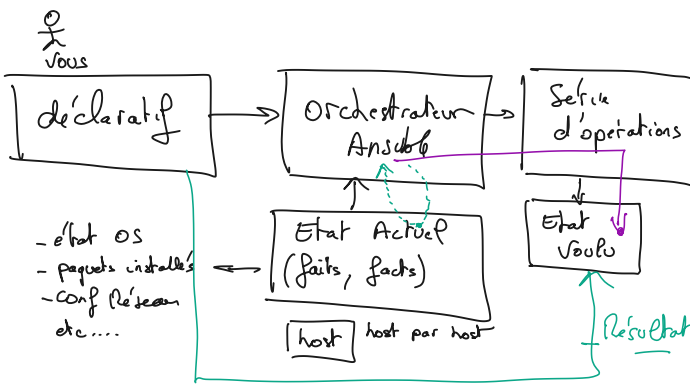
Idempotent

↳ l'exécution X fois engendre
Toujours le même résultat,
Sans erreur _

ex : - echo "ok" > file.txt
- apt update
apt install

Non Idempotent ex echo "ok" >> file.txt

ANSIBLE → VOUS DEVREZ
AGIR vous l'idempotence



•https://docs.ansible.com/ansible/latest/collections/index_module.html

[targets]

localhost
ansible_connection=local

snap install code --classic
code hosts.yaml

sudo apt install sshpass

cat hosts
srv1 ansible_user=stagiaire ansible_password="Pa$$w0rd"

ansible -i hosts.yaml -m ping all

hosts.yaml :

```
all:
  hosts:
    srv1:
    srv2:
    srv3:
    srv4:
  vars:
    ansible_user: stagiaire
    ansible_password: Pa$$w0rd
```

```
---
- hosts: all
  tasks:
    - ping: data="Hello
ping1"
    - ping:
        data: "Hello ping2"
```

Ex1 : mise en oeuvre, objectif :

ansible -i hosts -m ping all

[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host srv1 should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release
will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature will be
removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.

srv1 | SUCCESS => {

```
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

en Yaml : (mais il est nécessaire d'effectuer une connexion ssh manuelle préalable uniquement sur srv1)

```
all:
  hosts:
    srv1:
    srv2:
    srv3:
    srv4:
  vars:
    ansible_user: stagiaire
    ansible_password: Pa$$w0rd
```

ansible -i hosts.yaml -m ping all (sans connexion ssh préalable sur srv2) :

srv2 | FAILED! => {
    "msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
[DEPRECATION WARNING]: Distribution Ubuntu 20.04 on host srv1 should use /usr/bin/python3, but is using /usr/bin/python for backward compatibility with prior Ansible releases. A future Ansible release
will default to using the discovered platform python for this host. See https://docs.ansible.com/ansible/2.9/reference_appendices/interpreter_discovery.html for more information. This feature will be
removed in version 2.12. Deprecation warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
srv1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}

Solution host key checking :

Yaml :

```
all:
  hosts:
    srv1:
    srv2:
    srv3:
    srv4:
  vars:
    ansible_user: stagiaire
    ansible_password: Pa$$w0rd
    ansible_ssh_extra_args: "-o StrictHostKeyChecking=no"
```

INI :

```
srv1 ansible_user=stagiaire  ansible_password="Pa$$w0rd" ansible_ssh_extra_args='-o StrictHostKeyChecking=no'
srv2 ansible_user=stagiaire  ansible_password="Pa$$w0rd" ansible_ssh_extra_args='-o StrictHostKeyChecking=no'
```

ou :

```
srv1
srv2

[all:vars]
ansible_user=stagiaire
ansible_password="Pa$$w0rd"
ansible_ssh_extra_args='-o StrictHostKeyChecking=no'
```

---

le playbook minimum :

```
---
- hosts: all
  tasks:
    - ping:
```

L'évaluation des chaînes de caractères utilise le moteur de Templates Jinja2
https://pypi.org/project/Jinja2/

Appel de fonction :

```
- name: display fact
    debug:
      msg: "{{ansible_facts.cmdline.BOOT_IMAGE|wordcount}}"
```

Référence des fonctions Jinja2:
https://jinja.palletsprojects.com/en/3.0.x/templates/?highlight=filters#list-of-global-functions

Condtionnel s'applique tache par tache
L'evènementiel se met en cache, et ne s'execute qu'une fois à la fin de toutes les taches

Fonctionnement des playbooks

ansible-playbook — variables — pb.yaml

Ordre de Précédence.

inventaire
- variable
  - Par host
  - Par group

Var entête
- Par block
- Par tâche

- Conditions — when: Par condition
- Tests — crées par fichier (présence)
- blocks
- Événements
- ...

Structure du Playbook:

Play : { https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#play

  Task : { https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#task
    Block : { } https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#block
  }
  Role { } https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#role
}

```
---
- hosts: all
  tasks:
    - name: avant le bloc
      debug:
        msg: Hello World
    - name: retirer le fichier
      file:
        path: /tmp/myfile
        state: absent
    - name: controler la présence du fichier
      stat:
        path: /tmp/myfile
      register: myfile_exists
    - name: Faire apparaître un fichier
      file:
        path: /tmp/myfile
        state: touch
      when: not myfile_exists.stat.exists
    - block:
      - name: ma tache de bloc
        debug:
          msg: Hello World
        become: yes
      - name: ma tache de bloc2
        debug:
          msg: Hello World2
      - name: ma tache de bloc3
        debug:
          msg: Hello World2
      become: yes
      when: false
    - name: "après"
      debug:
        msg: Hello World
```

Exercice :

Installez docker, si il ne l'est pas déjà préalablement (avec une condition when)

- Soit via les paquets ubuntu (docker.io)
- option si vous avez le temps : avec la derniere version selon les recommandations d'installation depuis docker.io
- Méthode :
    a. installez docker manuellement
    b. cherchez un fichier qui va apparaître sur lequel vous appuyer pour tester sa présence (sous /etc/...)
    c. Elaborez le playbook avec une installation systématique de docker dans un premier temps.
    d. créez un autre playbook pour supprimer docker systématiquement pour les tests
    e. mettre la recherche du fichier (stat)
    f. ajoutez une condition when à l'installation du paquet

Solution :
Installation :

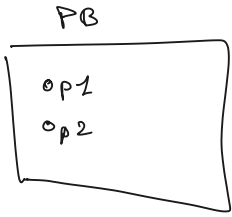
```
---
- name: Playbook qui installe docker
  hosts: all
  tasks:
    - name: controle de l'existence de /usr/bin/docker
      stat:
        path: /usr/bin/docker
      register: dockerexists
    - name: Installer le package Docker.io
      become: true
      apt:
        name: docker.io
        state: present
      when: not dockerexists.stat.exists
```

Désinstallation :

```
---
- name: Playbook qui retire docker
  hosts: all
  tasks:
    - name: Retirer le package Docker.io
      become: true
      apt:
        name: docker.io
        state: absent
```

```
---
- name: Playbook de syntaxe avancée
  hosts: all
  vars:
    users:
      - st1
      - st2
  tasks:
    - name:
      command: /bin/true
      ignore_errors: yes
      notify: mysecondhandler
    - name: commande qui ne fait rien
      command: /bin/true
      # notify appelle le handler
      notify: "Handler qui s'execute à la fin de toutes les autres commandes"
    - name: Force all notified handlers to run at this point, not waiting for normal sync
points
      meta: flush_handlers

    - name: commande qui ne fait rien
      block:
        - name: command
          command: /bin/true
          notify: myhandler
      # notify appelle le handler
      # meilleure pratique:

    - name: commande qui ne fait rien
      command: /bin/true
      # notify appelle le handler

    - name: afficher une boucle
      debug:
        msg: "Itération : {{item}}"
      loop: "{{users}}"

    # - name: test
    #   become: true
    #   block:
    #     - name: test
    #       debug:
    #   loop:
    #     - st1
    #     - st2

    # - name: "Création d'une collection d'utilisateurs"
    #   become: true
    #   block:
    #     - name: création de l'utilisateur
    #       user:
    #         name: "{{item}}"
    #     - name: créer un fichhier par user
    #       command: "touch /home/{{item}}/ok"
    #   with_items:
    #     - st1
    #     - st2
    #     - st3

  handlers:
    - name: "Handler qui s'execute à la fin de toutes les autres commandes"
      listen: myhandler
      debug:
        msg: "Coucou je suis un handler"
    - name: "Second handler"
      listen: mysecondhandler
      debug:
        msg: "Coucou je suis un second handler"
```

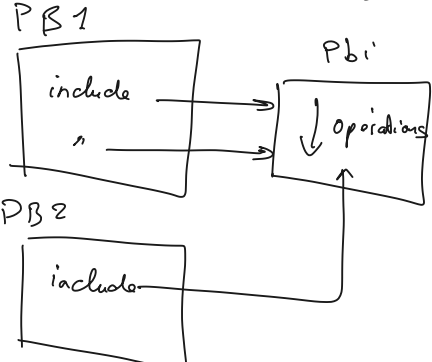
Réutilisation de contenu



op1 = op2
→ ne fait jamais de copier/coller

équivalent aux fonctions, modules, classes.

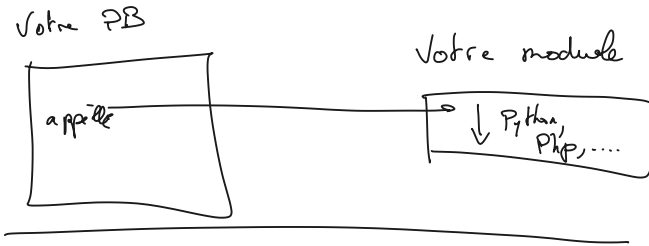
(1)



(2) Besoin très technique, spécifique, actuellement inexistant

- Par exemple : faire une requête dans une API Rest

- Dans une BDD
  - exécuter du code natif -



(3) Rôles

apt
nuget
maven
snap
etc...

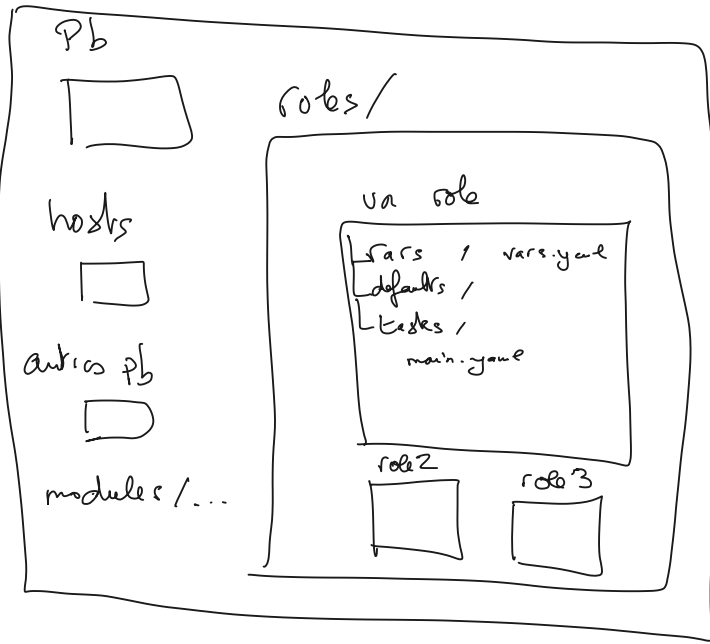
} utile de packages —

→ Réutilisation Communautaire

Galaxy
Hub Ansible de
Rôles

ansible local

→ Télécharger/
Installer
des Rôles
. Réutiliser —




```
stagiaire@master:~/roles$ tree .
.
├── hosts.yaml
└── roles
    └── geerlingguy.git
        ├── defaults
        │   └── main.yml
        ├── LICENSE
        ├── meta
        │   └── main.yml
        ├── molecule
        │   └── default
        │       ├── converge.yml
        │       ├── molecule.yml
        │       └── playbook-source.yml
        ├── README.md
        ├── tasks
        │   ├── install-from-source.yml
        │   └── main.yml
        └── vars
            ├── Debian.yml
            ├── Fedora.yml
            ├── main.yml
            └── RedHat.yml

8 directories, 14 files
```

Variable, paramètres
Possible (remplacable,
Surchargeable)

l'essentiel, le Point d'entrée
de votre rôle

Paramètres statiques, difficil
Par le dev, ne doit pas
changer (usage
GLOBALES interne)

Utilisation des roles :

1. Télécharger le role
    a. ex : ansible-galaxy role install -p roles geerlingguy.git
2. Appeler le/roles : faire un playbook :

```
---
- name : Installer git via le role Geerlingguy.git
  hosts: all
  become: true
  # tasks:
  #   - name: ma tache
  #     command: /bin/true
  roles:
    - geerlingguy.git
```

Exécuter, et valider que git est installé

Contenu de meta :

```
---
dependencies: []

galaxy_info:
  author: geerlingguy
  description: Git version control software
  company: "Midwestern Mac, LLC"
  license: "license (BSD, MIT)"
  min_ansible_version: 2.5
  platforms:
    - name: EL
      versions:
        - all
    - name: Fedora
      versions:
        - all
    - name: Debian
      versions:
        - all
    - name: Ubuntu
      versions:
        - all
  galaxy_tags:
    - development
    - system
    - git
    - vcs
    - source
    - code
```

Exercice :
Créez votre role "Hello World paramétré"

ansible-galaxy role init philippe59710fr.myrole

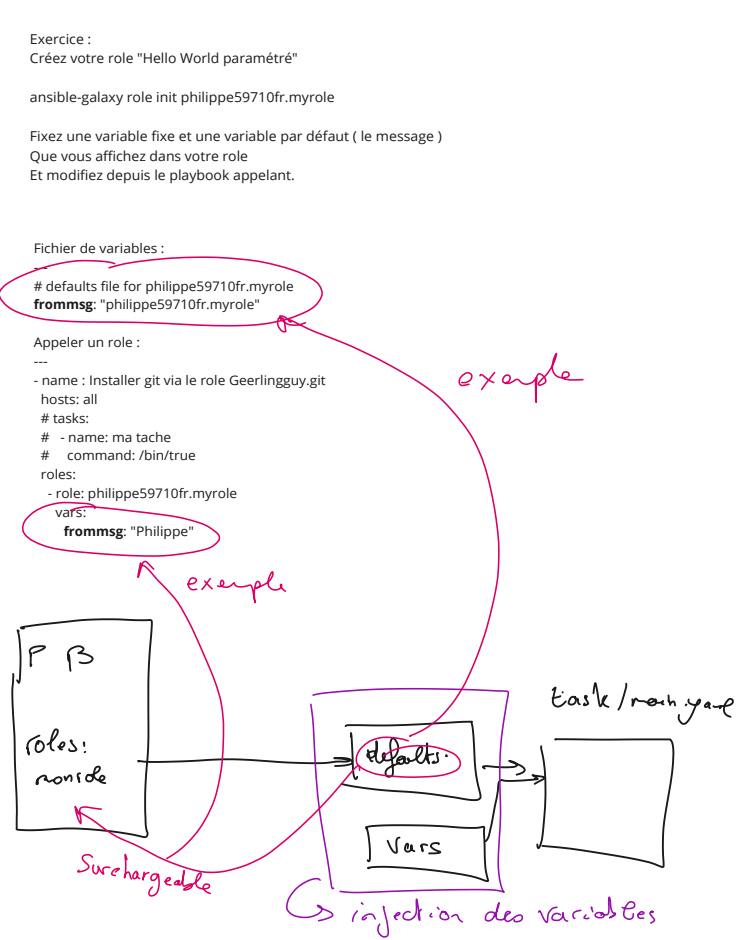
Fixez une variable fixe et une variable par défaut (le message)
Que vous affichez dans votre role
Et modifiez depuis le playbook appelant.

Fichier de variables :

```
---
# defaults file for philippe59710fr.myrole
frommsg: "philippe59710fr.myrole"
```

Appeler un role :

```
---
- name : Installer git via le role Geerlingguy.git
  hosts: all
  # tasks:
  #   - name: ma tache
  #     command: /bin/true
  roles:
    - role: philippe59710fr.myrole
      vars:
        frommsg: "Philippe"
```



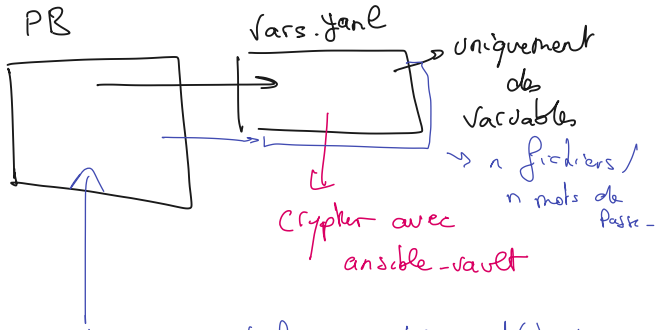
Positionnements possibles des variables (des moins prioritaires, aux plus prioritaires)
Les dernières variables de cette liste, remplacement les valeurs précédentes si préalablement définies :

1. command line values (for example, -umy_user, these are not variables)
2. role defaults (defined in role/defaults/main.yml)
3. inventory file or script group vars
4. inventory group_vars/all
5. playbook group_vars/all
6. inventory group_vars/*
7. playbook group_vars/*
8. inventory file or script host vars
9. inventory host_vars/*
10. playbook host_vars/*
11. host facts / cached set_facts
12. play vars
13. play vars_prompt
14. play vars_files
15. role vars (defined in role/vars/main.yml)
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include_vars
19. set_facts / registered vars
20. role (and include_role) params
21. include params
22. extra vars (for example, -e"user=my_user")(always win precedence)

tasks/main/yaml :

```
---
# tasks file for philippe59710fr.myrole
- name: Hello from philippe59710fr.myrole
  debug:
    msg: "{{hellomsg}} {{frommsg}}"
```

Cheat Sheet DigitalOcean :
https://www.digitalocean.com/community/cheatsheets/how-to-use-ansible-cheat-sheet-guide



executer en spécifiant le / les mot(s) de passe(s) des/du cryptage(s) fournit un mot de passe :

label Texte prompt	@	Prompt nom de fichier

1) Creer un PB, vars utilisés vars dans le PB, contrôler (debug) — mot de passe
2) Crypter le fichier vars
3) Executer avec id@<mot de passe> — SRC de
4) Changer le | mot de passe
Contenu crypté
5) Retirer / n mots de passe -

Inclusion des variables :
https://docs.ansible.com/ansible/latest/collections/ansible/builtin/include_vars_module.html

vars.yaml :

```
---
myvar: "my value"
```

pb.yaml :

```
---
- name: Playbook de mise en oeuvre du vault
  hosts: all
  tasks:
    - name: Chargement des variables
      include_vars: vars.yaml
    - name: Afficher une variable cryptée
      debug:
        var: myvar
```

```
stagiaire@master:~/mots_de_passe$ ansible-vault encrypt --vault-id prompt  vars.yaml
New vault password (default):
Confirm new vault password (default):
Encryption successful
```

vars.yaml :

```
$ANSIBLE_VAULT;1.1;AES256
36626566383736313139626230623135616662366630363239323138333630643266636662363962
34363961636566653330363431616631613439386166323 50a393834333133343131326565643033
39393363396564383364323862353236616462306432383764346238623461636362346439646137
63303362643865363 20a3864316539343139376266663365323331363661613035383962393938 32
34336634316439323935376539356337353062386133336463343030363836323930
```

stagiaire@master:~/mots_de_passe$ ansible-vault view vars.yaml

```
Vault password:
---
myvar: "my value"
```

Execution avec la demande du mot de passe :

```
ansible-playbook -i hosts.yaml pb.yaml --vault-id promp
```

ou vuia un fichier :

```
tansible-playbook -i hosts.yaml pb.yaml --vault-id thepassword.key
```

Cryptage via un fichier de mot de passe :

```
stagiaire@master:~/mots_de_passe$ ansible-vault encrypt --vault-id thepassword.key test.yaml
Encryption successful
stagiaire@master:~/mots_de_passe$ cat thepassword.key
password
```

Pour recrypter :

```
stagiaire@master:~/mots_de_passe$ ansible-vault rekey --vault-id @prompt vars.yaml
Vault password (default):
New Vault password:
Confirm New Vault password:
Rekey successful
```

```
---
- name: Playbook modèle pour l'essentiel des commandes et fonctions
  hosts: all   # le filtre dans l'inventaire est ici
  gather_facts: false
  ignore_unreachable: yes
  vars:
    ansible_python_interpreter: auto  # désactive le warning
    messages:
      - hello
      - world
  vars_prompt:
    - name: your_name
      prompt: "What is your name?"

  tasks:      # cette série de commande n'est qu'un exemple, il peut y en avoir bien
d'autres
    - name: Ping machine
      ping:
      register: pingres
    - name: afficher le résultat d'une commande
      debug:
        var: pingres
    - name: afficher le résultat d'une commande
      debug:
        msg: "{{pingres|upper}}"
    - name: Bloc
      block:
      - name: tache de block
        ping:
        notify: handle_error
        when: true  # condition de block
      - name: Force a failure, that will get caught ( trapped ) by exception handler
        ansible.builtin.command: /bin/false
      rescue:
        - name: Print when errors
          ansible.builtin.debug:
            msg: 'I caught an error, can do stuff here to fix it, :-)'
      always:
        - name: Always do this
          ansible.builtin.debug:
            msg: "This always executes, :-)"
    - name: Lancer le handler une seconde fois
      command: /bin/true
      notify: handle_error
    - name: Forcer l'exécution des handlers maintenant
      meta: flush_handlers

    - name: appel d'un PB n fois avec Items
      include_tasks: per_item.yaml
      loop: "{{messages}}"

    - name: Poursuivre, même en cas d'erreur non gérée
      command: /bin/false
      ignore_errors: yes
      ignore_unreachable: yes

    - name: Forcer une commande Ok en Fail
      command: /bin/true
      failed_when: true
      ignore_errors: yes

    - name: Forcer une commande Ok en Changed
      command: /bin/true
      changed_when: true

    - name: Executer un role n Fois en passant n valeurs différentes de variables
      include_role:
        name: philippe59710fr.myrole
      vars:
        frommsg: "{{item}}"
      with_items:
        - Titi
        - Toto


  handlers:
    - name: handle_error
      ansible.builtin.debug:
        msg: 'This handler runs even on error'
  roles:
    - role: philippe59710fr.myrole
      vars:
        frommsg: "{{your_name}}"
```