TP Ansible

ValT

Pour utiliser Ansible : **$ sudo apt install ansible**

L'objectif du TP va vous paraître un peu bête, mais on pourra l'extrapoler pour faire des choses plus utiles

En gros on va se ssh sur notre propre machine (oui oui c'est possible, même si l'intérêt est limité), et on va pouvoir modifier des choses dessus : afficher ses propriétés, créer un dossier, installer un logiciel.

Le but, c'est de montrer que si ça marche avec votre propre pc, ça peut fonctionner sur n'importe quelle machine

01

Avant de faire du ansible, on va d'abord préparer le terrain : si on veut se ssh avec ansible sur notre machine, il faut déjà vérifier qu'on peut se ssh sans passer par ansible

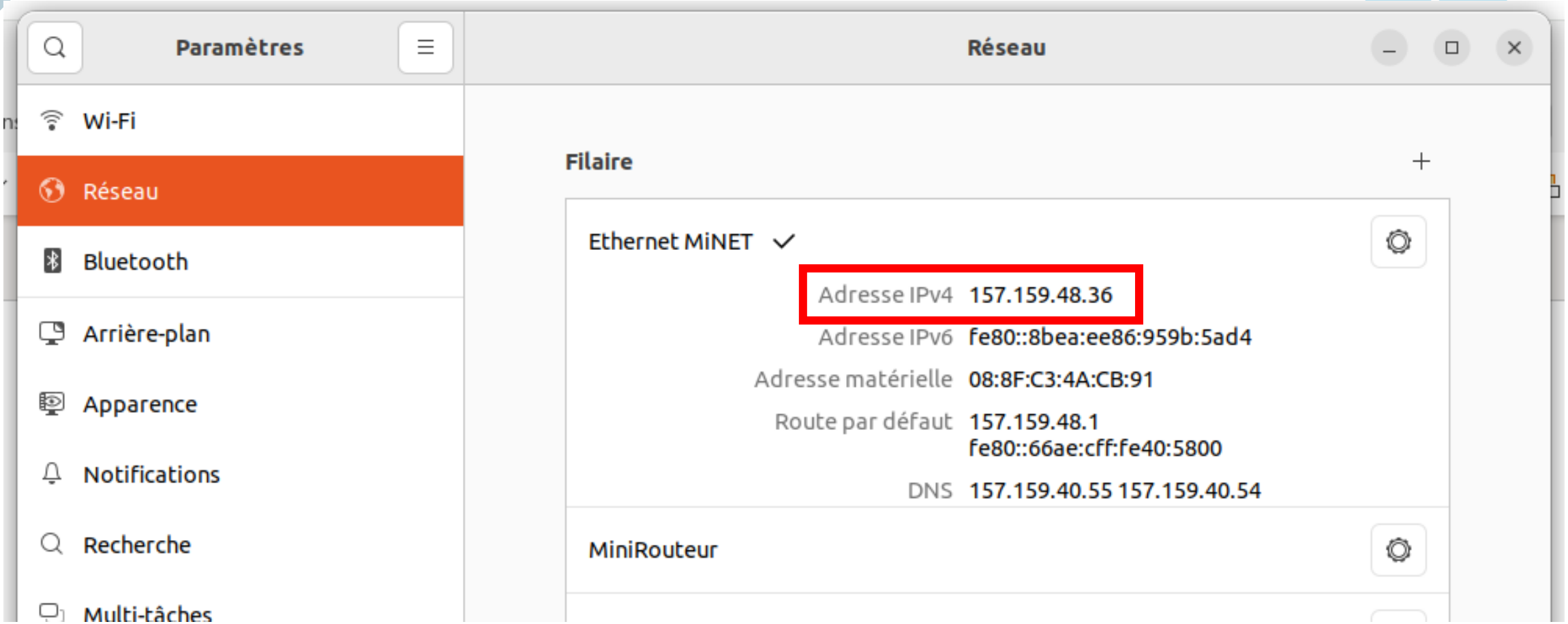
02

Et pour être rp, on va vraiment partir du principe que vous êtes pas sur votre machine de base.

03

Je ne rappelle pas comment on fait pour se ssh à un autre appareil :eyes::eyes:, mais un indice, il faut trouver son adresse IP sur le réseau, donc ici VOTRE adresse IP. 2ème indice on peut trouver ça en faisant "ip address", ou plus court : "ip a" (sinon vous pouvez aller voir dans les réglages, mais c'est pas très MiNET)

Mon adresse IP actuel :

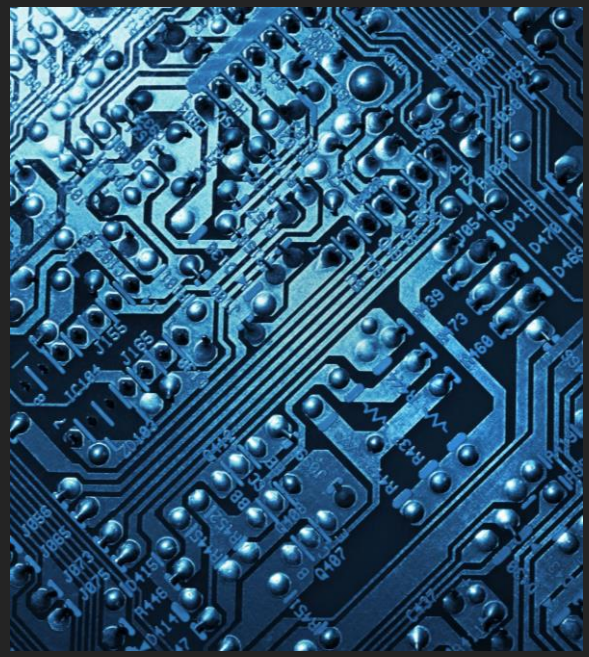


Ça marche pas ? C'est normal !
Pour se connecter en ssh, il faut verifier que :

1) Vous êtes apte à vous connecter en ssh

2) Que la machine en face accepte de vous laisser vous connecter en ssh

Et il faudra vérifier ces 2 conditions à chaque fois qu’on souhaite se ssh ou utiliser Ansible sur une machine



Pour se ssh sur quelqu'un d'autre, c'est simple :

Vérifier que ce package est bien installé (des fois on croit qu'il l'est, mais il l'est pas) : `$ sudo apt install ssh`

Ensuite il faut vous générer une clef ssh. Vous pouvez verifier si vous en avez déjà une en regardant dans votre dossier : ~/.ssh/

Sinon générez en une dans ce même dossier :

```
$ ssh-keygen -t rsa -b 4096 -C "votre_email@example.com"
```

Pour autoriser votre machine à ce qu'on se ssh dessus :

Pour ça il faut installer un autre package :

$ sudo apt install sshpass

Si vous voulez autoriser les connexions en utilisant le mot de passe de la machine, éditez le fichier de conf **/etc/ssh/sshd_config**, et décommentez l'instruction :

 # PasswordAuthentication yes

Sinon vous pouvez autoriser quelqu'un (par exemple vous même) à se connecter avec sa clef ssh. Pour ça il faut créer le fichier **~/.ssh/authaurized_keys**, puis copiez-collez votre clef publique (et toutes celles que vous voulez autoriser) dedans : **~/.ssh/id_rsa.pub**

Votre machine peut désormais se ssh, et elle peut accepter les demandes de ssh

Vous pouvez donc retenter de vous ssh sur votre adresse IP. Si ça fonctionne, votre terminal revient sur votre dossier perso ~, et si vous tapez la commande **$exit** ça ne quitte pas votre terminal, mais juste votre session

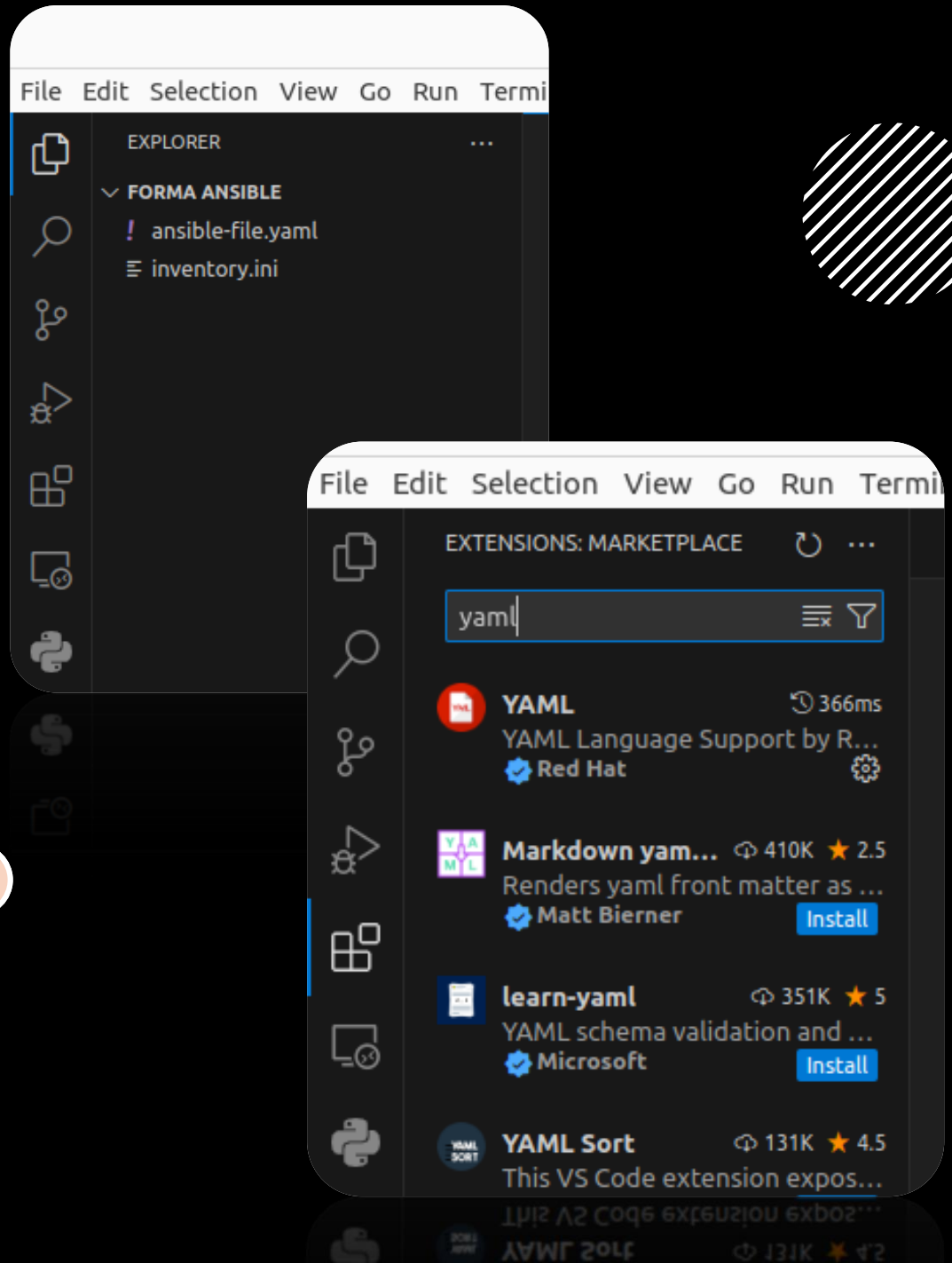
Maintenant on peut vraiment commencer Ansible. Et la meilleure manière de ansibler, c'est d'utiliser notre ami.e VSCod.ium.e

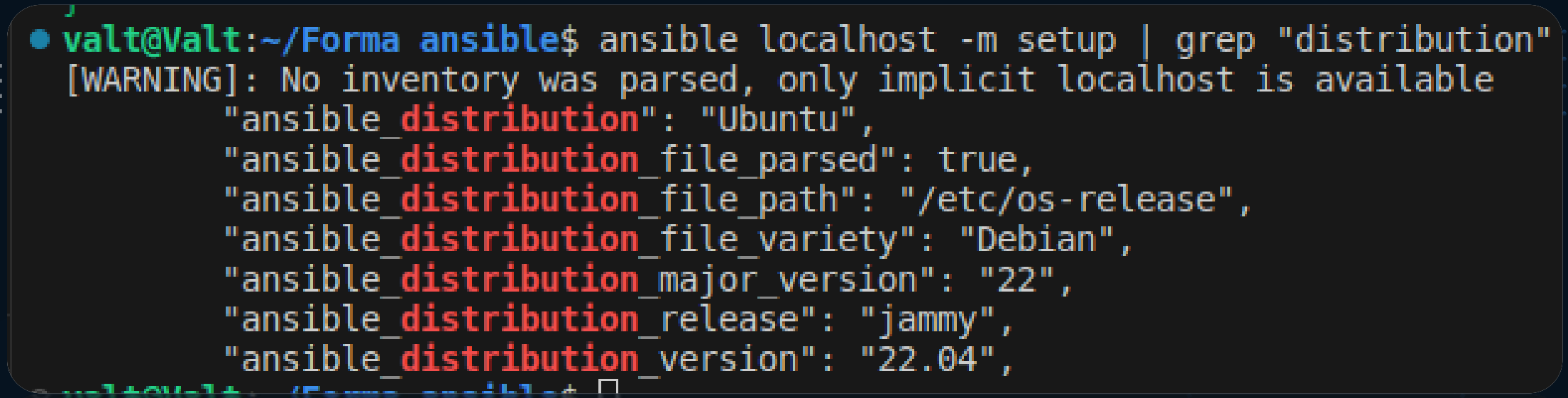
Comme d'hab vérifiez bien que vous avez l'extension correspondante d'installée (YAML)

Ensuite c'est bien de créer un dossier, puis dans VSCode : File > Open Folder

En enfin, on créer 2 fichiers : ansible-file.yml et inventaire.ini (IMPORTANT : les fichiers yaml doivent forcément contenir un tiret "–" dans leur nom

Le 1er c'est pour écrire nos instructions, et le 2ème pour lister les appareils sur lesquels on va appliquer nos instructions





Ok, notre premier objectif va être très simple, mais si on arrive à le faire fonctionner ça sera facile de faire plus complexe : On va lister 2 variables d'environnement de notre pc (WoW)

Enfait y'a pleins de variables d'environnement auquel ansible à accès par défaut s'il se connecte sur une machine

$ ansible localhost –m setup

Ansible se connecte sur "localhost", donc notre machine, et récupère des informations sur elle

$ ansible localhost -m setup | grep "distribution"

On précise ce qu'on recherche avec grep. Par exemple moi je voudrais afficher avec ansible la distribution de mon pc et sa version

Terminal Help

ansible-file.yaml | schema.json | inventory.ini

ansible-file.yaml > {} 0 > [ ] tasks > {} 0 > {} debug > msg

```yaml
---
- hosts: localhost
  remote_user: valt
  tasks:
  - name: afficher les variables globales
    debug:
      msg: "La distribution est une {{ ansible_distribution }} version {{ ansible_distribution_
```

Alors il se trouve que le yaml, qui est aussi utilisé pour du cicd sur gitlab, ou pour faire des docker-compose (déployer automatiquement pleins de container docker), est un langage de sorcier.

En gros, pour chaque instruction différente que tu veux faire, il y a une syntaxe précise à respecter. Et y'a pas moyen de la deviner sans regarder une config qui fait déjà à peu près ce que tu veux, de regarder la doc, ou demander à notre ami ChatGPT.

Mais c'est pas non plus très compliqué à modifier une fois qu'on connait la langue, du coup je vais essentiellement vous balancer du code, et vous pourrez l'adapter ensuite

```
---
- hosts: localhost
remote_user: valt
tasks:
- name: afficher les variables globales
debug:
msg: "La distribution est une {{ ansible_distribution }}
version {{ ansible_distribution_major_version }}"
```

Hosts : la machine sur laquelle on va exécuter nos instructions

Remote_user : la session sur laquelle on se connecte

Tasks: Alors il s'agit d'une liste. C'est pour ça qu'on va mettre des tirets. Même si en l'occurrence on aura qu'une seule tâche à accomplir

Name : le nom de la tâche (on peut mettre ce qu'on veut, c'est juste descriptif)

Debug : assez explicite, renvoie un message de debug

Notez les doubles accolades avec le nom des variables qu'on a vu juste avant

Dans Ansible on peut définir nos propres variables qui peuvent être utile pour automatiser (genre des adresses IP, des ports, des nom de service). Et on les appelle de la même manière entre double accolade {{}}

Sauf qu'ici, on vu tout à l'heure qu'ansible avait accès à ces variables par défaut. Du coup on peut les appeler où on veut.

Bravo ! Vous venez de créer votre premier "playbook"

On peut l'exécuter avec : **$ ansible-playbook ansible-file.yml**

Pour l'instant on a marqué "localhost" donc ça fonctionne forcément. Mais maintenant on va simuler une connexion à distance

Terminal Help

ansible-file.yaml | inventory.ini

inventory.ini

```
1  [mygroup]
2  Valt ansible_host=157.159.48.36 ansible_user=valt
3
```

Pour cela, il faut créer une liste d'appareil dans inventory.ini (une liste de UN appareil)

Entre crochet [], on peut préciser un nom de groupe pour mettre différents appareils. Bon encore une fois c'est un peu overkill pour un appareil, mais très pratique quand il faut manager 200 appareils aux caractéristiques très différentes.

On donne un nom à l'appareil (moi je vais l'appeler "Valt")

Et ensuite on peut donner toutes les infos utiles qu'on veut sur l'appareil. Les plus importantes étant les infos pour se connecter dessus

Du coup on précise le hostname, donc l'adresse IP qu'on a trouvé tout à l'heure, et le nom de la session sur laquelle on veut se connecter (là j'ai mis ma session perso, pour pas en créer une autre)

Faut noter que "ansible_host" et "ansible_user", c'est des variables propres à ansible, on peut pas les nommer autrement. Y'en a pleins d'autres comme ça.

Important

Pour autoriser la connexion ssh sur votre appareil, il faut également ajouter dans l'inventaire l'argument :

ansible_ssh_private_key_file=~/.ssh/id_rsa

En gros, vous précisez à votre fichier ansible où trouver votre clef privée, pour qu'il puisse l'utiliser pour se connecter à la machine. Je rappelle qu'on a déjà donné notre clef PUBLIQUE dans les authorized_keys de notre machine

Go Run Terminal Help

ansible-file.yaml ● inventory.ini

ansible-file.yaml > {} 0 > [ ] tasks > {} 0 > abc name

```
1  ---
2  - hosts: Valt
3    remote_user: ansible_user
4    tasks:
5    - name: afficher les variables globales
6      debug:
7        msg: "La distribution est une {{ ansible_distribution }} version {{ ansible_distribu
```

Ensuite, dans le fichier yaml, on change "localhost" en "Valt" (Enfin mettez le nom que vous avez mis vous), et pour remote_user vous mettez la variable "ansible_user“ (définie dans l’inventaire)

$ ansible-playbook -i inventory.ini ansible-file.yaml

L'option –i permet de préciser un fichier d'inventaire à utiliser pour le playbook

NoRmAlEmEnT ça fonctionne