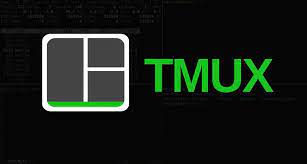
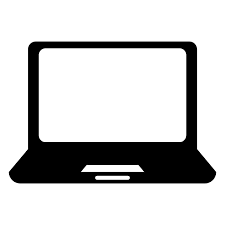
**Introduction**



**Tmux** est un gestionnaire de terminal en mode texte qui permet de créer, gérer et contrôler plusieurs fenêtres et panneaux à l'intérieur d'une seule session de terminal. C'est un outil puissant et largement utilisé par les développeurs, les administrateurs système et les utilisateurs avancés de la ligne de commande.

**NB :** la version utilisée dans cette formation et soit ***debian***, soit ***ubuntu***, pour d’autres distributions telles que ***Fodora***, ***Centos***,or ***Red*** ***Hat*** remplacez les commandes par celles qui leurs sont équivalentes

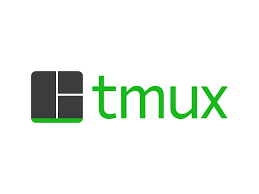
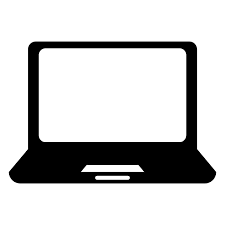
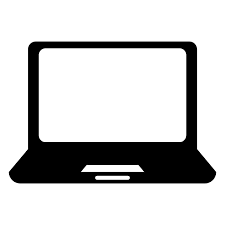
Dans cet atelier vous avez besoin de 2 machines virtuelles (3 si vous avez suffisamment de ressource). La machine v1 sera utilisée pour assurer un accès distant aux autres machines v2 (v3). Tmux sera installé sur la machine v1, qui sera la machine de contrôle (host manager) des autres vms.



Host manager

V1

V3



V2

Figure 1

* **Récupérer les IPs des machines v2 et v3**

~$ ip a

1: lo: <LOOPBACK,UP,LOWER\_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

inet 127.0.0.1/8 scope host lo

valid\_lft forever preferred\_lft forever

inet6 ::1/128 scope host

valid\_lft forever preferred\_lft forever

2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc fq\_codel state UP group default qlen 1000

link/ether 00:0c:29:0c:97:b6 brd ff:ff:ff:ff:ff:ff

altname enp2s1

inet 192.168.233.\*\*\*/24 brd 192.168.233.255 scope global dynamic ens33

valid\_lft 1472sec preferred\_lft 1472sec

inet6 fe80::20c:29ff:fe0c:97b6/64 scope link

valid\_lft forever preferred\_lft forever

sudo ip link set ens33 up

Si vous avez cloner une vm et que vous rencontrez un problème dans la détection de son @ip en tapant la commande précédente. Il semble que vous rencontriez un problème de configuration réseau persistant sur les machines virtuelles que vous avez clonées avec VMware. Ceci peut être dû à l’**UUID de l'interface réseau** : Lorsque vous clonez une machine virtuelle, l'UUID de l'interface réseau peut rester inchangé, ce qui peut entraîner des conflits. Pour résoudre ce problème vous pouvez taper les commandes suivantes sur chaque vm (v2 et v3)

~$ sudo ip link set [nom\_interface] up

~$ sudo dhclient -v [nom\_interface]

Vous pouvez aussi personnaliser le nom de vos machines en utilisant la commande suivante :

~$ hostnamectl set-hostname [nom\_machine]

* **Installation de tmux**

~$ sudo apt update

~$ sudo apt install tmux

* **Confgurer tmux**

Si vous avez installé tmux via *sudo apt install tmux*, il est possible que le fichier */.tmux.conf* n'existe pas par défaut. Vous pouvez créer ce fichier vous-même dans votre répertoire personnel (~/) et y ajouter vos configurations souhaitées.

~$ nano ~/.tmux.conf

# Set the status line style

**set -g status-style bg=default,fg=white**

# Highlight the active window in the status line

**set-window-option -g window-status-current-style bg=yellow,fg=white**

# Select the active window name with a solid background

**set-window-option -g window-status-current-format "#[fg=black,bg=white]#W"**

#vous pouvez utiliser **Ctrl + b** suivi de **r** pour recharger la configuration. Voici comment #vous pouvez le configurer :

**bind-key r source-file ~/.tmux.conf \; display-message "Config reloaded..."systemctl restart tmux**

Si pour une raison ou pour une autre vous ne pouvez pas créer votre fichier config dans votre home directory, ceci est dû à un problème de *permission*. Pour résoudre ce problème, vous devrez modifier le propriétaire (owner) de votre répertoire personnel pour la réattribuer à votre compte utilisateur. Vous pouvez le faire en suivant ces étapes :

* Ouvrez un terminal sur votre machine virtuelle Ubuntu VMware.
* Vérifiez le propriétaire actuel de votre répertoire personnel en exécutant la commande suivante :

~$ ls -ld /home/username

Remplacez "*username*" par votre nom d'utilisateur réel.

* Si le propriétaire est définie sur www-data ou autre, vous pouvez le réattribuer à votre compte utilisateur à l'aide de la commande ***chown*** :

~$ sudo chown -R username:username /home**/[username]**

* Après avoir modifié la propriété, vérifiez qu'elle a été mise à jour correctement en exécutant à nouveau la commande ls -ld.
* Vous devrez peut-être également ajuster les permissions sur votre répertoire personnel pour vous assurer que votre compte utilisateur dispose des droits d'accès nécessaires. Vous pouvez le faire en utilisant la commande ***chmod*** :

~$ sudo chmod 700 /home/username

The numbers in octal notation represent the permissions as follows:

4: Read permission

2: Write permission

1: Execute permission

The numbers are added together to represent different combinations of permissions. For example:

**4** (read) + **2** (write) + **1** (execute) = 7: Full permissions (read, write, and execute)

4 (read) + 0 (no write) + 0 (no execute) = 4: Read-only permissions

0 (no read) + 2 (write) + 0 (no execute) = 2: Write-only permissions

0 (no read) + 0 (no write) + 1 (execute) = 1: Execute-only permissions

So, when you see "700" in the context of file permissions, it means:

The ***owner*** (user) has read (**4**), write (**2**), and execute (**1**) permissions, giving a total of **7**.

The ***group*** and ***others*** have no permissions (**0**).

* **Lancer une première session tmux**

~$ tmux

* **Fermer toutes les session tmux**

~$ tmux kill-server

* **Vérifier si tmux est installé**

~$ command -v tmux

* **Récupérer la version**

~$ tmux -V

* **Détacher une session tmux**

Le "prefix key" +d

Le "***prefix key***" dans Tmux fait référence à la touche initiale que vous appuyez avant de taper une commande Tmux. Par défaut, la touche de préfixe est **Ctrl + b**, ce qui signifie que pour exécuter une commande Tmux, vous devez d'abord appuyer sur **Ctrl + b**, puis sur la touche correspondant à la commande spécifique que vous souhaitez exécuter.

* **Lister les sessions tmux détachées ou pas**

~$ tmux list-sessions

* **Rattacher une session tmux**

~$ tmux attach -t nom\_de\_la\_session

Si vous n'utilisez pas l'option **"-t",** "tmux attach" attachera automatiquement à la dernière session utilisée. Cependant, il est toujours recommandé de spécifier la session que vous souhaitez attacher pour éviter toute confusion, surtout si plusieurs sessions sont ouvertes.

* **Sectionner une fenêtre tmux**

|  |  |  |  |
| --- | --- | --- | --- |
| Section verticale | Section horizontale | Déplacement horizontale | Déplacement verticale |
| "prefix key" + % | "prefix key" + ‘’ | "prefix key" +-> ou <- | "prefix key" + >haut ou > bas |

Dans la machine host, créez une session tmux, créer une section verticale, puis créez dans la section verticale de gauche une section horizontale.

1

2

3

Taper la commande **htop** dans la section 1, ping google.fr dans la section 2 et ls dans la section3

La commande htop est un outil interactif de gestion des processus sur les systèmes Linux. Pour l’installer tapez la commande sudo apt install htop

Une image contenant texte, logiciel, Logiciel multimédia, Icône d’ordinateur

Description générée automatiquement

* **Quitter un panel**

~$ exit

Ou bien

"prefix key" + x

**exit** tapé dans la dernière fenêtre ou panel, ferme la session complètement

* **Afficher les numéros des panels**

"prefix key" + q

* **Créer une nouvelle fenêtre**

~$ tmux new-window -t nom\_de\_la\_session -n "Nom\_de\_la\_fenêtre"

~$ Tmux new-window

Si vous exécutez tmux new-window sans spécifier de session, tmux crée en fait une nouvelle fenêtre dans la dernière session utilisée, sinon il en crée une nouvelle session.

* **Renomer une fenetre**

"prefix key" + ,

~$ tmux rename-window -t nom\_de\_la\_session:nom\_actuel nouveau\_nom

~$ tmux rename-window nouveau\_nom (si je suis dans la fenetre)

* **Switching entre les fenêtres**

"prefix key" + p (previous) or "prefix key" + n (next)

* **Fermer une fenêtre (tue les process exécutés à l’intérieur de cette fenêtre)**

"prefix key" + &

* **Renommer une session**

~$ tmux rename-session -t ancien\_nom nouveau\_nom

~$ tmux rename-session nouveau\_nom (si je suis dans la session)

"prefix key" + $

* **Switching entre les sessions**

"prefix key" + $ 🡪 les flèches pour sélectionner une session particulière 🡪 enter

* **Configiration**

Le fichier de configuration de tmux, appelé **.tmux.conf**, est utilisé pour personnaliser et configurer les paramètres de tmux selon les préférences de l'utilisateur. Il peut contenir des commandes pour définir des raccourcis clavier, modifier l'apparence de l'interface utilisateur, activer des plugins et bien plus encore. En résumé, ce fichier permet de personnaliser et d'optimiser l'expérience d'utilisation de tmux en fonction des besoins spécifiques de chaque utilisateur.

**# Initial setup**

set -g default-terminal xterm-256color

set -g status-keys vi

**# Use Alt-arrow keys without prefix key to switch panes**

bind -n M-Left select-pane -L

bind -n M-Right select-pane -R

bind -n M-Up select-pane -U

bind -n M-Down select-pane -D

**# Set easier window split keys**

bind-key v split-window -h

bind-key h split-window -v

**# Shift arrow to switch windows**

bind -n S-Left previous-window

bind -n S-Right next-window

**# Easy config reload**

bind-key r source-file ~/.tmux.conf \; display-message "tmux.conf reloaded."

**# Mouse Mode**

set -g mouse on

**# Theme**

set-window-option -g window-status-current-style bold,bg=blue,fg=colour234

set-window-option -g window-status-style fg=colour35

set -g window-status-activity-style bold,bg=colour234,fg=white

set-option -g message-style bg=colour237,fg=colour231

set-option -g pane-border-style fg=colour36

set-option -g pane-active-border-style fg=colour35

**# Status Bar**

set -g status-justify centre

set -g status-bg black

set -g status-fg colour35

set -g status-interval 60

set -g status-left-length 50

set -g status-left "#[bg=colour35]💻#[fg=colour234,bold] #H#[bg=colour34]#[bg=colour35,nobold]#[fg=colour234] [#S] $tmux\_target\_lower"

set -g status-right '#[bg=colour35] 🕔 #[fg=colour234,bold]%H:%M '

|  |  |
| --- | --- |
| Une image contenant capture d’écran, logiciel, Logiciel multimédia, texte  Description générée automatiquement | Une image contenant capture d’écran, logiciel, Logiciel multimédia, texte  Description générée automatiquement |

# Synchronize panes

**bind-key y set-window-option synchronize-panes\; display-message "synchronize mode toggled."** : Elle active l'option **synchronize-panes** pour la fenêtre actuelle. Lorsque cette option est activée, toutes les autres panes dans la fenêtre actuelle seront synchronisées, ce qui signifie que tout ce qui est tapé dans une pane sera répliqué dans toutes les autres panes synchronisées

# Status Bar

**set -g status-justify centre** : Cette commande aligne le contenu de la barre d'état au centre de la fenêtre de tmux.

**set -g status-bg black** : Cette commande définit la couleur de fond de la barre d'état sur noir.

**set -g status-fg colour35** : Cette commande définit la couleur du texte de la barre d'état sur la couleur 35 (une nuance de gris).

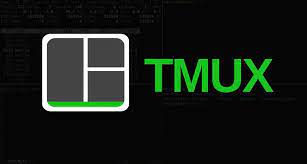
**set -g status-interval 60** : Cette commande spécifie l'intervalle de mise à jour de la barre d'état en secondes. Dans ce cas, la barre d'état est mise à jour toutes les 60 secondes.

**set -g status-left-length 50** : Cette commande spécifie la longueur de la partie gauche de la barre d'état. Dans ce cas, la partie gauche de la barre d'état a une longueur de 50 caractères.

**set -g status-left "#[bg=colour35]💻#[fg=colour234,bold] #H#[bg=colour34] #[bg=colour35,nobold]#[fg=colour234] [#S] $tmux\_target\_lower"** : Cette commande définit le contenu de la partie gauche de la barre d'état. Il affiche l'icône d'ordinateur "💻", le nom de l'hôte (**#H**), le nom de la session (**#S**), et la variable **$tmux\_target\_lower** qui représente la cible actuelle de tmux (comme la fenêtre ou le panneau actuellement sélectionné).

**set -g status-right '#[bg=colour35] 🕔 #[fg=colour234,bold]%H:%M'** : Cette commande définit le contenu de la partie droite de la barre d'état. Il affiche une icône d'horloge "🕔" et l'heure actuelle en format HH:MM.

**LAB**



Dans cet atelier vous allez préparer l’infrastructure nécessaire pour tester les commandes ansible. Pour cela :

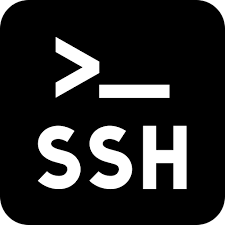
* Vous allez créer trois machines virtuelles VMware Pro 17.0 : v1 (hostManager), v2 (server1), et v3 facultative (server2 optionnelle si vous n’avez pas des ressources).
* Vérifiez les adresses IP de chacune de vos machines.
* Installez tmux sur la machine hostManager.
* Lancez une session tmux appelée "Ansible".
* Renommez la fenêtre par défaut qui a été créée en "HostManager".
* Créez une nouvelle fenêtre dans cette session appelée "Server1".
* Créez une nouvelle fenêtre appelée "Server2".
* Divisez la fenêtre de Server2 en deux panneaux verticaux.
* Dans le premier panneau, lancez la commande "htop", et dans le deuxième panneau, lancez la commande "ping google.com".
* Détachez-vous de la session.
* Rattachez-vous à la session.
* Quittez la fenêtre "Server3".
* Ajoutez un fichier de configuration. tmux.conf
* Copiez le contenu présenté précédemment.
* Rechargez la configuration.

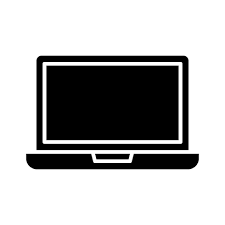
**Introduction**



**SSH**, acronyme de Secure Shell, est un protocole réseau permettant une communication sécurisée entre deux ordinateurs. Il s'agit d'un prérequis pour ***Ansible***, utilisé par ce dernier pour provisionner des serveurs distants de manière sécurisée.

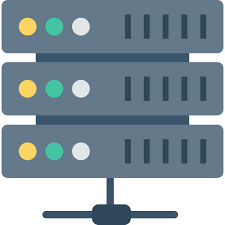
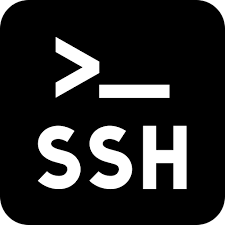
Dans cet atelier, vous allez configurer vos machines pour qu'elles puissent communiquer via le protocole SSH. Vous allez conserver la même configuration matérielle que lors de l'atelier précédent. Dans mon cas, j'ai trois machines virtuelles, dont une qui fait office de **host manager**, comme illustré dans la figure ci-dessous.

 Vous êtes libre de configurer les noms d'utilisateurs systèmes selon votre choix. Quant aux adresses IP, elles sont fournies à titre indicatif et seront différentes d'une machine à l'autre, dépendant du serveur DHCP.



Username : xirtam

Hostname : ubuntu

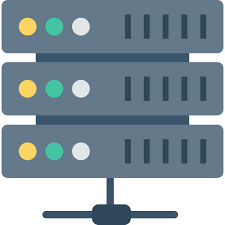


Serveur 1

Username : xyz

Hostname : Server1

IP : 192.168.233.133



Serveur 2

Username : xyz2

Hostname : Server2

IP : 192.168.233.133

* Sur le host manager sup primer la session tmux, et en créer une nouvelle dont le même nom ***Ansible***
* Créer deux fenêtres, la première est appelés **hostmanager**, la deuxième **server1**, et la troisième **server2**.
* Récupérer les IPs de vos serveurs en accédant une à une à vos machines virtuelles ( dans mon cas les adresse sont 192.168.233.133 et192.168.233.134
* Dans la fenêtre hostmanager assurez-vous que vous êtes capables de vous connecter aux deux serveurs.

~$ ping 192.168.233.133

~$ ping 192.168.233.133

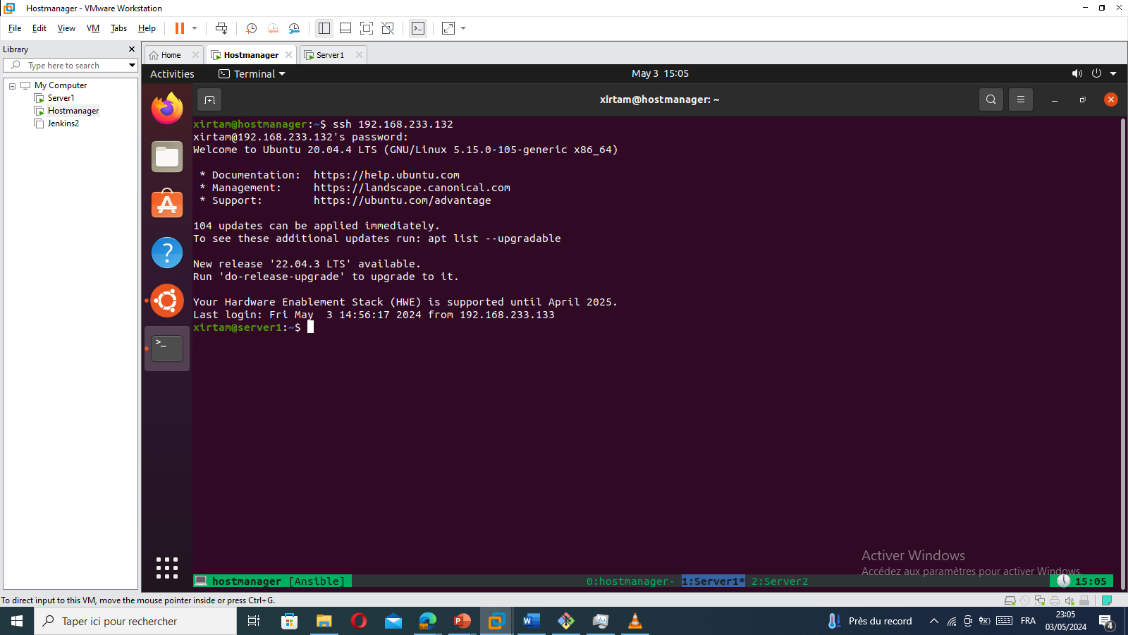
* Installer sur chaque serveur, **openssh server**

~$ sudo apt install openssh-server

* A partir de la session Asure de tmux connecter votre manager aux différents serveurs.
* Fenêtre server1 : **~$**  ssh username@ip [ssh [xyz@192.168.233.133](mailto:xyz@192.168.233.133)]
* Fenêtre server2 : **~$** ssh username@ip [ssh [xyz2@192.168.233.133](mailto:xyz2@192.168.233.133)]
* Si les trois machines virtuelles ont le même username, vous pouvez simplifier la commande en tapant : ssh ip
* Un message affiché vous demandera à chaque fois le mot de passe de l’utilisateur système du serveur. Si le mot de passe est correct, vous serez connectés à partir de l’host à votre serveur et vous voyez le prompt changé.

Dans ce qui suit vous allez modifier le modèle d’authentification de ssh, d’un modèle basé sur le mot de passe à un modèle basé sur les paires de clés (pub,privé). En fait vous allez créer deux paires de clés, une pour le *usersystem* et une deuxième pour *Ansible*.

Hostmanager VM : tmux : Ansible session



fenêtre 0 : hostmanager : relatif à la vm hostmanager

fenêtre 1 : server1 : connecté via ssh à la vm server1

fenêtre 2 : server1 : connecté via ssh à la vm server1

Trois fenêtres

* **Création d’une clé pour l’utilisateur système**

1. Placé vous dans la fenêtre hostmanager est créer une paire de clé pour le usersystème

**~$ ssh-keygen -t ed25519 -C "Default KEY"**

Generating public/private ed25519 key pair.

Enter file in which to save the key (/home/xirtam/.ssh/id\_ed25519): **1**

Enter PassWord : **2**

Enter passphrase (empty for no passphrase): **3**

Enter same passphrase again:

Your identification has been saved in /home/xirtam/.ssh/id\_ed25519

Your public key has been saved in /home/xirtam/.ssh/id\_ed25519.pub

The key fingerprint is:

SHA256:Xv7DE+mTUng9VAI8q2Foxz+0elWtF6C+tneGLTnCe+E Default KEY

The key's randomart image is:

+--[ED25519 256]--+

| ... |

| o.. .|

| o .o.o.|

| o =.o ..o|

| .Soo\* = o.|

| . oo.O.= .|

| . o\*o==o |

| o\*OE + |

| .=B+= |

+----[SHA256]-----+

1. Si vous avez exécuté la commande sans sudo, vos clés seront sauvegarder dans le répertoire ***.ssh de votre home repository.*** Vous pouvez appuiez sur entrer pour accpter la proposition du système ou bien, vous pouvez spécifier un autre nom et un autre emplacement pour vos clés. vous devrez utiliser l'option **-i** chaque fois que vous souhaitez vous connecter à un serveur distant en utilisant ces clés spécifiques. (-i) pour dire input file. (-t) pour type
2. Vous devez spécifier le mot de passe de l’utilisateur de la machine distante
3. Les phrases de passe sont utilisées pour renforcer la sécurité d'accès aux ressources distantes via SSH. Cela signifie que le serveur distant demande cette phrase pour autoriser l'accès. Cela peut sembler contradictoire avec l'avantage de l'utilisation des clés de sécurité, qui évitent de saisir les mots de passe à chaque tentative de connexion. Cependant, nous éviterons de taper les phrases de passe à chaque tentative de connexion en utilisant ssh-agent, sujet de la prochaine section. De plus, dans des scénarios d'automatisation des processus d'accès qui éliminent l'intervention, les phrases de passe peuvent être omises, permettant ainsi une connexion aussi fluide que possible.

* **Assurez vous de l’existence de vos clés :**

**~$ ls -al .ssh**

total 20

drwx------ 2 xirtam xirtam 4096 May 3 15:42 .

drwxr-xr-x 20 xirtam xirtam 4096 May 3 05:51 ..

-rw------- 1 root root 444 May 3 15:42 id\_ed25519

-rw-r--r-- 1 root root 93 May 3 15:42 id\_ed25519.pub

-rw-r--r-- 1 xirtam xirtam 222 May 3 06:13 known\_hosts

* **afficher le contenue de vos clé publique et privé :**

**~$ sudo cat .ssh/id\_ed25519**

[sudo] password for xirtam:

-----BEGIN OPENSSH PRIVATE KEY-----

b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmyyyyyyyyyyyyAAAABDhvT9ciR

J4Pu8eboPt2ciiAAxxxxAAAAzAAAAC3NzaC1lZDI1NTE5AAAAIBmtZ8lbFob2ijXa

Fr%%Nu9SHwp4f3IbOtsgXq9GLX3eAAAAkJu81FMypdZWGPcVogm3VIq34bUv43Hp4cJWRE

kfD5PSXGgx4iGds1UERuecb3iDsUxxxxxxxxxxxxxxxxxybRBKijLoR1WBisVF72Mr7WG4A

IP%%%%%%kr8oMmLPY7KJcHQ43tgFlxVLFN68c4qkbClDt1VzvQ56vl0iO+Pk757+PqVtg8+3rs

cmPmkku4H7jBFDHA==

-----END OPENSSH PRIVATE KEY-----

* **Créez une clé pour Ansible**
* Placé vous dans la fenêtre hostmanager est créer une paire de clé pour le usersystème

**~$ ssh-keygen -t ed25519 -C "Ansible"**

Generating public/private ed25519 key pair.

Enter file in which save the key(/home/xirtam/.ssh/id\_ed25519): /home/xirtam/.ssh/ansible **1**

Enter PassWord : **2**

Enter passphrase (empty for no passphrase): **3**

Enter same passphrase again:

Your identification has been saved in /home/xirtam/.ssh/ansible

Your public key has been saved in /home/xirtam/.ssh/ansible.pub

The key fingerprint is:

SHA256:Xxxxxxxxxxxxxxxxxxq2Foxz+0elWtF6C+tneGLTnCe+E Default KEY

The key's randomart image is:

+--[Ansible 256]--+

| ... |

| o.. .|

| o .o.o.|

| o =.o ..o|

| .\*\*\*\* = o.|

| . o\*.O.= .|

| ……… o\*o==o |

| o\*OE + |

| .=B+= |

+----[SHA256]-----+

1. Pour ne pas écraser la clé crée précédemment (ed25519) vous devez spécifie un autre nom à votre nouvelle clé (ansible dans ce cas)
2. Vous devez spécifier le mot de passe de l’utilisateur de la machine distante
3. Vous ignorer la phrase de sécurité dans ce cas, car cette clé sera utilisé afin d’automatisé la connexion de Ansible au différente machines à provisionner.

* **Copiez la clé publique vers les deux serveurs** **à partir de la fenêtre *hostmanager***

~$ ssh-copy-id -i ~/.ssh/**id\_ed25519.pub** [xirtam@192.168.233.122](mailto:xirtam@192.168.233.122)

~$ ssh-copy-id -i ~/.ssh/**alias.pub** [xirtam@192.168.233.123](mailto:xirtam@192.168.233.123)

Vérifiez qu’un fichier nommé ***authorized-keys*** a été ajouté dans chaque serveur, c’est à l’intérieur de ce fichier que seront copier les clés publiques utilisé par le serveur afin d’authentifier les demandes de connexions distantes.

* **Visualiser le contenu de cette clé (au niveau de chaque serveur)**

~$ cat ~/.ssh/authorized-keys

SHA256:Xv7DE+mTUng9VAI8q2Foxz+0elWtF6C+tneGLTnCe+E **Default KEY**

**Astuce**

Afin d’éviter d’utiliser la phrase de sécurité à chaque connexion à un serveur ssh, vous pouvez utiliser **ssh-agent** : qui est un programme qui gère les clés privées utilisées pour l'authentification SSH. L'agent SSH stocke les clés privées en mémoire tampon et les fournit aux clients SSH lorsque cela est nécessaire, sans que l'utilisateur ait à saisir manuellement la phrase de passe associée à la clé privée. Pour cela vous pouvez procéder ainsi

* Editer le fichier .bashrc

~$ nano .bashrc

* Défiler le fichier jusuqu’à que vous arrivez à la partie consacrée à la définition des alias et ajouetr la ligne suivante :

#alis ssh agent

alias ssha='eval $(ssh-agent) && ssh-add'

* Sauvegarder le fichier, fermer le terminal et ouverez un nouveau terminal, puis saisir la commande suivante :

~$ ssha

Agent pid 3255

Enter passphrase for /home/xirtam/.ssh/id\_ed25519:

Identity added: /home/xirtam/.ssh/id\_ed25519 (Default Key)

//Cette commande vous demandra de saisir les phrases relatif à chaque clé que vous avez renforcé avec une phrase de sécurité.

Vérifier maintenant que vous pouvez vous connecter sans que la hrase de sécurité ne vous sera demandé à chaque connexion

~$ ssh 192.168.233.132

Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.15.0-105-generic x86\_64)

\* Documentation: https://help.ubuntu.com

\* Management: https://landscape.canonical.com

\* Support: https://ubuntu.com/advantage

103 updates can be applied immediately.

To see these additional updates run: apt list --upgradable

New release '22.04.3 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.

Last login: Sat May 4 16:10:01 2024 from 192.168.233.133

**xirtam@server1:~$ <--- serveur connecté**

**Introduction**



GitHub est une plateforme de développement de logiciels qui utilise Git pour le contrôle de version et la collaboration. L'intégration d'Ansible avec GitHub offre plusieurs avantages :

**Gestion centralisée du code** : GitHub permet de stocker le code source de manière centralisée, offrant ainsi un emplacement unique pour gérer tous les artefacts liés au développement de logiciels.

**Contrôle de version efficace** : Git, intégré à GitHub, offre un système de contrôle de version distribué puissant. Cela permet de suivre les modifications apportées au code au fil du temps, de revenir à des versions antérieures si nécessaire et de fusionner les modifications de plusieurs contributeurs.

**Intégration continue** : Ansible peut être intégré à des services d'intégration continue tels que Jenkins ou bien GitHub Actions. Cela permet d'automatiser le déploiement, les tests et la validation du code à chaque nouvelle modification du référentiel GitHub.

**Collaboration et gestion des workflows** : GitHub facilite la collaboration entre les membres de l'équipe en fournissant des fonctionnalités telles que les demandes d'extraction, les commentaires sur le code et la gestion des problèmes. Ansible peut être utilisé pour automatiser certains aspects de ces workflows, tels que le déploiement d'une nouvelle version de logiciel en réponse à une demande d'extraction approuvée.

**Gestion des secrets et des variables sensibles** : Ansible Vault peut être utilisé pour chiffrer et stocker en toute sécurité les variables sensibles, tels que les jetons d'API ou les informations d'identification, dans les référentiels GitHub. Cela permet de maintenir la sécurité des informations sensibles tout en les rendant accessibles aux tâches d'automatisation Ansible.

* **Créez un depôt github (remote repository)**

1. Si vous n’avez pas encore un compte Github, créez-en un
2. Créez un projet Github intitule : ansible

GitHub🡪**new**🡪nommez le projet ansible🡪 cochez la case **Add a README file**🡪**Create repository**

1. Si Git n'est pas installé sur votre machine, veuillez procéder à son installation

**~$** sudo apt upadate

**~$** sudo apt install git

1. Ajouter votre clé publique **id\_25519.pub** crée à votre repo github pour pouvoir réussir à pousser vos modifications

Sur la machine hostmanager copier le contenu de la clé id\_25519.pub

**~$** clip < ~/.ssh/id\_ed25519.pub

Dans github 🡪avatar/profil🡪settings🡪ssh and GPG keys🡪New ssh key🡪nommez puis coller la clé publique copiée🡪Add ssh key

**Introduction**



Ansible est une plateforme open-source d'automatisation informatique. Il permet de simplifier la gestion des configurations, le déploiement d'applications, l'orchestration des infrastructures et la gestion des mises à jour à grande échelle. Voici quelques points clés à connaître sur Ansible :

**Agentless** : Ansible est un outil agentless, ce qui signifie qu'il n'a pas besoin d'installer un agent sur les machines cibles pour les gérer. Il utilise SSH (Secure Shell) pour se connecter aux machines distantes et exécuter des tâches à travers des modules.

**Infrastructure as Code** : Ansible adopte le paradigme d'Infrastructure as Code (IaC), ce qui permet de décrire toute l'infrastructure d'un environnement informatique dans des fichiers texte, appelés playbooks, qui peuvent être versionnés et partagés avec l'équipe.

**Déclaratif** : Ansible utilise une approche déclarative pour décrire l'état souhaité du système plutôt que d'écrire des scripts impératifs. Les playbooks décrivent simplement les actions à effectuer, et Ansible se charge de les exécuter de manière séquentielle et cohérente.

**Extensibilité** : Ansible est hautement extensible et dispose d'une large gamme de modules intégrés pour gérer divers aspects des systèmes d'exploitation, des applications et des services. De plus, il est possible de créer des modules personnalisés pour répondre à des besoins spécifiques.

**Facilité d'utilisation** : Ansible est conçu pour être facile à apprendre et à utiliser. Son langage simple basé sur YAML (YAML Ain't Markup Language) le rend accessible aux administrateurs système, aux développeurs et aux ingénieurs DevOps.

**Large communauté et écosystème** : Ansible bénéficie d'une large communauté d'utilisateurs et de contributeurs, ce qui se traduit par un écosystème riche en ressources, en modules et en intégrations avec d'autres outils et plates-formes.

* **Installation de ansible**

~$ sudo apt-add-repository ppa:ansible/ansible

~$ sudo apt iupdate

~$ sudo apt install ansible

Bien qu'Ansible crée généralement un fichier d'inventaire par défaut à **etc/ansible/hosts**, vous êtes libre de créer des fichiers d'inventaire à n'importe quel emplacement qui correspond mieux à vos besoins. Dans ce cas, vous devrez fournir le chemin vers votre fichier d'inventaire personnalisé avec le paramètre -i lors de l'exécution des commandes et des playbooks Ansible. Utiliser des fichiers d'inventaire par projet est une bonne pratique pour minimiser le risque d'exécuter un playbook sur le mauvais groupe de serveurs.

* **Créez un fichier d'inventaire qui contiendra la liste des machines qui seront configurer par ansible**

Accédez au projet cloné

~$ cd your project

~/yourproject$ nano inventory

192.168.233.133

* **Poussez les modifications réalisées vers GitHub**

~/yourproject$ git add inventory

~/yourproject$ git commit -m « fichier inventaire crée »

~/yourproject$ git origin main

* **Saisissez votre première commande ansible**

~/yourproject$ ansible all –key-file ~/.ssh/ansible all --key-file ~/.ssh/ansible -I inventory -m ping

192.168.233.132 | **SUCCESS** => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"changed": false,

"ping": "pong"

}

**ansible** : C'est la commande principale d'Ansible, utilisée pour exécuter des tâches sur des hôtes distants.

**all** : C'est l'argument qui spécifie que la tâche doit être exécutée sur tous les hôtes répertoriés dans l'inventaire. "all" est un alias qui désigne tous les hôtes.

**--key-file** **~/.ssh**/**ansible** : C'est l'option qui spécifie le chemin vers la clé privée SSH à utiliser pour se connecter aux hôtes distants. Dans cet exemple, la clé privée est située dans ~/.ssh/ansible.

**-I** inventory : C'est l'option qui spécifie le chemin vers l'inventaire à utiliser. L'inventaire est un fichier ou un script qui répertorie les hôtes sur lesquels Ansible doit agir. Dans cet exemple, l'inventaire est situé dans le fichier inventory.

**-m** **ping** : C'est l'option qui spécifie le module à utiliser. Dans ce cas, le module utilisé est "ping". Le module "ping" est un module de test qui envoie une requête de ping à tous les hôtes pour vérifier s'ils sont accessibles et répondent.

* **Simplifiez la commande**

~/yourproject$ nano ansible.cfg

[defaults]

inventory = inventory

private\_key\_file = ~/.ssh/ansible

Sauvegarder le fichier puis retapez la commande simplifiée

~/yourproject$ ansible all -m ping

* **Listez les hosts**

~/yourproject$ ansible all --list-host

hosts (1):

192.168.233.132

* **Récupérez les informations des serveurs**

~/yourproject$ ansible all -m gather\_facts –limit 192.168.233.132

192.168.233.132 | SUCCESS => {

"ansible\_facts": {

"ansible\_all\_ipv4\_addresses": [

"172.18.0.1",

"192.168.233.132",

"172.17.0.1"

],

"ansible\_all\_ipv6\_addresses": [

"fe80::42:9dff:fe4a:b96b",

"fe80::a406:b970:ded2:54ec",

"fe80::e093:5bff:feab:7576"

],

Le **module gather\_facts** collecte des informations détaillées sur l'hôte, telles que l'adresse IP, la version du système d'exploitation, la version du noyau, la mémoire disponible et d'autres détails liés au système. Ces informations peuvent être utiles pour des tâches telles que la gestion de la configuration, la surveillance et le dépannage.

Cette commande n’effectue aucun changement auprès des serveurs

Le problème c’est que lorsque vous voulez exécuter des commandes demandant des privilèges plus élevés, c’est à dire des commande qui peuvent modifier l’état du système, ces commandes échouent à s’exécuter sur les serveurs.

~/yourproject$ ansible all -m apt -a update-cache=true

92.168.233.132 | **FAILED**! => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"changed": false,

"msg": "Failed to lock apt for exclusive operation: Failed to lock directory /var/lib/apt/lists/: E:Could not open lock file /var/lib/apt/lists/lock - open (13: Permission denied)"

}

**-a update-cache=true :** Cela spécifie les arguments à passer au module. Dans ce cas, il indique au module apt de mettre à jour le cache des paquets.

Ce message nous indique que nous devons avoir le privilège root ou équivalent pour pouvoir exécuter la commande. Ceci veut dire que vous devez utiliser **sudo** pour exécuter cette commande et saisir le mot de passe qui sera demandé par le système si c’est nécessaire, celui est en contradiction avec le principe d’automatisation de ansible

Cependant, vous avez ajouté l'option ***--ask-become-pass***, ce qui signifie qu'Ansible vous demandera le mot de passe de l'utilisateur root sur chaque hôte pour l'exécution de la commande. Ceci suppose bien sûr dans ce cas que tous vos serveurs ont le même mot de passe.

~/yourproject$ ansible all -m apt -a update-cache=true --become --ask-become-pass

BECOME password:

192.168.233.132 | CHANGED => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"cache\_update\_time": 1715119226,

"cache\_updated": true,

"changed": true

}

Si vous avez un problème timeout suite à l’exécution de vos commande ansible, vous pouvez résoudre ce problème en modifiant le fichier ansible.cfg en ajoutant la ligne suivante

timeout = 10 ( change to 60 )

* Installez de **vim-nox** (spécifique pour pyton)

~/yourproject$ ansible all -m apt -a name=vim-nox --become --ask-become-pass

BECOME password:

92.168.233.132 | CHANGED => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"cache\_update\_time": 1715119226,

"cache\_updated": false,

"changed": true,

"stderr": "",

"stderr\_lines": [],

"stdout": "Reading package lists...\nBuilding dependency tree...\nReading state information...\nThe following …

* **Vérifier l’existante de vim-nox au niveau du serveur 1**

xyz@server1:~$ Which vim

/usr/bin/vim

Apt search vim-nox

orting... Done

Full Text Search... Done

vim-nox/focal-updates,focal-security,now 2:8.1.2269-1ubuntu5.22 amd64 [**installed**]

Vi IMproved - enhanced vi editor - with scripting languages support

vim-tiny/focal-updates,focal-security,now 2:8.1.2269-1ubuntu5.22 amd64 [installed,automatic]

Vi IMproved - enhanced vi editor - compact version

Si vous exécutez la commande encore une fois vous aurez une indication que le programme est déjà installé sur chaque serveur, et qu’il n’y a pas eu de changements sur les serveurs.

~/yourproject$ ansible all -m apt -a name=vim-nox --become --ask-become-pass

BECOME password:

192.168.233.132 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"cache\_update\_time": 1715119226,

"cache\_updated": false,

**"changed": false**

}

* **Tester l’existence de l’application installée au niveau du serveur 1**

xyz@server1:~$ cd /var/log

xyz @server1:/var/log$ ls

alternatives.log dmesg gpu-manager.log syslog unattended-upgrades vmware-network.log

alternatives.log.1 dmesg.0 hp syslog.1 vmware vmware-vmsvc-root.1.log

**apt**  dmesg.1.gz installer syslog.2.gz vmware-network.1.log vmware-vmsvc-root.2.log

auth.log dmesg.2.gz journal syslog.3.gz vmware-

…….

xyz@server1:/var/log$ cd apt/

xyz @server1:/var/log/apt$ ls

eipp.log.xz **history.log** history.log.1.gz term.log term.log.1.gz

xyz@server1:/var/log/apt$ cat history.log

Commandline: apt install ansible

Requested-By: xirtam (1000)

**Start-Date: 2024-05-05 02:05:49**

Install: **python3-xmltodict**:amd64 (0.12.0-1, automatic), python3-requests-ntlm:amd64 (1.1.0-1, automatic), python3-requests-kerberos:amd64 (0.12.0-2, automatic), python3-winrm:amd64 (0.3.0-2, automatic), python3-jmespath:amd64 (0.9.4-2ubuntu1, automatic), python3-resolvelib:amd64 (0.5.4-2ppa~focal, automatic), python3-pyparsing:amd64 (2.4.6-1, automatic), python3-markupsafe:amd64 (1.1.0-1build2, automatic), ansible-core:amd64 (2.12.10-1ppa~focal, automatic), python3-kerberos:amd64 (1.1.14-3.1build1, automatic), python3-paramiko:amd64 (2.6.0-2ubuntu0.3, automatic), python3-bcrypt:amd64 (3.1.7-2ubuntu1, automatic), python3-jinja2:amd64 (2.10.1-2ubuntu0.2, automatic), ansible:amd64 (5.10.0-1ppa~focal), python3-ntlm-auth:amd64 (1.1.0-1, automatic), sshpass:amd64 (1.06-1, automatic), python3-packaging:amd64 (20.3-1, automatic)

End-Date: 2024-05-04 21:03:42

**Start-Date: 2024-05-07 15:15:49**

Commandline: /usr/bin/apt-get -y -o Dpkg::Options::=--force-confdef -o Dpkg::Options::=--force-confold install **vim-nox**

Requested-By: xirtam (1000)

Install: javascript-common:amd64 (11, automatic), ruby2.7:amd64 (2.7.0-5ubuntu1.12, automatic), rake:amd64 (13.0.1-4, automatic), ruby-net-telnet:amd64 (0.1.1-2, automatic), libtcl8.6:amd64 (8.6.10+dfsg-1, automatic), libjs-jquery:amd64 (3.3.1~dfsg-3, automatic), vim-nox:amd64 (2:8.1.2269-1ubuntu5.22), ruby-minitest:amd64 (5.13.0-1, automatic), ruby-xmlrpc:amd64 (0.3.0-2, automatic), libruby2.7:amd64 (2.7.0-5ubuntu1.12, automatic), ruby:amd64 (1:2.7+1, automatic), vim-runtime:amd64 (2:8.1.2269-1ubuntu5.22, automatic), liblua5.2-0:amd64 (5.2.4-1.1build3, automatic), ruby-power-assert:amd64 (1.1.7-1, automatic), rubygems-integration:amd64 (1.16, automatic), fonts-lato:amd64 (2.0-2, automatic), ruby-test-unit:amd64 (3.3.5-1, automatic)

End-Date: 2024-05-07 15:16:02

Le répertoire /var/log sous Ubuntu (et d'autres distributions Linux) contient des fichiers journaux qui enregistrent différentes activités système, telles que les événements système, les messages du noyau, les activités des utilisateurs, les informations de démarrage, les messages des services, les erreurs et bien plus encore.

Ansible ne fait pas de magie, ansible utilise les paquets intégrés avec le système

xyz@server1:~$ sudo apt update

Hit:1 http://security.ubuntu.com/ubuntu focal-security InRelease

Hit:2 http://ppa.launchpad.net/ansible/ansible/ubuntu focal InRelease

Hit:3 http://us.archive.ubuntu.com/ubuntu focal InRelease

Hit:4 https://download.docker.com/linux/ubuntu focal InRelease

Hit:5 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease

Get:6 https://packages.microsoft.com/repos/vscode stable InRelease [3,594 B]

Get:7 https://packages.microsoft.com/repos/code stable InRelease [3,590 B]

Hit:8 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease

Fetched 7,184 B in 2s (4,715 B/s)

Reading package lists... Done

Building dependency tree

Reading state information... Done

**106 packages can be upgraded. Run 'apt list --upgradable' to see them.**

La commande sudo apt dist-upgrade est plus agressive que apt upgrade dans la gestion des dépendances. Elle prend en compte les nouvelles dépendances introduites par les mises à jour et peut désinstaller ou installer de nouveaux paquets pour résoudre les conflits.

Elle met à jour tous les paquets installés sur le système vers les versions les plus récentes disponibles dans les dépôts de logiciels configurés.

xyz@server1:~$ sudo apt update

sudo apt dist-upgrade

Reading package lists... Done

Building dependency tree

Reading state information... Done

Calculating upgrade... Done

The following packages were automatically installed and are no longer required:

gir1.2-goa-1.0 libfwupdplugin1 libxmlb1

Use 'sudo apt autoremove' to remove them.

The following NEW packages will be installed:

libfwupdplugin5 libxmlb2

The following packages will be upgraded:

alsa-ucm-conf apparmor apport apport-gtk apt apt-utils base-files bolt code command-not-found distro-info distro-info-data dns-root-data

…

python3-problem-report python3-tz python3-update-manager rsync sbsigntool snapd systemd systemd-sysv systemd-timesyncd tcpdump

tracker-extract tracker-miner-fs ubuntu-advantage-tools ubuntu-drivers-common ubuntu-pro-client ubuntu-release-upgrader-core

ubuntu-release-upgrader-gtk ufw unattended-upgrades unzip update-manager update-manager-core update-notifier update-notifier-common

….

106 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.

1 standard LTS security update

Need to get 378 MB of archives.

After this operation, 36.2 MB of additional disk space will be used.

**Do you want to continue? [Y/n]**

Vous pouvez vérifier que les mises à jour ont été réalisé au niveau serveur 1 à partir de la machine hôte. Il suffit dans ce cas de relancer la dernière commande ansible. Vous remarquerez que l’état du programme reste inchangé. Vous pouvez vérifier que snapd n’est plus présent dans la liste des modules nécessitant des mises à jour.

* **Vérifier l’état de u programme à partir de la machine hôte**

~/yourproject$ ansible all -m apt -a “name=snapd state=latest” --become --ask-become-pass

BECOME password:

192.168.233.132 | SUCCESS => {

"ansible\_facts": {

"discovered\_interpreter\_python": "/usr/bin/python3"

},

"cache\_update\_time": 1715121945,

"cache\_updated": false,

**"changed": false**

}

**Snapd** est le démon (daemon) de gestion de paquets pour le système de gestion de paquets snap. Snap est un système de gestion de paquets et de distribution d'applications pour Linux, développé par Canonical, la société derrière Ubuntu.

**sudo apt update** : Actualise la liste des paquets disponibles dans les dépôts de logiciels.

**sudo apt upgrade** : Met à jour les paquets installés sur votre système vers les versions les plus récentes

Vous pouvez réaliser la mm chose pour tous les modules avec une seule commande, qui en fait réalisera toutes les mis à jour sur tous les serveurs

~/yourproject$ ansible all -m apt -a upgrade=dist --become --ask-become-pass

**-a "upgrade=dist"** : Cela spécifie les arguments à passer au module. Dans ce cas, il indique au module apt d'exécuter une mise à jour de distribution (dist-upgrade).

**Premier playbook ansible**

Un Playbook Ansible est un fichier YAML utilisé pour décrire une série d'actions à exécuter sur un ensemble d'hôtes cibles. Il est utilisé pour automatiser des tâches de gestion système, de déploiement d'applications et de configuration d'infrastructure. Playbook définit en fait l’état désirable de nos systèmes installés sur les différents serveurs, en précisant les différentes tâches et actions à réaliser afin de ramener nos systèmes à cet état.

~/yourproject$ nano install\_apache.yaml

#équivalent de la commande ansible all

- hosts: all

become: true

tasks:

# le message qui va être affiché lorsque la tâche sera exécutée

- name: install apache2 package

apt:

name: apache2

* **Exécuter votre premier playbook**

~/yourproject$ ansible-playbook --ask-become-pass install\_apache.yaml

BECOME password:

PLAY [all] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

ok: [192.168.233.139]

TASK [install apache2 package] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.132 : ok=2 **changed=1** unreachable=0 failed=0 skipped=0 rescued=0

ignored=0

**TASKS**: C'est la section qui résume le nombre total de tâches exécutées dans le playbook.

**ok=XX**: Indique le nombre de tâches qui se sont terminées avec succès. Le chiffre après le signe "ok=" représente le nombre de tâches réussies.

**changed=XX**: Indique le nombre de tâches qui ont modifié l'état du système. Par exemple, si une tâche a copié un fichier ou redémarré un service, elle sera considérée comme modifiée. Le chiffre après le signe "changed=" représente le nombre de tâches modifiées.

**unreachable=XX**: Indique le nombre d'hôtes cibles sur lesquels Ansible n'a pas pu se connecter. Cela peut être dû à un problème de connectivité réseau ou à des paramètres d'authentification incorrects. Le chiffre après le signe "unreachable=" représente le nombre d'hôtes inaccessibles.

**failed=XX**: Indique le nombre de tâches qui ont échoué. Le chiffre après le signe "failed=" représente le nombre de tâches qui ont rencontré des erreurs lors de leur exécution.

**skipped=XX**: Indique le nombre de tâches qui ont été ignorées. Cela peut se produire lorsque les conditions pour exécuter une tâche ne sont pas remplies ou lorsqu'une tâche est explicitement marquée comme ignorée.

Le fait que le récapitulatif de la tâche indique "ok=2" signifie que deux tâches ont été exécutées avec succès. Dans votre Playbook, vous avez une seule tâche définie pour installer le paquet Apache2 sur tous les hôtes (apt: name=apache2). Cependant, le nombre total de tâches réussies est 2, ce qui peut sembler étrange car vous n'avez qu'une seule tâche définie.

La raison pour laquelle vous obtenez "ok=2" est probablement due au fait qu'Ansible exécute des tâches supplémentaires en interne pour gérer les privilèges (become: true). Lorsque vous définissez become: true, Ansible utilise sudo ou su pour exécuter les tâches avec les privilèges de super utilisateur, si nécessaire. Donc, même si vous avez une seule tâche explicite dans votre playbook, Ansible exécute une tâche supplémentaire en arrière-plan pour gérer les privilèges, ce qui explique pourquoi vous obtenez "ok=2".

En résumé, le fait que vous obteniez "ok=2" indique que deux tâches ont été exécutées avec succès : une pour l'installation du paquet Apache2 et une autre pour la gestion des privilèges.

* **Une réexécution de cette commande donne le résultat suivant**

~/yourproject$ ansible-playbook --ask-become-pass install\_apache.yaml

BECOME password:

PLAY [all] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

ok: [192.168.233.139]

TASK [install apache2 package] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**ok: [192.168.233.132]**

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.132 : ok=2 **changed=0** unreachable=0 failed=0 skipped=0 rescued=0

ignored=0

Maintenant vous pouvez accéder a votre serveur apache en ouvrant un anglais sur la machine hôte et en tapant la l’adresse suivante : <http://192.168.233.132>

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Description générée automatiquement

* **Un autre scénario dans lequel vous allez installer un package non existant**

~/yourproject$ nano install\_apache.yaml

#équivalent de la commande ansible all

- hosts: all

become: true

tasks:

# le message qui va être affiché lorsque la tâche sera exécutée

- name: install apache2 package

apt:

name: **'cocomo'**

ansible-playbook --ask-become-pass install\_apache.yaml

BECOME password:

PLAY [all] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

TASK [install apache2 package] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**fatal: [192.168.233.132]: FAILED! => {"changed": false, "msg": "No package matching 'cocomo' is available"}**

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.132 : **ok=1** changed=0 unreachable=0 **failed=1** skipped=0 rescued=0 ignored=0

* **Ajouter une nouvelle tâche à votre playbook, qui consiste à mettre à jour l’index des repo avant d’exécuter le module apache**

~/yourproject$ nano install\_apache.yaml

---

#équivalent de la commande ansible all

- hosts: all

become: true

tasks:

# le message qui va être affiché lorsque la tâche sera exécutée

- name: update repository index

apt :

update\_cache: yes

- name: install apache2 package

apt:

name: apache2

~/yourproject$ ansible-playbook --ask-become-pass install\_apache.yaml

BECOME password:

PLAY [all] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

TASK [update repository index] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

TASK [install apache2 package] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.132 **: ok=3** changed=1 unreachable=0 failed=0 skipped=0 rescued=0

* **Ajoutez un troisième module et observez le comportement du module, cette fois ci vous allez installer la dernière version de chaque module en ajoutant l’option state : latest dans le fichier yml.**

**~/yourproject$ nano install\_apache.yaml**

---

---

#équivalent de la commande ansible all

- hosts: all

become: true

tasks:

# le message qui va être affiché lorsque la tâche sera exécutée

- name: update repository index

apt :

update\_cache: yes

- name: install apache2 package

apt:

name: apache2

state: latest

- name: add php support for apache

apt:

name: libapache2-mod-php

state: latest

**~/yourproject$ ansible-playbook --ask-become-pass install\_apache.yaml**

BECOME password:

PLAY [all] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

TASK [update repository index] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

TASK [install apache2 package] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

TASK [add php support for apache] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.132 : ok=4 changed=2 unreachable=0 failed=0 skipped=0 rescued=0

Il est possible de créer un deuxième fichier dont l’objectif est de purger les systèmes des différents serveurs. Ça reviendra en fait à supprimer tout ce qui a été installé.

* **Purger les systèmes des serveurs**

#créerz un nouveau fichier remove\_ansible.yaml est effectuez les modifications nécessaires pour nettoyer les système.

~/yourproject$ cp install-apache.yaml remove\_apache.yaml

~/yourproject$ nano install\_apache.yaml

---

#équivalent de la commande ansible all

- hosts: all

become: true

tasks:

# le message qui va être affiché lorsque la tâche sera exécutée

- name: update repository index

apt :

update\_cache: yes

- name: install apache2 package

apt:

name: apache2

state: **absent**

- name: add php support for apache

apt:

name: libapache2-mod-php

state: **absent**

~/Formation-Ansible$ ansible-playbook --ask-become-pass remove\_apache.yaml

BECOME password:

PLAY [all] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.132]

TASK [update repository index] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

TASK [install apache2 package] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

TASK [add php support for apache] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.132]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.132 : ok=4 changed=3 unreachable=0 failed=0 skipped=0 rescued=0

En accedant à l ‘adresse <http://192.168.233.132> vous remarquerez que le serveur apache n’est plus opérationnel

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

while

xirtam@hostmanager:~/Formation-Ansible$ nano whilePlaybook.yml

--

- name: while

hosts: all

become: yes

tasks:

- name: creation de rep

file:

state: directory

path: /home/xirtam/ansibleLab/{{ item }}

recurse: yes

with\_items:

- rep1

- rep2

xirtam@hostmanager:~/Formation-Ansible$ ansible-playbook --ask-become-pass whilePlaybook.yml

BECOME password:

PLAY [while] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Gathering Facts] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.233.135]

TASK [creation de rep] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [192.168.233.135] => (item=rep1)

changed: [192.168.233.135] => (item=rep2)

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

192.168.233.135 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

xirtam@hostmanager:~/Formation-Ansible$

Enseignant dans l'enseignement supérieur depuis 2002, j'ai acquis une expérience de plus de 20 ans dans ce domaine. Pendant cette période, j'ai dispensé une variété de modules informatiques, allant des concepts fondamentaux tels que l'algorithmique, les bases de données et la programmation orientée objet, aux modules avancés comme l'architecture orientée services (SOA), le développement et l'exploitation (DevOps), ainsi que les tests et la validation des logiciels.

Actuellement, je suis également impliqué dans une petite entreprise de développement spécialisée dans les personnalisation des solutions informatique, basées principalement sur la technologie .NET. Cette entreprise se concentre sur le développement de solutions de contrôle d'accès et de gestion de parking.

En plus de mon engagement dans le secteur privé, j'ai animé des ateliers de formation professionnelle pour des professionnels travaillant en entreprise, des universités et des centres de formation. J'ai dispensé des formations dans des écoles d'ingénieurs et j'ai également participé à la formation de formateurs, dont les candidats étaient des enseignants du réseau ISET, dans le cadre d'un programme collaboratif avec l'association EFE. J'ai également contribué à la formation d'un groupe d'ingénieurs en reconversion professionnelle.

Mon objectif principal a toujours été de créer un environnement d'apprentissage stimulant et inclusif, où les participants peuvent acquérir de nouvelles compétences et développer leur compréhension de manière significative.

**Réponse** : L'adaptation de mon discours aux besoins et au niveau de compréhension des étudiants est une partie essentielle de mon approche pédagogique. Pour ce faire, je commencerais par évaluer le niveau de compréhension de mes étudiants en posant des questions ou en organisant des activités interactives pour sonder leur compréhension. Ensuite, je m'efforcerais d'utiliser un langage simple et clair, d'éviter les termes techniques complexes lors de l'introduction de nouveaux concepts, et d'illustrer les idées abstraites avec des exemples concrets et des analogies familières. En outre, je favoriserais un environnement d'apprentissage collaboratif où les étudiants peuvent poser des questions et discuter librement des sujets abordés.

**Quels sont les défis que vous pourriez rencontrer lors de l'enseignement de sujets techniques à un public non technique comme les étudiants en licence ?**

Réponse : L'un des défis principaux est de trouver le juste équilibre entre la simplification des concepts techniques pour les rendre accessibles et la préservation de leur précision et de leur pertinence. De plus, il peut être difficile de maintenir l'intérêt et l'engagement des étudiants tout au long de la session, surtout lorsqu'il s'agit de sujets complexes et abstraits. Enfin, il est important de reconnaître et de respecter la diversité des styles d'apprentissage et des expériences préalables des étudiants, en veillant à ce que chacun ait l'opportunité de réussir et de s'épanouir dans l'environnement d'apprentissage.

**Comment mesurez-vous le succès de vos sessions de formation ?**

Réponse : Je mesure le succès de mes sessions de formation à l'aide de plusieurs indicateurs. Tout d'abord, je recueille les retours des participants à travers des évaluations de cours anonymes, où je sollicite des commentaires sur la qualité du contenu, de la livraison et de l'organisation de la session. Ensuite, j'examine les résultats des évaluations formatives, telles que les quiz ou les exercices pratiques, pour évaluer la compréhension des participants des concepts enseignés. Enfin, je surveille les progrès des participants tout au long de la session, en observant leur niveau d'engagement, leur participation et leur capacité à appliquer les compétences nouvellement acquises dans des contextes pratiques.

**Pouvez-vous me parler d'une expérience où vous avez dû adapter votre communication à un public diversifié ?**

Réponse : Bien sûr. Lorsque j'ai animé des ateliers de formation pour des professionnels en entreprise, j'ai été confronté à un public diversifié en termes de niveaux de compétence technique, de domaines d'expertise et de contextes professionnels. Pour répondre à ces défis, j'ai adopté une approche flexible en adaptant mon langage et mes explications en fonction du niveau de compréhension de chaque groupe. J'ai également encouragé la participation active de tous les participants en posant des questions ouvertes et en favorisant les discussions en petits groupes. En fin de compte, cette approche a permis à tous les participants de s'engager pleinement dans le processus d'apprentissage et de tirer le meilleur parti de l'expérience de formation.

**Comment aborderiez-vous la gestion des différences de niveau de compréhension et d'expérience parmi les étudiants en licence terminale ?**

Réponse : Pour gérer les différences de niveau de compréhension et d'expérience parmi les étudiants, je commencerais par évaluer le profil de la classe pour comprendre les besoins individuels. Ensuite, j'adopterai une approche différenciée en proposant des activités et des ressources supplémentaires pour les étudiants ayant besoin d'un soutien supplémentaire, tout en offrant des défis supplémentaires aux étudiants plus avancés. Je favoriserais également la collaboration et le partage entre pairs pour encourager l'apprentissage entre pairs et la création d'une communauté d'apprentissage inclusive.

Video 1

Très bien, très bien, vous gagnez, je ferai une série ansible. Vous savez, cette série a été la série de tutoriels la plus demandée depuis un moment et c'est quelque chose que je voulais faire mais je n'ai jamais pu avoir le temps de le faire créer cette série mais finalement j'ai eu un peu de temps pour m'asseoir pour rédiger cette série et trouver un plan qui, je pense, fonctionne et nous voilà, je fais en fait la série ansible après tout ce temps où vous me demandez encore et encore. Je suis tellement heureux d'avoir enfin une chance de faire ça pour vous.

Maintenant, alors que je réfléchissais à la façon de faire cette série, je voulais essentiellement m'assurer de décrire Ansible d'une manière qui ait du sens. Et quand j'ai commencé avec ansible, j'étais un peu confus, je ne savais pas par où commencer et c'était tellement déroutant d'essayer de comprendre toutes les meilleures pratiques. Donc ce que j'ai décidé de faire, c'est d'aller de l'avant et de vous montrer ansible de la manière dont j'aimerais que quelqu'un le montre moi quand je viens de commencer ou quand j'ai commencé avec et je pense que les grandes lignes de cette série vont y parvenir. Donc ce que je vais faire maintenant, c'est vous donner quelques notes sur le démarrage de cette série va couler et le scénario que j'ai proposé et qui va servir de thème pour le reste de la série.

Allons de l'avant et vérifions-le pour l'essentiel, le but d'ansible est de provisionner les serveurs. Maintenant ici, j'ai trois serveurs, je ne leur ai pas vraiment donné de but, je nomme simplement le serveur un, le serveur deux et le serveur trois, je ne me sentais pas très créatif, mais l'idée est que nous voulons provisionner nos serveurs pour lesquels nous ne voulons pas avoir à effectuer de travail manuel. Faites-les tourner, nous voulons essentiellement automatiser autant que possible et c'est exactement ce qu'ansible nous permet de faire, donc le nombre de serveurs dont vous disposez est un peu arbitraire, vous pourriez avoir 500 serveurs dans votre flotte ou peut-être seulement trois comme nous le faisons ici. Donc le nombre n'a pas vraiment d'importance.

Maintenant, ce qui manque dans cette image ici est ansible, nous avons ces trois serveurs, mais où se situe ansible. Maintenant la plupart du temps, vous avez un hôte de contrôle ansible qui n'est fondamentalement qu'un autre mot pour un serveur ansible essentiellement le serveur ansible ou l'hôte de contrôle se connectera aux trois serveurs dans ce cas ou quel que soit le nombre de commandes que vous devrez émettre pour le provisionner maintenant, les trois serveurs sur lesquels ils n'ont pas réellement besoin d'avoir ansible installé, donc il n'y a pas d'ansible installé ici ou ici ou ici nous n'avons qu'ansible sur l'hôte de contrôle, puis le serveur sensible établira des connexions via ssh aux serveurs pour continuer et les provisionner.

Mais la chose la plus déroutante pour moi quand j'ai commencé avec ansible est qu'il n'y a jamais eu de guide que j'ai eu à ce moment-là, cela m'a vraiment expliqué cela et j'ai dû trouver tout cela par moi-même et ce que j'ai découvert, c'est qu'il n'y a pas une seule façon de configurer ansible ansible. J'aime donner l'équation selon laquelle c'est comme si quelqu'un jetait un tas de legos sur vos genoux et ils vous disent de le construire comme vous le souhaitez maintenant, évidemment avec ansible, il existe de meilleures pratiques, il y a des choses que vous voulez vous assurer de faire, il existe des moyens efficaces de faire les choses de manières inefficaces, mais il n'y a pas une seule bonne façon de faire les choses, vous n'avez donc pas besoin d'avoir un hôte de contrôle ansible ou un serveur ansible. Il existe un scénario alternatif et c'est le scénario que nous allons suivre dans cette série et c'est ce scénario ici. Maintenant, au lieu d'avoir un hôte de contrôle Ansible, nous avons mon ordinateur portable, nous avons un poste de travail et essentiellement, ce sera cet ordinateur portable qui contiendra les scripts ou les playbooks qui seront utilisés pour exécuter Ansible sur les serveurs. Tout va se passer à partir de cet ordinateur portable maintenant, cela pourrait être un ordinateur de bureau, cela n'a pas vraiment d'importance car cela fonctionne dans le monde réel si vous pouvez avoir un référentiel git et à l'intérieur de ce référentiel get se trouvent tous vos playbooks ansibles et allons-y. Dites simplement que vous avez d'autres personnes qui travaillent avec vous, d'autres administrateurs dont ils ont besoin pour pouvoir télécharger ce référentiel pour pouvoir également exécuter les playbooks sur les serveurs et généralement dans ses magasins de nos jours, tout cela est collaboratif, vous avez une personne qui fera modifications, les autres administrateurs téléchargeront ces modifications et pourront également les exécuter et c'est essentiellement ce que nous allons faire ici encore. Nous avons cet ordinateur portable ici, donc je viens de le nommer l'ordinateur portable de Jay, il y aura les playbooks là-bas et ils vont être stockés dans git et c'est important parce que si quelque chose arrive à cet ordinateur portable, peut-être qu'il plante, le disque dur meurt quelque chose comme ça, alors tous ces playbooks ansibles l'accompagneront et mourront aussi, nous ne voulons pas que cela ait tout dans git signifie que nous sommes raisonnablement protégés contre des choses comme ça, mais cela signifie également que nous pouvons travailler en collaboration avec d'autres personnes de notre équipe pour bénéficier de la nature de contrôle de version de git ainsi que de la puissance d'ansible.

Maintenant dans mon cas. J'ai créé trois machines virtuelles pour simuler cela. J'ai par exemple Ubuntu 2004, puis Ubuntu 2042 et ensuite Ubuntu 2004 iii. Maintenant, normalement, je n'aime pas utiliser de machines virtuelles sur ma chaîne parce que j'aime vous montrer tout, du bare metal, mais quand il s'agit de vous apprendre Ansible, je pense que ce scénario fonctionne très bien pour les machines virtuelles, nous allons donc configurer ces machines virtuelles avec ansible, puis nous allons le faire à partir d'une autre machine virtuelle que j'appelle ma machine virtuelle de poste de travail qui le fera. simulez votre ordinateur portable ou de bureau, puis nous allons continuer et l'utiliser pour exécuter toutes les commandes sur ces trois serveurs et je vous expliquerai toutes les bases comme l'installation de packages créant des utilisateurs jusqu'à la création de rôles. Ainsi, vous apprendrez tout ce que vous devez savoir pour être productif avec ansible dans cette série et c'est essentiellement le plan que nous allons suivre en premier. Je vais vous montrer comment utiliser les clés ssh pour vous authentifier auprès de votre serveurs qu'il est important de savoir, je vais également vous montrer comment créer un référentiel git, le retirer, apporter des modifications et le repousser vers master, puis je vais vous montrer comment installer ansible et puis à partir de là, nous sommes je vais aller de plus en plus profondément dans le monde d'ansible jusqu'à ce que vous en sachiez assez pour être productifs maintenant, la première série de vidéos devrait déjà être disponible sur ma chaîne si je ne les ai pas déjà toutes au moment où vous regardez alors allez-y, cliquez sur la deuxième vidéo et commençons.

Video 2

"Bonjour et bienvenue à tous dans ma série sur Ansible. Je sais que vous êtes impatients de commencer et de plonger dans le monde d'Ansible, mais dans cette vidéo, nous devons d'abord couvrir un concept de base très important avant de pouvoir avancer et faire cela, et ce concept est OpenSSH.

OpenSSH est à peu près le standard par défaut en matière d'administration à distance sous Linux. C'est ce que nous utilisons pour nous connecter à un serveur Linux afin d'exécuter des commandes, de réaliser certaines tâches d'administration, et c'est aussi ce qu'Ansible utilise pour sa provision. Dans cette vidéo, je vais vous donner un aperçu de base de ce qu'est SSH et aussi comment effectuer une gestion de clés de base, ce qui est très important en ce qui concerne Ansible. Aucun des concepts que je vais aborder dans cette vidéo n'est spécifique à Ansible, mais il est très important que vous ayez une compréhension de base de ces sujets avant que nous ne continuions.

Jetons un coup d'œil rapide à OpenSSH, en particulier à la manière de gérer les clés. Vous voudrez certainement comprendre comment faire cela, mais nous allons couvrir tout cela et plus encore dans cette vidéo. Allons-y et commençons.

Avant de commencer, je veux juste m'assurer que nous avons une compréhension complète de ce que nous essayons de réaliser dans cette vidéo. Ce que nous essayons de faire pour cet épisode, c'est tout ce qui concerne OpenSSH ou plus simplement SSH. Techniquement, cela n'a rien à voir avec Ansible, mais puisque Ansible utilise SSH, il est important que nous ayons au moins une compréhension fondamentale de ce que c'est et comment l'utiliser.

Revenons à notre thème ici. Nous avons une station de travail, évidemment, cela pourrait être un ordinateur de bureau, un ordinateur portable, peu importe, mais fondamentalement, si vous êtes un administrateur Linux, vous utiliserez très probablement SSH pour vous connecter à vos serveurs et faire votre travail. Fondamentalement, SSH est très facile à utiliser, c'est généralement juste une question d'utiliser la commande SSH, puis de lui donner une adresse IP ou un nom d'hôte d'un serveur auquel vous voulez vous connecter, puis soit vous serez invité pour le mot de passe, soit si vous avez une clé SSH, vous ne serez pas invité pour un mot de passe et vous vous connecterez directement au serveur.

L'authentification basée sur un mot de passe est la norme, mais il est toujours préférable d'utiliser des clés SSH, qui sont facultatives, mais essentiellement, elles aident à sécuriser votre connexion et rendent beaucoup plus difficile pour quelqu'un de l'extérieur de s'introduire. SSH est assez sécurisé en soi, mais pour être honnête, si vous avez un mot de passe facile, il suffit d'une question de temps avant que quelqu'un de l'extérieur ne parvienne à craquer ce mot de passe. Mais l'utilisation d'une clé SSH rend en fait les choses beaucoup plus difficiles et est fortement recommandée.

Tout d'abord et avant tout, OpenSSH est un prérequis pour Ansible. C'est requis, car encore une fois, c'est ainsi qu'Ansible fait sa magie. Il utilise OpenSSH pour se connecter à chacun des serveurs et les provisionner sur la station de travail ou sur n'importe quelle machine à partir de laquelle vous exécutez Ansible, tout ce dont vous aurez besoin est le client SSH, et si vous exécutez Linux, cela vient par défaut, il ne devrait pas y avoir besoin de faire quoi que ce soit sur la station de travail. En ce qui concerne les serveurs, le serveur OpenSSH est en fait facultatif. Si le serveur est un Linode, un droplet ou une sorte de VPS, alors vous aurez en fait le serveur SSH installé par défaut, et vous serez prêt à partir. Si vous configurez une installation manuelle d'un serveur Ubuntu à partir du support d'installation, alors une partie du processus d'installation vous demandera en fait si vous voulez ou non le serveur OpenSSH. Si vous avez répondu oui, vous êtes prêt, si vous n'avez pas répondu oui, alors vous devrez essentiellement installer le paquet serveur OpenSSH sur les serveurs pour vous assurer qu'ils l'ont.

Une fois que nous avons vérifié que nous avons OpenSSH installé, la prochaine chose à faire est de faire cette première connexion depuis la station de travail via SSH vers chacun des serveurs. La raison pour laquelle nous devons nous occuper de cela est que la première chose qu'elle va faire est de vous demander si vous êtes sûr de vouloir vous connecter, puis vous répondrez oui, et alors la connexion se fera, vous ne serez pas à nouveau invité pour cela. C'est seulement la première fois que vous vous connectez, donc nous voulons essentiellement nous assurer de prendre soin de cette première connexion sur chacun des serveurs, puis répondre à cette invitation. La raison pour laquelle nous faisons cela est que cette invitation peut en fait embrouiller Ansible, il y a en fait une façon de faire en sorte qu'Ansible accepte automatiquement la connexion initiale, mais cela dépasse le cadre de la vidéo, il est plus facile de simplement faire la première connexion vous-même depuis l'ordinateur portable, la station de travail, quoi que vous ayez, vers les serveurs pour en finir avec cela.

Le prochain sujet que nous allons aborder dans cette vidéo est comment créer une paire de clés SSH, et ce n'est pas obligatoire, mais c'est fortement recommandé. C'est une excellente pratique à adopter et cela renforce essentiellement votre connexion SSH, donc c'est généralement une excellente idée. La clé que nous créons pour notre compte utilisateur, qui est la première des deux clés que nous allons créer, aura une phrase secrète qui la rendra encore plus sécurisée. Maintenant, nous n'avons pas vraiment besoin d'une clé SSH pour notre compte utilisateur normal, et la création d'une clé pour notre compte utilisateur n'a rien à voir avec Ansible, c'est juste une bonne pratique à avoir, votre propre clé en plus de toute autre clé que vous pourriez créer à d'autres fins. Après avoir créé cette clé, je vais vous montrer comment la copier sur chaque serveur pour que nous puissions réellement l'utiliser, puis nous allons créer une clé SSH spécifique à Ans

Haut du formulaire

Video 10 :

Nous avons déjà beaucoup avancé dans la série Ansible jusqu'à présent. Dans la dernière vidéo, je vous ai montré comment cibler des hôtes spécifiques, ou plutôt comment regrouper les hôtes en groupes individuels, de sorte que vous ayez différentes catégories ou finalités pour ces serveurs. Mais ce qui n'est pas si génial, c'est qu'à chaque fois que nous voulons tester le playbook, nous devons l'exécuter contre chaque serveur à chaque fois. Dans cette vidéo, je vais vous montrer comment utiliser des tags pour ajouter essentiellement des métadonnées à nos tâches, de sorte que nous puissions exécuter uniquement les tâches que nous voulons tester, ce qui va faciliter le test de nos playbooks.

Tout d'abord, j'ajoute des tags à ce playbook. Voyons maintenant comment les tags fonctionnent. Pour commencer, nous avons nos pré-tâches ici, ce sont des tâches qui seront exécutées en premier, quoi qu'il arrive. Ajoutons des tags "always" à ces tâches. Je vais faire la même chose pour celle-ci. Ensuite, dans la section des serveurs web, ajoutons des tags aux tâches. Faisons la même chose pour la version CentOS de la même tâche. Et plus bas, nous avons nos serveurs de base de données. Ajoutons des tags à cette tâche ici. Ensuite, encore une fois ici. Enfin, nous avons cette dernière tâche où nous avons simplement ajouté le package Samba à notre unique serveur de fichiers. Je vais simplement ajouter un tag "samba" à celui-ci pour simplifier.

Revenons maintenant au début. J'ai ajouté des tags "always" ici, car je veux que ces tâches s'exécutent toujours, je veux toujours que les serveurs soient à jour et entièrement patchés. Donc, quel que soit le tag que je cible, celles-ci devraient toujours s'exécuter. Mais plus bas, nous avons essentiellement une ligne qui dit "tags", et j'ai simplement ajouté quelques tags séparés par des virgules. Vous pouvez ajouter les tags que vous voulez, j'ai simplement ajouté "apache" parce que cela concerne cela, "apache2" car c'est le nom du package, "ubuntu" car c'est la distribution. Vous pouvez voir que j'ai fait la même chose ici pour CentOS, Apache, le nom de la distribution CentOS et le nom du package Apache. Vous pourriez venir avec votre propre système de tags, cela n'a vraiment pas d'importance. Et ensuite, ici, j'ai fait essentiellement la même chose avec les autres et ajouté les tags pour ceux-ci également.

Sauvegardons le fichier. Exécutons le playbook pour nous assurer que tout fonctionne correctement. Fondamentalement, nous voulons juste nous assurer qu'il n'y a pas d'erreurs de syntaxe. Je l'exécute contre tout, je ne lui ai pas réellement donné de tags à cibler, et rien n'a changé, mais surtout, rien n'a échoué. Nous savons que tout fonctionne maintenant.

Maintenant, voyons quels tags sont en fait disponibles. Nous savons quels tags sont disponibles parce que nous les avons ajoutés manuellement, mais examinons une commande spéciale que nous pourrions exécuter si nous avions besoin d'un rappel. Nous faisons ansible-playbook, comme nous l'avons toujours fait, puis --list-tags, puis le nom du playbook, qui pour nous est juste site.yml. Il liste les tags que nous avons disponibles. Par exemple, si je veux cibler le tag CentOS, je pourrais exécuter ansible-playbook --tags, puis cibler CentOS. J'ai toujours besoin du double tiret demander le mot de passe, et puis site.yml. J'entre le mot de passe magique, voyons ce qui se passe.

Parcourons la sortie ici et voyons ce qui s'est réellement passé. Tout d'abord, il a installé les mises à jour sur CentOS et Ubuntu. Nous avons appliqué le tag "always" à ceux-ci pour nous assurer que cela se produit toujours. Mais ensuite, lorsque je fais défiler un peu, nous pouvons voir qu'il a recueilli des faits pour nos serveurs web, mais il n'a réellement été exécuté que sur le serveur CentOS ici. Il n'a même pas tenté de s'exécuter sur le serveur Ubuntu qui a été identifié dans le fichier d'inventaire. En ce qui concerne nos serveurs de base de données, nous n'avons que ce serveur avec cette adresse IP, nous n'avons même pas de système CentOS comme membre de ce groupe, donc il a été ignoré car cet hôte ici exécute Ubuntu. Et en ce qui concerne les serveurs de fichiers, il dit qu'il recueille des faits, mais il n'a réellement rien exécuté. Comme vous pouvez le voir, il n'a exécuté que les tâches ayant le tag que j'ai identifié dans la ligne de commande. Nous avons pu exécuter uniquement les tâches que nous voulions exécuter. Par exemple, si je testais une sorte de changement de syntaxe sur des systèmes CentOS ou si je voulais spécifiquement cibler le playbook entier vers des systèmes CentOS pour lesquels j'avais créé le tag CentOS manuellement, j'ai pu utiliser cette commande avec les tags --tags et ensuite CentOS pour cibler spécifiquement cela.

Si, par exemple, je voulais tester le serveur de base de données, c'est un autre tag que j'ai ajouté, donc cela devrait fonctionner très bien. Et comme vous pouvez le voir, cela a fonctionné très bien. Ensuite, je pourrais faire la même chose mais avec le tag "apache" pour faire exactement cela. Maintenant, que faisons-n