

## Rapport de Projet de stage d'été

# **Ingénierie des Systèmes Informatiques " Computer Engineering "** **Spécialité : Ingénierie des Réseaux et Systèmes** **Par** **Nidhal Ghazouani**

---

*Étude et Mise en place d'une plateforme d'automatisation à base de  
Redhat*

---

**Encadrant professionnel :** Ghazi Oueslati

*Réalisé au sein de Tunisie Télécom*



# SOMMAIRE

<b>Introduction Générale</b> .....	3
<b>I-Description du projet</b> .....	3
I.1-Contexte et justification .....	3
I.2-Problématique.....	4
<b>II -Analyse du projet</b> .....	4
II.1-C'est quoi Ansible ? .....	4
II.2-Pourquoi Ansible ? .....	5
<b>III-Fonctionnement</b> .....	5
III.1-Protocole SSH .....	5
III.2-fichier inventaire d'Ansible.....	6
III.2-fichier de configuration d'Ansible .....	7
III.3-Patterns .....	8
III.4-Commande ad-hoc.....	9
III.4.1-Les Modules .....	11
III.5-Les playbooks .....	12
<b>IV-Réalisation</b> .....	19
IV.1-Installation d'Ansible.....	19
IV.2-Architecture.....	20
IV.3 -Déploiement.....	20
Installation et configuration d'un serveur apache.....	20
<b>Conclusion</b> .....	27

## Liste des figures

<b>Figure 1 : Génération de la clé sur Control node</b> .....	6
<b>Figure 2 : Fichier inventaire d'Ansible</b> .....	7
<b>Figure 3 : Fichier de configuration d'Ansible (ansible.cfg)</b> .....	8
<b>Figure 4: Afficher tous les modules</b> .....	11
<b>Figure 5 :ACCÉDER À LA DOCUMENTATION D'UN MODULE</b> .....	12
<b>Figure 6 :Architecture</b> .....	20
<b>Figure 7: Configuration Ansible</b> .....	21
<b>Figure 8 :Fichier inventaire d'Ansible (Inventory File)</b> .....	21
<b>Figure 9 : Tester la connectivité entre les Machines</b> .....	22
<b>Figure 10 : Test.yml</b> .....	23
<b>Figure 11 : Exécution de playbook</b> .....	23
<b>Figure 12 : Installer_httpd.yml</b> .....	24
<b>Figure 13 : Verifier le syntaxe du playbook</b> .....	24
<b>Figure 14 : Exécution de Playbook</b> .....	25

<b>Figure 15 : Vérifier l'état de la serveur Apache .....</b>	<b>25</b>
<b>Figure 16 : Page par défaut Apache http server CentOS .....</b>	<b>26</b>
<b>Figure 17 : Supprimer le serveur http.....</b>	<b>26</b>
<b>Figure 18 : Créer Utilisateur et Group .....</b>	<b>26</b>

## Introduction Générale

Le présent rapport entre dans le cadre de stage d'été en Tunisie Télécom

Les systèmes informatiques sont devenus des outils indispensables au fonctionnement des entreprises. Ils sont aujourd'hui déployés dans tous les secteurs professionnels, à savoir, le secteur bancaire, les assurances, la médecine et d'autres.

Au fur et à mesure que l'entreprise s'agrandit, le nombre de projets qui apparaissent augmente. Par conséquent, le nombre de serveurs à gérer par l'administrateur augmente de jours en jours par rapport au nombre de projets qui apparaissent. Ainsi l'administrateur a plusieurs serveurs à gérer. Ces serveurs peuvent être virtuels ou physiques.

Dès lors, l'administrateur a besoin d'un outil efficace pour pouvoir effectuer plus facilement les tâches d'administration tels que l'installation d'un logiciel, l'arrêt et le redémarrage d'un service, automatisation des mises à jour, effectuer les configurations, créer des utilisateurs en plus d'autres à distance sur plusieurs serveurs parallèlement. Ce qui évitera à l'administrateur de se déplacer sur chaque serveur pour faire des configurations.

Dans ce document nous utiliserons Ansible qui est une plate-forme logicielle libre pour effectuer les tâches d'administration à distance à travers le protocole sécurisé SSH.

Ce Rapport est structuré en quatre parties:

- Dans la première partie nous allons faire une description du projet
- Dans la deuxième nous ferons une analyse du projet
- Dans la troisième partie nous étudierons le fonctionnement de Ansible
- Et enfin dans la quatrième partie nous exposerons notre réalisation.

## I-Description du projet

### I.1-Contexte et justification

Administrer c'est de gérer quelque chose. Nous pouvons dire qu'administrer un système informatique c'est la gestion d'un parc informatique. Qu'il soit en

entreprise ou en situation d'utilisation personnelle, l'administrateur système effectue des tâches communes : Installer, configurer et faire évoluer le matériel : une nouvelle carte réseau, un nouveau disque ; Installer les applications : installation sur un ou plusieurs postes en même temps ; gérer les utilisateurs : ce qui comprend l'ajout, la suppression d'un profil, la mise en place de quotas, la gestion des droits spécifiques ou par groupe et en plus sécuriser le système : par les mises à jours liées à la distribution ainsi que celles liées aux applications, par les sauvegardes régulières des données. Depuis le début de l'informatique en réseau, le déploiement et la gestion fiables restent un défi. Historiquement, les administrateurs système ont généralement géré les serveurs, installer des logiciels, modifier des configurations et administrer des services manuellement. Mettre en place un serveur manuellement peut rapidement s'avérer pénible, surtout si nous avons plusieurs machines à configurer de la même manière. La virtualisation des serveurs a aussi considérablement augmenté la tâche d'un administrateur ou d'une équipe d'administrateurs. Face au nombre de serveurs qui augmente au fil des années dans l'entreprise, ce serait fastidieux de se déplacer sur les serveurs un à un pour leurs configurations et surtout si le serveur est sur un site distant. Le fait de se déplacer sur les serveurs entraîne une perte de temps et par conséquent le retard dans le service offert aux utilisateurs. Ansible est un outil de gestion de Configuration et de déploiement, permettant d'automatiser simplement et efficacement la mise en place d'infrastructures complexes.

## **I.2-Problématique**

Le problème qui se pose est de trouver une solution pour effectuer les tâches d'administration de façon efficace, éliminer les contraintes géographiques, gagner du temps en gérant plusieurs serveurs en même temps c'est-à-dire de façon parallèle.

## **II -Analyse du projet**

Une solution pour une gestion optimisée des serveurs d'une entreprise est : Ansible.

### **II.1-C'est quoi Ansible ?**

Ansible est un outil de pilotage de systèmes, sans agent, faisant partie de la famille des outils DevOps. Il permet de simplifier des opérations d'orchestration complexes, de faire du management de configuration centralisé sur un grand nombre de machines. Il permet également le pilotage de plateformes Cloud telles qu'Amazon Web Services ou OpenStack. Il est un moteur d'automatisation informatique radicalement simple qui automatise la gestion de configuration, le déploiement d'application, intra-orchestration de services. Il utilise YAML qui est un langage très simple qui permet de décrire les travaux d'automatisation d'une manière simple qui

se rapproche du pur anglais. Les modules Ansible fonctionnent via JSON. Ansible est extensible avec des modules qu'on peut écrire dans n'importe quel langage de programmation.

## II.2-Pourquoi Ansible ?

Ansible a plusieurs avantages dont nous citons quelques-uns :

- Gagnez du temps et être plus productif
- Moins d'erreurs
- Améliorer la collaboration et la satisfaction professionnelle
- surmonter la complexité
- Plus de ressources pour l'innovation
  
- Éliminer les tâches répétitives

## III-Fonctionnement

### III.1-Protocole SSH

Ansible utilise le protocole SSH pour communiquer de façon sécurisée avec les machines à configurer. Ainsi pour utiliser ansible, il faut installer openssh et générer une clé sur la machine contrôleur qui sera ensuite copiée sur les serveurs distants sur lesquels est installé aussi le serveur openssh. Ces étapes peuvent être réalisées de la façon suivante :

#### Installation de openssh sur les machines :

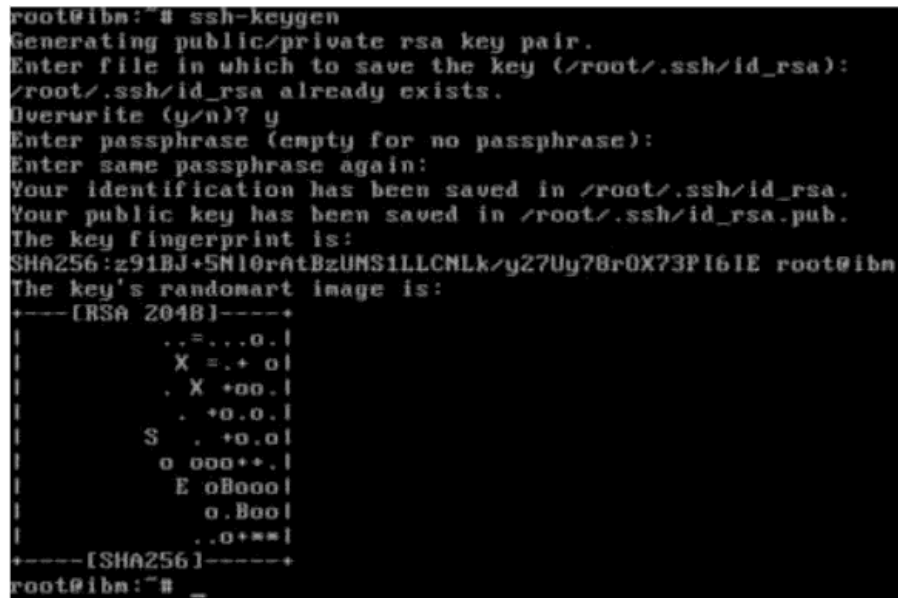
```
$ sudo apt-get install openssh-server openssh-clients
```

Remplissage du fichier /etc/hosts sur la machine contrôleur : supposons qu'on a deux machines à gérer. On aura dans le fichier /etc/hosts de la machine contrôleur par exemple ceci:

```
192.168.1.13      node1
192.168.1.14      node2
```

## Génération de la clé sur la machine contrôleur :

```
$ ssh-keygen
```

A terminal window showing the execution of the 'ssh-keygen' command. The output includes prompts for file location, passphrase, and key fingerprint. The key is saved in /root/.ssh/id\_rsa. The key fingerprint is SHA256:z91BJ+5N10rAtBzUMS1LLCNLk/y27Uy78r0X73Pl6IE. The key's randomart image is displayed as a series of characters representing the key's visual representation.

```
root@ibn:~# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:z91BJ+5N10rAtBzUMS1LLCNLk/y27Uy78r0X73Pl6IE root@ibn
The key's randomart image is:
+---[RSA 2048]-----+
|  ..=...o.|
|    X =. + o|
|  . X +oo.|
|    . +o.o.|
|    S . +o.o|
|    o ooo+|.
|    E oBooo|
|    o.Boo|
|    .o+==|
+----[SHA256]-----+
root@ibn:~#
```

*Figure 1 : Génération de la clé sur Control Node*

## Copie de la clé sur les serveurs distants:

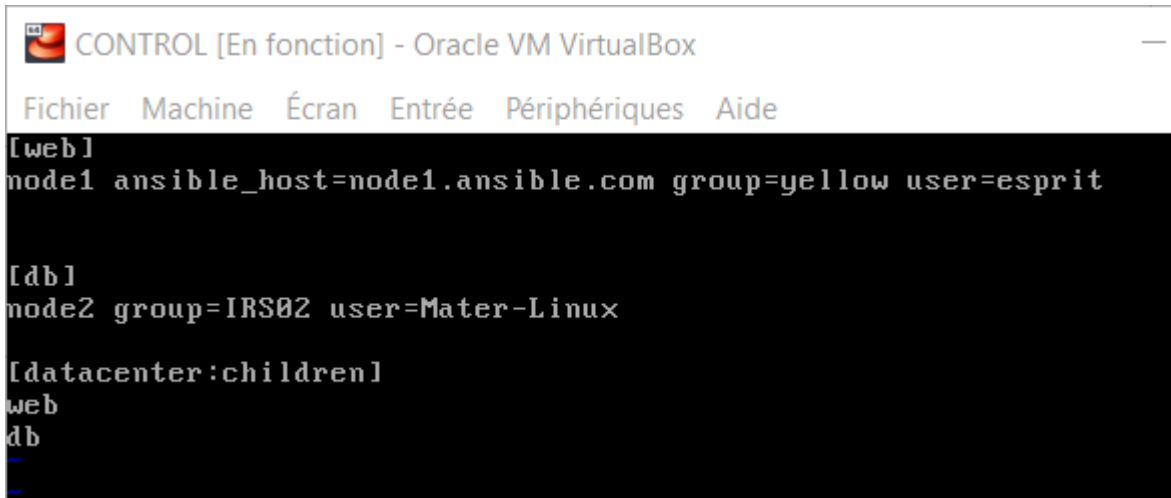
```
$ ssh-copy-id node1
$ ssh-copy-id node2
```

Après cette configuration, on pourra ensuite se connecter aux serveurs de façon sécurisée.

## III.2-fichier inventaire d'Ansible

Le fichier inventaire de ansible comporte ou contient les hôtes du réseau. Ces hôtes peuvent être uniques ou groupés en utilisant les crochets. Le fichier inventaire par défaut est /etc/ansible/hosts. Par ce fichier, ansible peut travailler sur plusieurs systèmes d'une infrastructure en même temps. Nous pouvons spécifier un fichier d'inventaire différent en utilisant l'option -i <chemin> sur la ligne de commande. Nous pouvons également utiliser plusieurs fichiers d'inventaire en même temps.

Ce fichier peut se présenter de la façon suivante



```
CONTROL [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
[web]
node1 ansible_host=node1.ansible.com group=yellow user=esprit

[db]
node2 group=IRS02 user=Mater-Linux

[datacenter:children]
web
db
```

*Figure 2 : Fichier inventaire d'Ansible*

#### Explication du fichier :

Dans ce fichier nous avons *node1* comme nom d'hôte. Nous avons aussi deux groupes: [web] et [db]. Le groupe **[web]** contient tous les serveurs web du réseau pendant que nous avons mis dans le groupe [db] les serveurs databases. Notons cette répartition est faite par l'administrateur selon ses besoins et le but qu'il poursuit. Un serveur peut appartenir à plusieurs groupes.

### III.2-fichier de configuration d'Ansible

On peut changer ces variables de configuration en renseignant un fichier de configuration. Ansible cherchera dans l'ordre (le premier trouvé sera utilisé et les autres seront ignorés):

1. Le contenu de la variable d'environnement ANSIBLE\_CONFIG (Si la variable d'environnement est valorisée)
2. L'emplacement ./ansible.cfg (dans le dossier courant, le répertoire de travail)
3. L'emplacement ~/.ansible.cfg (à la racine du dossier utilisateur comme fichier caché)
4. L'emplacement /etc/ansible/ansible.cfg (dans le dossier de configuration du logiciel)

La commande `ansible-config view` permet de visualiser le contenu du fichier de configuration courant.

Ce fichier peut se présenter de la façon suivante :



```

[ansible@Control project1]$ cat ansible.cfg
[defaults]
inventory = inventory
remote_user = ansible
host_key_checking = false
INJECT_FACTS_AS_VARS = False
roles_path= /home/ansible/project1/roles
[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ack_pass = false

```

*Figure 3 : Fichier de configuration d'Ansible (ansible.cfg)*

### III.3-Patterns

Patterns dans ansible est la façon dont nous décidons gérer les hôtes du réseau. Cela signifie que nous décidons de quel hôte nous voulons communiquer.

Pour utiliser Ansible, nous devons savoir comment dire à Ansible de communiquer avec les hôtes qui sont dans le fichier inventaire. Ceci est fait en désignant particulièrement le nom des hôtes ou un groupe de serveurs.

Les patterns suivants sont Equivalents et cible tous les hôtes du fichier inventaire :

```
all
```

```
*
```

Il est aussi possible d'adresser un hôte spécifique ou des hôtes par leur nom :

```

node1
node1.ansible.com
node2
192.168.1.13
192.168.1. 14

```

Nous pouvons cibler les hôtes par leurs positions dans un groupe.

Supposons-nous avons le groupe suivant :

```

[webservers]
Cobweb
Webbing weber

```

Nous pouvons référencer les hôtes par le groupe de la façon suivante :

```
webservers[0]      # == cobweb
webservers[1]      # == webbing
webservers[2]      # == weber
webservers[0:1]    # == cobweb, webbing
```

### III.4-Commande ad-hoc

Une commande ad-hoc est une commande qu'on peut taper pour faire quelque chose de rapide, mais qu'on ne veut pas enregistrer. D'une manière générale, la vraie puissance de Ansible réside dans les playbooks mais nous pouvons utiliser les commandes ad hoc pour de petite et simple tâche. Donnons quelques exemples :

D'une manière générale une commande ad-hoc Ansible se présente sous cette forme :

- *ansible <HOST\_GROUP> -m <module\_name> -a <arguments>*

ansible: est une commande

<HOST GROUP> : représente les hôtes sur lesquels on veut exécuter la commande

-m : permet de spécifier un module.

-a : permet de spécifier les arguments

- *ansible datacenter -m ping*

Cette commande va exécuter le module de ping vers les serveurs qui sont dans le groupe datacenter

#### Commande ad-hoc en utilisant le module de copie:

- *ansible datacenter -m copy -a "src=/root/ansible.txt dest=/tmp/" -f 2*

On copie le fichier "/root/ansible.txt" placé sur la machine ansible à tous les hôtes définis dans "datacenter" dans /tmp pour tous les hôtes. Le -f 2 option indique à ansible d'ouvrir deux processus pour exécuter cette option.

#### Commande ad-hoc en utilisant le module file:

En utilisant le module de fichier, on peut également modifier les propriétés des fichiers sur les hôtes.

- *ansible all -m file -a "dest=/tmp/ansibletest.sh mode=755"*

Définit l'autorisation de fichier à 755 sur tous les hôtes pour le fichier

/tmp/ansibletest.sh

```
- ansible all -m file -a "dest=/tmp/monster state=directory  
mode=755"
```

Ceci créera le répertoire /tmp/monster sur tous les hôtes avec 755 autorisations.

```
- ansible all -m file -a "dest=/tmp/monster state=absent"
```

Ceci supprimera le répertoire "/tmp/monster" récursivement

#### **Commande ad hoc en utilisant le module de gestion de package:**

```
- ansible all -m yum -a "name=docker  
state=latest"
```

S'assurer que le paquetage docker est le dernier.

#### **Commande ad hoc en utilisant le module de gestion des utilisateurs:**

- Le module "utilisateur" de Ansible peut être utilisé pour ajouter et supprimer des hôtes du système facilement. Pour ajouter un utilisateur, nous devons envoyer un mot de passe chiffré comme indiqué ci-dessous.

```
ansible all -m user -a  
"name=monster password=$6$random_salt$Bn0QxEG8Gk2rzFYwoWXjr  
59zLVYzwshvca5oV0PtU8fAfT4a571evgca .  
E0hLnYNCdfq//zw9Yy$N33ÇttztI10 "
```

- On peut supprimer un utilisateur de cette manière

```
ansible all -m user -a "name=monster state=absent"
```

- Pour supprimer le répertoire d'accueil des utilisateurs après avoir supprimé l'utilisateur, le module 'fichier' peut être utilisé comme indiqué ci-dessous

```
ansible all -m file -a "dest=/home/monster

state=absent"
```

Commande ad hoc en utilisant le module de gestion des services:

- Ceci va s'assurer si le service httpd est en cours d'exécution

```
ansible node1 -m service -a "name=httpd state=started"
```

- Ceci va s'assurer si le service httpd est redémarré

```
ansible node2 -m service -a "name=httpd state=restarted"
```

### III.4.1-Les Modules

Bouts de code copiés sur le système cible. Exécutés pour satisfaire à la déclaration de tâche Personnalisables

# AFFICHE TOUS LES MODULES :

- ansible-doc -l

```
fortios_router_community_list    Configure community lists in Fortinet's FortiOS and FortiGate
azure_rm_devtestlab_info        Get Azure DevTest Lab facts
ecs_taskdefinition              register a task definition in ecs
avi_alertscriptconfig            Module for setup of AlertScriptConfig Avi RESTful Object
tower_receive                   Receive assets from Ansible Tower
netapp_e_iscsi_target            NetApp E-Series manage iSCSI target configuration
azure_rm_acs                    Manage an Azure Container Service(ACS) instance
fortios_log_syslogd2_filter      Filters for remote system server in Fortinet's FortiOS and FortiGate
junos_rpc                       Runs an arbitrary RPC over NetConf on an Juniper JUNOS device
na_elementsw_vlan               NetApp Element Software Manage VLAN
pn_ospf                         CLI command to add/remove ospf protocol to a vRouter
pn_snmp_vacm                    CLI command to create/modify/delete snmp-vacm
cp_mgmt_service_sctp            Manages service-sctp objects on Check Point over Web Services API
onyx_ospf                       Manage OSPF protocol on Mellanox ONYX network devices
icx_command                     Run arbitrary commands on remote Ruckus ICX 7000 series switches
cs_snapshot_policy              Manages volume snapshot policies on Apache CloudStack based clouds
nxos_install_os                 Set boot options like boot, kickstart image and issu
cnos_static_route               Manage static IP routes on Lenovo CNOS network devices
win_eventlog                    Manage Windows event logs
vmware_category                 Manage VMware categories
vmware_host_feature_info        Gathers info about an ESXi host's feature capability information
avi_cluster                     Module for setup of Cluster Avi RESTful Object
na_ontap_user                   NetApp ONTAP user configuration and management
aci_l3out                       Manage Layer 3 Outside (L3Out) objects (l3ext:Out)
memset_server_info              Retrieve server information
gcp_compute_subnetwork_info     Gather info for GCP Subnetwork
azure_rm_virtualmachinescalesetextension Manage Azure Virtual Machine Scale Set (VMSS) extensions
fortios_report_dataset          Report dataset configuration in Fortinet's FortiOS and FortiGate
avi_api_session                 Avi API Module
avi_networkprofile              Module for setup of NetworkProfile Avi RESTful Object
avi_backup                      Module for setup of Backup Avi RESTful Object
aci_interface_policy_cdp        Manage CDP interface policies (cdp:IfPol)
fortios_firewall_vip            Configure virtual IP for IPv4 in Fortinet's FortiOS and FortiGate
gcp_compute_backend_service     Creates a GCP BackendService
iam_policy                      Manage IAM policies for users, groups, and roles
fortios_system_fips_cc          Configure FIPS-CC mode in Fortinet's FortiOS and FortiGate
fortios_log_null_device_setting Settings for null device logging in Fortinet's FortiOS and FortiGate
win_inet_proxy                  Manages proxy settings for WinINet and Internet Explorer
fortios_firewall_auth_portal    Configure firewall authentication portals in Fortinet's FortiOS and FortiGate
jenkins_plugin                  Add or remove Jenkins plugin
iosxr_interfaces                 Manage interface attributes on Cisco IOS-XR network devices
:
```

*Figure 4: Afficher tous les modules*

# ACCÉDER À LA DOCUMENTATION D'UN MODULE

- ansible-doc <module\_name>

```

[ansible@Control ~]$ ansible-doc user
> USER (/usr/lib/python2.7/site-packages/ansible/modules/system/user.py)

    Manage user accounts and user attributes. For Windows targets, use the [win_user] module instead.

    * This module is maintained by The Ansible Core Team
OPTIONS (= is mandatory):

- append
    If 'yes', add the user to the groups specified in 'groups'.
    If 'no', user will only be added to the groups specified in 'groups', removing them from all other groups.
    Mutually exclusive with 'local'
    [Default: False]
    type: bool

- authorization
    Sets the authorization of the user.
    Does nothing when used with other platforms.
    Can set multiple authorizations using comma separation.
    To delete all authorizations, use 'authorization=''.
    Currently supported on Illumos/Solaris.
    [Default: (null)]
    type: str
    version_added: 2.8

- comment
    Optionally sets the description (aka 'GECOS') of user account.
    [Default: (null)]
    type: str

- create_home
    Unless set to 'no', a home directory will be made for the user when the account is created or if the home directory does not exist.
    Changed from 'createhome' to 'create_home' in Ansible 2.5.
    (Aliases: createhome)[Default: True]
    type: bool

- expires
    An expiry time for the user in epoch, it will be ignored on platforms that do not support this.
    Currently supported on GNU/Linux, FreeBSD, and DragonFlyBSD.

```

*Figure 5 :ACCÉDER À LA DOCUMENTATION D'UN MODULE*

### III.5-Les playbooks

Les playbooks sont l'une des principales caractéristiques d'Ansible. Ils disent à Ansible ce qu'il faut exécuter. Ils sont comme une liste de tâches que Ansible doit exécuter. Chaque tâche se connecte en interro à un morceau de code appelé module . Les playbooks sont exprimés en format YAML qui est très faciles à lire, alors que les modules sont un morceau de code qui peut être écrit dans n'importe quelle langage avec la condition que sa sortie doit être au format JSON. Nous pouvons avoir plusieurs tâches énumérées dans un playbook et ces tâches seraient exécutées en série par Ansible. Les jeux d'instructions sont exécutés séquentiellement de haut en bas. Toutefois, on peut effectuer l'exécution conditionnelle des tâches afin qu'elles puissent être ignorées si les conditions ne sont pas remplies. Autrement dit, les playbooks sont la base d'un système de déploiement de gestion de configuration de plusieurs machines. Ils sont la configuration, le déploiement et le langage d'orchestration de Ansible. Ils peuvent décrire une politique qu'on veut que les systèmes distants respectent, ou un ensemble d'étapes dans un processus informatique général.

Les playbooks de Ansible sont constitués d'une ou plusieurs jeux d'instructions. Un jeu se compose de trois sections:

La **section cible** qui définit les hôtes sur lesquels le jeu sera exécuté, et la façon dont il sera exécuté. C'est là où nous mettons le nom d'utilisateur SSH et d'autres paramètres liés à SSH.

La **section variable** qui définit les variables, qui seront mis à la disposition du jeu en exécution

La **section des tâches** qui répertorie tous les modules dans l'ordre que nous voulons qu'ils soient gérés par Ansible

La commande pour exécuter un playbook est *audible-playbook*

Exemple de playbook : *example.yml*

```
- - -  
- hosts: webservers  
user: root  
vars:  
    apache_version: 2.6  
    motd_warning: 'WARNING: Use by ACME Employees ONLY'  
    testserver: yes  
tasks:  
    - name: setup a MOTD  
      copy:  
        dest: /etc/motd  
        content: "{{ motd_warning }}"
```

Nous pouvons inclure autant de jeu d'instruction que nous voulons dans un fichier YAML unique. Les fichiers YAML commencent avec `—` et contiennent de nombreuses valeurs et listes de clés. En YAML, l'indentation des lignes est utilisée pour indiquer l'imbrication variable à l'analyseur, ce qui rend également le fichier plus facile à lire.

Explication des différentes parties du fichier :

### La section cible :

La section cible ressemble à l'extrait de code suivant

```
- hosts: webservers  
  user: root
```

Ceci est une version incroyablement simple, mais susceptible d'être tout ce qu'il faut dans la plupart des cas. Chaque jeu existe dans une liste. Selon la syntaxe YAML, la ligne doit commencer par un tiret. Les hôtes sur lesquels un jeu sera exécuté doivent être mentionnés dans la valeur des hôtes. Cette valeur utilise la même syntaxe que celui utilisé lors de la sélection des hôtes en utilisant la ligne de commande Ansible. Dans la dernière ligne, l'utilisateur indique au playbook d'Ansible, l'utilisateur auquel il doit se connecter sur la machine distante.

Les autres lignes que nous pouvons fournir dans cette section sont les suivantes:

**sudo:** on peut mettre 'yes' pour cette option si nous voulons que ansible utilise sudo pour devenir root une fois connecter aux machines dans le jeu.

**user:** Ceci définit le nom d'utilisateur pour se connecter à la machine à l'origine, avant d'exécuter sudo si configuré.

**sudo\_user:** Ceci est l'utilisateur que Ansible va essayer et devenir en utilisant sudo

**connection:** Cela nous permet de dire Ansible le protocole de transport à utiliser pour se connecter à l'hôte distant.

**gather\_facts:** Ansible exécutera automatiquement le module d'installation sur les hôtes distants à moins que nous ne le disions pas.

### **La section variable :**

Ici, nous pouvons définir des variables applicables à l'ensemble du Jeu sur toutes les machines.

Les variables dans un playbook sont définies par la clé vars. Cette clé prend une paire clé-valeur, où la clé est le nom de la variable et la valeur est la valeur réelle de la variable. Cette variable remplacera les autres variables qui sont définies par un fichier de variable global ou à partir d'un fichier d'inventaire.

Nous pouvons faire de sorte qu'Ansible demande les variables si elles n'ont pas été saisies dans la ligne de commande. Ceci permet de rendre plus facilement maintenable les Jeux d'instructions et nous évite modifier les mêmes choses dans

plusieurs parties de la pièce. Cela nous permet également d'avoir tout le jeu stocké en haut, où nous pouvons facilement le lire et le modifier sans se soucier de ce que le reste du Jeu fait. Les variables dans cette section d'un jeu peuvent être remplacées par des actions de la machine (ceux qui sont définis par modules), mais ils remplacent eux-mêmes les faits que nous avons définis dans notre inventaire. Donc, ils sont utiles pour définir les défauts que nous pourrions collecter dans un module plus tard, mais ils ne peuvent pas être utilisés pour maintenir les valeurs par défaut pour les variables d'inventaire, car elles remplaceront ces valeurs par défaut.

Les déclarations de variables, qui se produisent dans la section vars, ressemblent aux valeurs dans la cible et contiennent un dictionnaire YAML ou une liste. Un exemple ressemble au fragment de code suivant:

```
vars:
  apache version: 2.6
  motd warning: ' WARNING: Use by ACNE Employees ONLY'
  testserver: yes
```



Les variables peuvent également être chargées à partir de fichiers YAML externes en donnant à Ansible une liste de variable fichiers à charger. Cela se fait d'une manière similaire en utilisant la directive `vars_files`. Ensuite nous pouvons tout simplement fournir le nom d'un autre fichier YAML qui contient son propre dictionnaire. Ceci veut dire qu'au lieu de stocker les variables dans le même fichier, elles peuvent être stockées et distribuées séparément, ce qui nous permet de partager notre playbook avec d'autres.

En utilisant `vars_files`, les fichiers ressemblent à l'extrait de code suivant dans notre manuel:

```
vars_files:
    conf/country-AU.yml
    conf/datacenter-SYD.yml
    conf/cluster-mysql.yml
```

Dans l'exemple précédent, Ansible recherche le *pays-UA.yml*, *dotacenter-SYD.yml* et *Cluster-mysql.yml* dans le dossier *conf* par rapport au chemin du playbook. Chaque fichier YAML ressemble à l'extrait de code suivant

```
---

ntp: ntp1. au. example. com
TZ: Australia/Sydney
```

## La section des tâches

La section de tâches est la dernière section de chaque jeu. Il contient une liste d'actions que nous voulons qu'Ansible effectue dans l'ordre que nous voulons qu'ils soient effectués. Il existe plusieurs styles dans lesquels nous pouvons exprimer les arguments de chaque module. L'extrait de code suivant est une section de tâches ressemblant à trois styles montrés:

```
-tasks:
  -name: install apache
    yum: name=httpd state=installed
  - name: configure apache
    copy: src=files/httpd.conf dest=/etc/httpd/conf/httpd.conf
  - name: restart apache
    service:
      name: httpd
      state: absent
```

Mous voyons ici les trois différents styles de syntaxe utilisés pour installer, configurer et démarrer le serveur Web Apache comme il sera le cas sur une machine CentOS. La première tâche nous montre comment installer Apache en utilisant la syntaxe originale, ce qui nous oblige à appeler le module comme le premier dans une clé d'action. La deuxième tâche copie le fichier de configuration d'Apache en utilisant le deuxième style de la tâche. Dans ce style, on utilise le nom du module à l'action et sa valeur devient simplement son argument. Enfin, la dernière tâche, le troisième style, montre comment utiliser le module de service pour redémarrer Apache. Dans ce style, nous utilisons le module comme la clé, mais nous fournissons les arguments en tant que dictionnaire Y AML.

Cela peut être pratique lorsque nous fournissons un grand nombre d'arguments à un seul module, ou si le module veut les arguments sous une forme complexe. Ce dernier style devient rapidement la manière préférée d'écrire car un nombre croissant de modules nécessitent des arguments complexes.

Notons que les noms ne sont pas requis pour les tâches. Cependant, ils font de la documentation et nous nous référons à chaque tâche plus tard, si nécessaire.

Les noms sont également envoyés à la console lorsque le playbook est exécuté, de sorte que l'utilisateur peut dire ce qui se passe.

## La section des gestionnaires

La section des gestionnaires est syntaxiquement la même que la section de tâche et prend en charge le même format pour appeler les modules. Les gestionnaires ne sont appelés que lorsque la tâche à laquelle ils ont été appelés, enregistre que quelque chose a changé pendant l'exécution. Pour déclencher un gestionnaire, nous ajoutons une clé de notification à la tâche avec la valeur définie au nom de la tâche.

Les gestionnaires sont exécutés s'ils ont été déclenchés auparavant lorsque Ansible a terminé l'exécution de la liste des tâches.

Ils sont exécutés dans l'ordre où ils sont répertoriés dans la section des gestionnaires et même s'ils sont appelés plusieurs fois dans la section des tâches, elles ne seront exécutées qu'une seule fois. Ceci est souvent utilisé pour redémarrer les démons après leur mise à niveau et leur configuration. Le jeu suivant démontre comment nous allons mettre à niveau un DHCP ISC (Dynamic Host Configuration Protocol) à la dernière version, le configurer et le configurer pour démarrer au démarrage. Si ce Playbook est exécuté sur un serveur où le démon DHCP ISC exécute déjà les versions et les fichiers de configuration ne sont pas modifiés, le gestionnaire ne sera pas appelé et DHCP ne sera pas redémarré. Considérons le code suivant par exemple:

```
---

- hosts: dhcp
  tasks:
    - name: update to latest DHCP
      yum :
        name: dhcp
        state: latest
        notify: restart dhcp
    - name: copy the DHCP config
      copy :
        src: dhcp/dhcpd.conf
        dest: /etc/dhcp/dhcpd.conf
        notify: restart dhcp
    - name: start DHCP at boot
      service:
        name: dhcpd
        state: started
        enabled: yes
  handlers:
    - name: restart dhcp
      service :
        name: dhcpd
        state: restarted
```

## IV-Réalisation

### IV.1-Installation d'Ansible

Actuellement Ansible peut être exécuté à partir de toute machine avec Python 2.6 ou 2.7 installé. Les Windows ne sont pas pris en charge pour la machine de contrôle.

#### Ubuntu / Debian

```
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository ansible / ansible
$ sudo apt-get update
$ sudo apt-get install ansible
```

#### Fedora / RHEL / CentOS

```
$ yum install ansible
```

#### Installation à partir de source

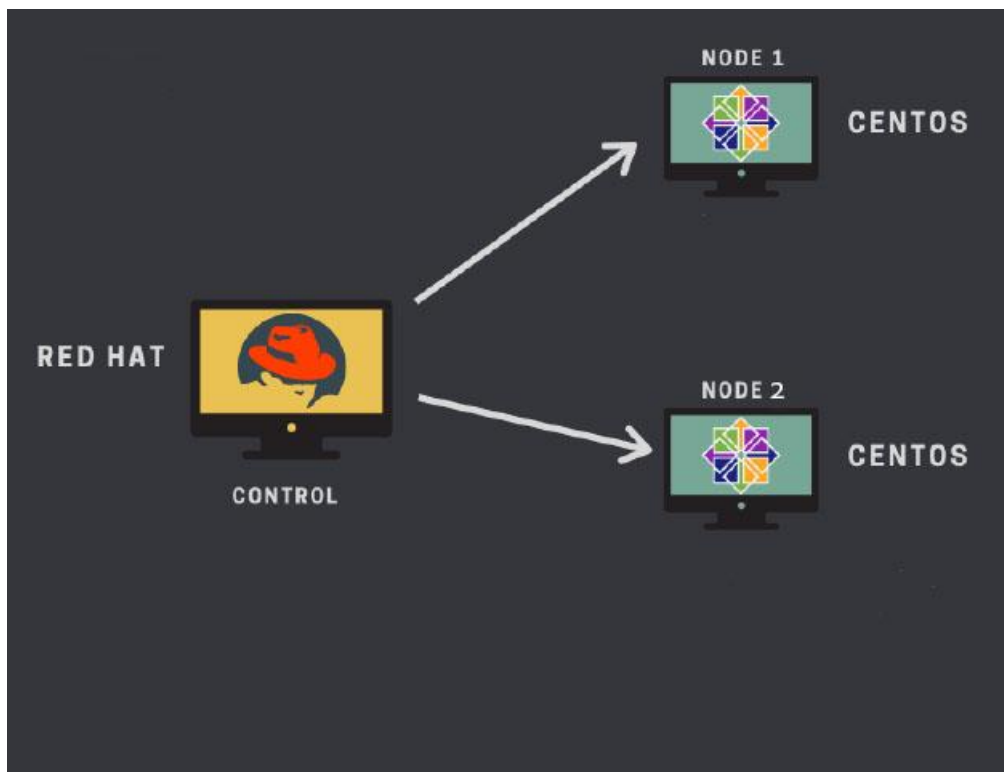
```
$ git clone git://github.com/ansible/ansible.git
$ cd ansible
$ sudo make install
```

Une fois qu'Ansible est installé, pour s'assurer qu'il fonctionne correctement en entrant `ansible --version` dans la ligne de commande on peut retrouver la version.

```
$ ansible --version
```

## IV.2 -Architecture

Pour cette architecture nous allons utiliser trois machines dont nous allons installer Ansible sur une qui sera le contrôleur et les autres seront contrôlées à distance.



*Figure 6 :Architecture*

Dans cette architecture nous disposons de deux serveurs web et un serveur d'application qui sont gérés à distance par la machine de contrôle.

## IV.3 -Déploiement

### Installation et configuration d'un serveur apache

Pour notre démonstration on va chercher à installer serveur apache sur nos serveurs

#### Configurer Apache Utilisant Ansible sur CentOS

Apache est l'un des serveurs Web les plus populaires actuellement utilisés sur Internet. Il est facile à installer et à configurer sur les distributions Linux comme Ubuntu et Debian comme il vient dans les dépôts de paquets et comprend une configuration par défaut qui fonctionne hors de la boîte

### Etape1 : Configuration Ansible

Dans cette section , nous allons configurer Ansible pour être en mesure de gérer notre serveur .

Créons un nouveau répertoire que nous allons utiliser pour cette configuration :

\$ mkdir Project

Créons un nouveau fichier appelé ansible.cfg et éditons-le :

```
[defaults]
inventory = inventory
remote_user = ansible
host_key_checking = false
INJECT_FACTS_AS_VARS = False
roles_path= /home/ansible/project1/roles
[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ack_pass = false
```

*Figure 7: Configuration Ansible*

Ensuite, le fichier hosts doit être édité. Il y a beaucoup d'options disponibles pour le fichier hosts .Cependant, nous pouvons commencer avec quelque chose de très simple.

Créons un fichier hosts nommé inventory et éditons-le .

```
[ansible@Control project1]$ nano inventory
```

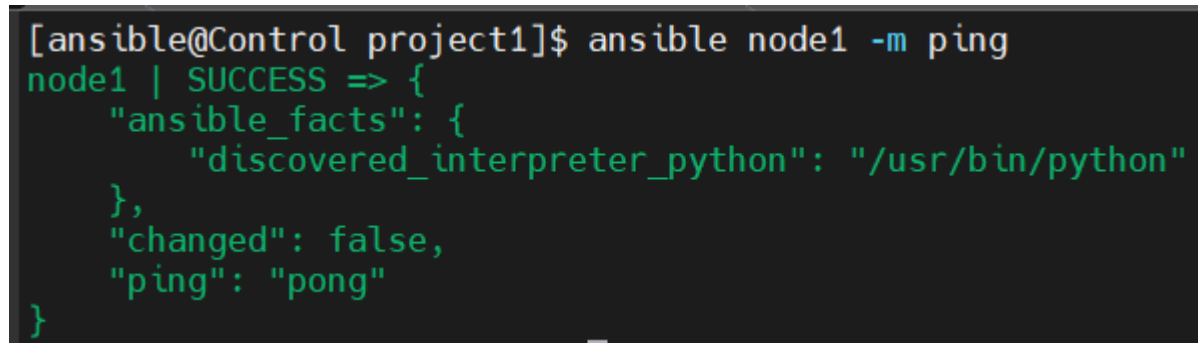
```
[web]
node1
#node1.ansible.com
#node1 ansible_host=node1.ansible.com group=yellow user=esprit
#node1.ansible.com ansible_host=node1

[db]
node2 group=IRS02 user=Mater-Linux

[datacenter:children]
web
db
~
```

*Figure 8 :Fichier inventaire d'Ansible (Inventory File)*

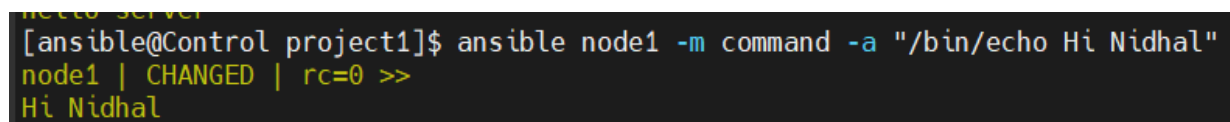
Pour vérifier que ansible fonctionne et peut communiquer aux hôtes (node1,node2), nous pouvons exécuter une commande de base ansible. Utilisation de la commande de base ansible prend le groupe hôte et le nom du module : `ansible <group> -m <module>`. Pour exécuter la commande ping, entrons la commande suivant :



```
[ansible@Control project1]$ ansible node1 -m ping
node1 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/bin/python"
  },
  "changed": false,
  "ping": "pong"
}
```

*Figure 9 : Tester la connectivité entre les Machines*

Un autre module Ansible qui est utile pour le Test est le module **command** .Il exécute des commandes personnalisées sur l'hôte et renvoie les résultats. Pour s'exécuter la commande command utilise echo ,une commande Unix qui fait écho à une chaîne au terminale , entrons la commande suivante.



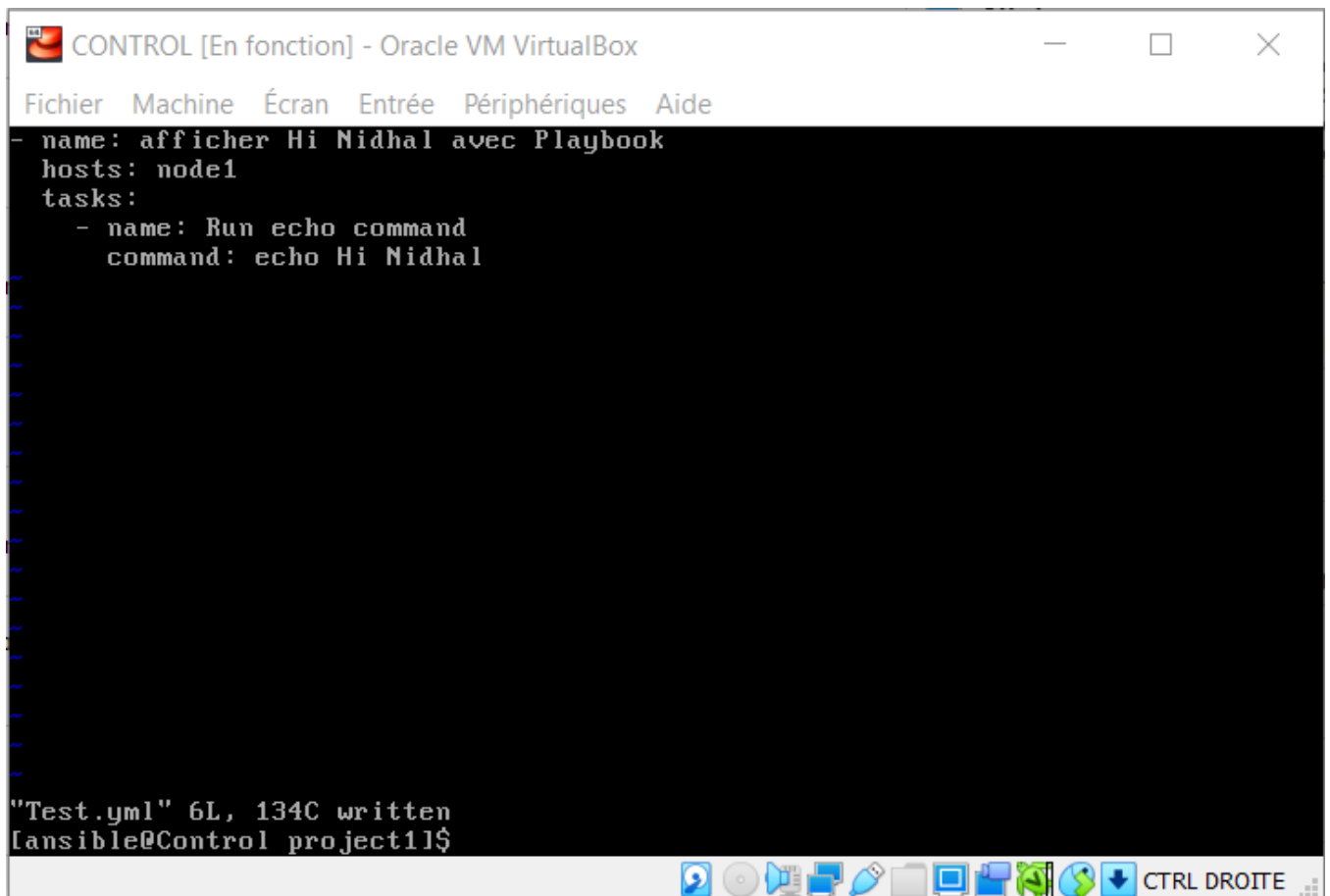
```
hello server
[ansible@Control project1]$ ansible node1 -m command -a "/bin/echo Hi Nidhal"
node1 | CHANGED | rc=0 >>
Hi Nidhal
```

### Etape2 : Création d'un Playbook :

Dans cette section, nous allons créer un playbook Ansible de base pour nous permettre d'exécuter facilement des modules plus complexes.

Nous allons créer une version basique de playbook de la commande **hello server** ci-dessous

Créer un fichier appelé Test1.yml et éditons le.

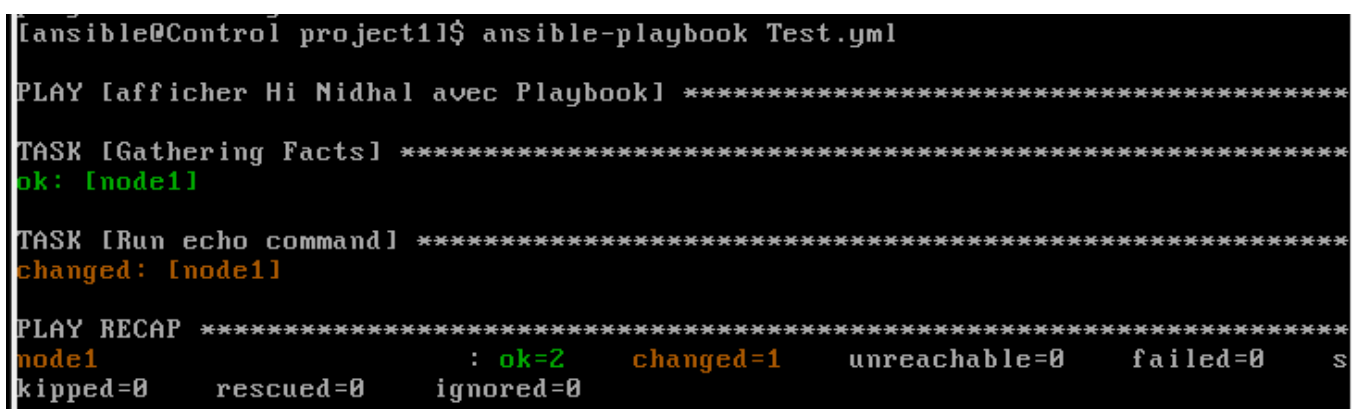


```
CONTROL [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide
- name: afficher Hi Nidhal avec Playbook
  hosts: node1
  tasks:
    - name: Run echo command
      command: echo Hi Nidhal

"Test.yml" 6L, 134C written
[ansible@Control project1]$
```

*Figure 10 : Test.yml*

La commande `ansible-playbook Test.yml` est utilisée pour exécuter les playbooks, et l'utilisation la plus simple est la suivante : Nous pouvons exécuter le playbook que nous venons de créer avec la commande suivante :



```
[ansible@Control project1]$ ansible-playbook Test.yml
PLAY [afficher Hi Nidhal avec Playbook] *****
TASK [Gathering Facts] *****
ok: [node1]
TASK [Run echo command] *****
changed: [node1]
PLAY RECAP *****
node1 : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

*Figure 11 : Exécution de playbook*

La chose la plus importante à noter ici est que playbooks ne revient pas la sortie du module, donc contrairement à la commande directe, que nous avons utilisé à



l'étape1, on ne peut pas voir si hello server a été effectivement imprimé. Cela signifie que les playbooks sont mieux adaptés pour les taches ou vous n'avez pas besoin de voir la sortie. Ansible nous dira s'il y avait une erreur lors de l'exécution d'un module, de sorte que nous avons seulement besoin de cela pour savoir si quelque chose va mal

### Etape 3 :Installation d'Apache :

Maintenant que nous avons les introduits playbooks, nous allons écrire les taches pour installer le serveur web Apache

Normalement sur CentOS, L'installation d'Apache est un cas simple d'installer le package httpd via Yum . Pour ce faire via Ansible, nous utilisons le module yum d'Ansible. Le module yum contient un certain nombre d'options spécialisés pour la fonctionnalité yum. Les options qui nous intéressent sont les suivants :

**name** : Le nom du package à installer, soit un nom de paquet unique ou une liste de paquets.

**state** : Accepte soit latest, absent ou present

latest assure l'installation de la dernière version

present verifie simplement qu'il est installé

absent il supprime si elle est installée.

Maintenant, nous allons éditer notre Playbook Installer\_httpd.yml avec le module yum

```
--
- name: Install httpd
  hosts: web
  tasks:
    - name: Install Required Packages
      yum:
        name: httpd
        state: present
    - name: Enable httpd service
      service:
        name: httpd
        enabled: yes
        state: started
~
~
```

*Figure 12 : Installer\_httpd.yml*

pour vérifier le syntaxe du playbook :

```
[ansible@Control project1]$ ansible-playbook install_httpd.yml --syntax-check
playbook: install_httpd.yml
```

*Figure 13 : Vérifier le syntaxe du playbook*

Maintenant, exécutant le playbook :

```
[ansible@Control project1]$ ansible-playbook install_httpd.yml

PLAY [Install httpd] *****

TASK [Gathering Facts] *****
ok: [node1]

TASK [Install Required Packages] *****
ok: [node1]

TASK [Enable httpd service] *****
changed: [node1]

PLAY RECAP *****
node1                : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

*Figure 14 : Exécution de Playbook*

pour plus de détails ,on va vérifier dans la machine node1

Voici avant l'exécution du playbook

```
[root@node1 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
     Docs: man:httpd(8)
           man:apachectl(8)
```

Voici après l'exécution du playbook

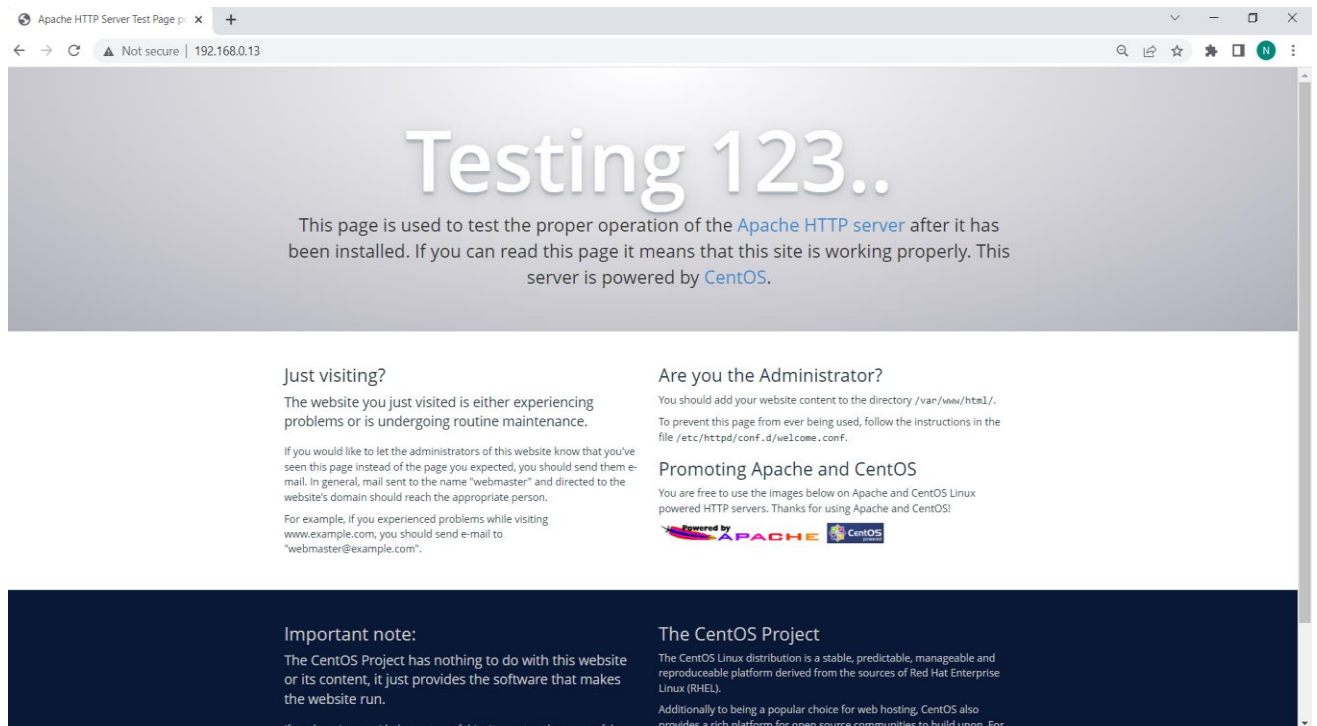
```
[root@node1 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2022-10-07 19:23:27 CET; 1min 46s ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Main PID: 2236 (httpd)
    Status: "Total requests: 0; Current requests/sec: 0; Current traffic:  0 B/sec"
   CGroup: /system.slice/httpd.service
           └─2236 /usr/sbin/httpd -DFOREGROUND
             └─2237 /usr/sbin/httpd -DFOREGROUND
               └─2238 /usr/sbin/httpd -DFOREGROUND
                 └─2239 /usr/sbin/httpd -DFOREGROUND
                   └─2241 /usr/sbin/httpd -DFOREGROUND
                     └─2242 /usr/sbin/httpd -DFOREGROUND

Oct 07 19:23:26 node1.ansible.com systemd[1]: Starting The Apache HTTP Server...
Oct 07 19:23:27 node1.ansible.com systemd[1]: Started The Apache HTTP Server.
[root@node1 ~]#
```

*Figure 15 : Vérifier l'état de la serveur Apache*

Cela nous indique que le serveur httpd a déjà installé

Si nous visitons le nom d'hôte de node1 ou l'adresse IP dans votre navigateur, nous devrions maintenant obtenir une page par défaut Apache http server CentOS pour nous saluer .



*Figure 16 : Page par défaut Apache http server CentOS*

2 autres playbooks supplémentaire :

**Le premier pour supprimer le serveur http :**

```

- name: Delete httpd from all hosts
  hosts: all
  tasks:
    - name: Delete httpd
      yum:
        name: httpd
        state: absent

```

*Figure 17 : Supprimer le serveur http*

**Le second pour Créer utilisateur et Group :**

```

- name: Create User and Group
  hosts: all
  vars:
    username: Midhal
    groupname: ISI1
  tasks:
    - name: Create Group
      group:
        name: "{{ groupname }}"
        state: present
    - name: Create User
      user:
        name: "{{ username }}"
        group: "{{ groupname }}"
        password: "{{ 'ohmyGod' | password_hash('sha512', 'mysecretsalt') }}"
        state: present

```

*Figure 18 : Créer Utilisateur et Group*

## Conclusion

En somme, ansible est un outil puissant permettant la gestion de plusieurs machins à distance. A travers ce Rapport nous avons montré quelques tâches qu'on peut effectuer avec ansible. Ce dernier nous a permis de déployer des applications, de démarrer les services, de configurer et de déployer les serveurs web à distance.