

Mise en place d'une infrastructure Cloud avec Ansible



AGENCE NATIONALE DE REGLEMENTATION DES TELECOMMUNICATIONS
INSTITUT NATIONAL DES POSTES ET TELECOMMUNICATIONS

Préparé par : Kaoutar Boussaoud

Encadré par : Oussama El hafyani

Remerciement

D'abord, je tiens à remercier l'équipe de 4D, et spécialement l'équipe wakanda qui m'a acceptée comme stagiaire au sein de leur entreprise.

Mes vifs remerciements sont adressés plus particulièrement à Monsieur Oussama El Hafyani mon encadrant du stage, pour son encadrement estimable et son accueil chaleureux et son humanisme, et je tiens à le remercier de nouveau car il était toujours à l'écoute de mes questions.

Je n'oublie pas l'amabilité et la gentillesse de tous les membres de l'équipe wakanda qui m'ont prêté la main forte et m'aider à résoudre les problèmes rencontrés, à savoir : Mr. Zakaria Anbari, Abdelouahab El OUAZZAM.

En fin, ma gratitude mes formatrices et formateurs, pour leur compréhension, disponibilité, tolérance, gentillesse et leur soutien durant mes études à l'MPT ainsi que ma famille qui m'a soutenu tout au long de mes études.

Table de matière

Présentation de l'entreprise d'accueil.....	3
Automatisation du cloud avec ansible.....	4
I. Concept de cloud computing.....	4
II. Le concept d'Ansible	8
III. Le concept de vagrant.....	10
IV. Ansible et vagrant.	13
V. les commandes Ad-hoc :	17
VI. Les modules d'ansibles.	24
VIII. Les rôles.	32
IX. Include statement.....	39
Automatisation d'une architecture avec Nginx et Wakanda.....	41
I. C'est quoi wakanda ?.....	41
II. Présentation de l'application créée sur Windows.....	41
III. C'est quoi alors Nginx ?	46
Conclusion.....	47

Présentation de l'entreprise d'accueil

4D a été créé en 2008 pour servir le GCC, Moyen-Orient et Afrique du Nord comme conseiller régional en gestion, en services technologiques et en externalisation de l'entreprise, avec des consultants experts au service des clients. 4D est située au Maroc, en Tunisie, en Égypte et en Jordanie dans la région MENA

4D LOGICIELS MAROC se trouve à Rabat, Maroc. 4D permet de créer des solutions d'entreprise modernes, puissantes et ouvertes sur le monde. Elle aide à développer des systèmes compatibles et compétitifs, prêts pour toutes les évolutions du futur.

Elle propose aux développeurs et aux utilisateurs finaux une plateforme logicielle intégrée et puissante, incluant les standards reconnus ou naissants, tout en maintenant la compatibilité avec les systèmes existants.

4D met en œuvre cette stratégie au service de PME, de grandes entreprises, d'universités prestigieuses, d'organismes gouvernementaux, de développeurs indépendants, de vendeurs de solutions verticales, depuis l'avènement de la micro-informatique.

4D aide les organisations à confronter les changements avec succès, tout en développant et en opérant évolutivement, des processus d'affaire efficaces et agiles permettant de créer à la fois des avantages compétitifs immédiats et à long terme. Si de nombreuses entreprises fournissent une assistance à des éléments spécifiques de BPM (Business Process Management), la combinaison unique de compétences à 4D, ainsi que les démarches et les outils utilisés génèrent rapidement de la valeur, fournissent des résultats mesurables, et créent des solutions durables de BPM.

Automatisation du cloud avec ansible

I. Concept de cloud computing.

➤ Cloud computing.

Cloud computing est un modèle permettant d'offrir un accès simple, en tout lieu à la demande, à un ensemble des ressources informatiques configurables et partagées : réseaux, serveurs, stockage, applications et services. On paie pour accéder à ces services qui peuvent être :

- ✓ Software
- ✓ Application
- ✓ infrastructure

Il existe 3 modèles de services qui sont : iaas, paas, et saas, il existe 3 modèles de déploiement tels que : cloud privé, cloud communautaire, cloud public et cloud hybride

➤ Cloud privé

L'infrastructure du Cloud est réservée à l'usage exclusif d'une seule organisation. Elle peut être possédée, gérée et opérée par cette organisation, un intervenant extérieur ou une combinaison des deux. Elle est située dans les locaux de l'organisation ou dans ceux d'un hébergeur externe.

➤ Cloud public

L'infrastructure du Cloud est destinée à un usage public. Elle peut être possédée, gérée et opérée par un organisme privé, public, académique ou une combinaison de ceux-ci. Elle est située chez un hébergeur.

➤ Cloud communautaire

L'infrastructure du Cloud est réservée à l'usage d'une communauté spécifique de consommateurs partageant des intérêts communs : missions, exigences de sécurité, partage d'informations et ou d'applications,... Elle peut être possédée, gérée et opérée par un ou plusieurs organismes participant à la communauté, un intervenant extérieur ou une combinaison d'entre eux. Elle est située dans les locaux de l'organisation ou dans ceux d'un hébergeur externe.

➤ Cloud hybride

L'infrastructure du Cloud est composée d'au moins deux infrastructures différentes (privée, publique ou communautaire) qui conservent leur autonomie mais qui sont liées entre elles par des technologies (propriétaires ou non) assurant la portabilité des données et des applications

➤ SAAS (Software as a Service)

Définition :

Software as a Service est le modèle Cloud le plus simple pour utiliser un ou plusieurs logiciels. Les entreprises accèdent à leurs applications depuis tout poste connecté, depuis un simple navigateur web. Les clients paient un accès à leurs logiciels sous forme d'abonnement.

Autrement dit, il s'agit d'une utilisation d'un logiciel, comme d'un service.

Vous n'installez pas le logiciel que vous voulez utiliser sur votre ordinateur, mais vous l'utilisez à distance (le logiciel « tourne » sur des serveurs dans des datacenters). La plupart du temps, ces logiciels s'utilisent via un navigateur Web. Les clients ne paient pas pour posséder le logiciel en lui-même mais plutôt pour l'utiliser.

✓ **Avant le cloud :**

- il fallait acheter un logiciel,
- installer le logiciel sur votre ordinateur.
- Refaire l'opération en cas de changement d'ordinateur.
- Ne pas oublier de copier les données en cas de changement d'ordinateur.
- Penser à faire des sauvegardes pour ne pas perdre les données en cas de panne de disque dur.

✓ **Avec le cloud :**

- vous choisissez le logiciel en ligne qui vous convient,
- vous payez ou pas le service et vous utilisez le logiciel.

Avantage :

1. Pas d'installation de logiciels, vous accédez à l'application depuis votre navigateur internet, autrement dit accessible via n'importe quel ordinateur.
2. Vous n'achetez plus une licence d'utilisation mais un accès à votre application.
3. Vos logiciels bénéficient de nombreuses mises à jour de l'éditeur, sans maintenance particulière de votre part.

Ce type de cloud dédiée aux : internautes.

➤ IAAS (infrastructure as a service).

Utilisation d'une infrastructure, comme d'un service.

Vous n'avez pas besoin d'acheter un ensemble de matériel pour installer votre infrastructure. Vous vous contentez de louer ce matériel comme s'il vous appartenait. Vous installez les serveurs que vous souhaitez utiliser. Vous gérez l'ensemble des OS installés sur les serveurs que vous louez.

- ✓ Les entreprises utilisent beaucoup le IaaS car elles peuvent ainsi disposer de serveurs disponibles, de dernières génération et très rapidement disponibles.

- ✓ L'utilisateur crée des machines virtuelles via des machines images existantes. et de choisir des images, créer des machines virtuelles pour DBMS et configure BDMS.
- ✓ Choisir image + créer VM, configurer VM pour l'application.

Mon entreprise doit installer 1 serveur. Voici les étapes, dans les 2 cas :

Etapes	Mode classique (sans le IaaS)	Mode IaaS
Avant	Chercher une offre ou appeler un fournisseur pour acheter un serveur.	Se connecter au site d'un hébergeur.
	Commander le serveur	Choisir le serveur.
	Attendre sa livraison, (1 jour, 2 jour, ou plus ?)	Attendre sa livraison (quelques minutes)
	Trouver une place pour le mettre (prises de courant, et prises réseau)	
	Réceptionner la livraison	
	Déballer le serveur du carton et vérifier le contenu de la livraison	
	Le brancher Electriquement et le brancher au réseau	
	Le démarrer et enfin commencer à travailler dessus	
Mon serveur est enfin opérationnel	Au final : j'ai passé plusieurs heures au minimum, et j'ai dû patienter plusieurs jours pour la livraison de mon serveur	Quelques minutes (ou quelques dizaine de minutes).
Pendant sa durée d'utilisation	il faut quand même que je vérifie si le matériel et notamment les disques ne tombent pas en panne	rien à faire, l'hébergeur s'en charge.
	Que je m'assure qu'il est toujours alimenté en électricité	rien à faire, l'hébergeur s'en charge.
Après	Il faudra que je gère son recyclage !	rien à faire, l'hébergeur s'en charge.

Ce type de cloud dédie aux administrateurs.

➤ PAAS (Platform as as Service)

Utilisation d'une plateforme, comme d'un service.

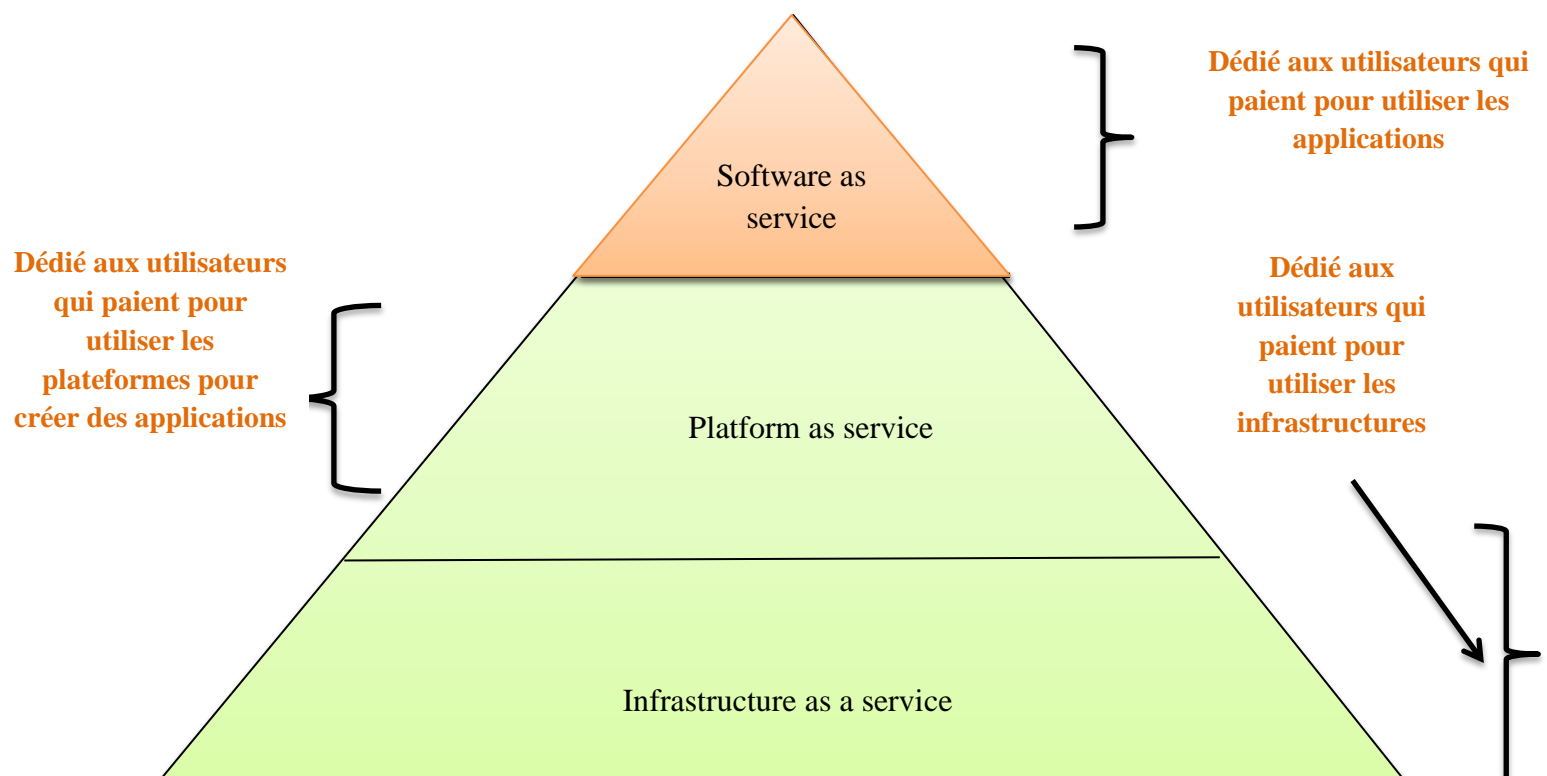
Vous louez une plateforme, c'est à dire une machine avec un OS, le tout prêt à l'emploi. L'utilisateur crée des applications, il ne crée pas des machines virtuelles, il crée des bases de données, créer des tableaux, ajouter des données et déployer les applications, il permet la création et le développement des applications avec prix bas.

Voici quelques exemples de Paas :

- Louer un serveur, avec le logiciel Exchange déjà installé dessus. (Pour ceux qui ne savent pas ce que c'est : Exchange est un logiciel permettant de gérer la messagerie des entreprises. Cette location est bien en mode PaaS, car dans ce cas, vous louez : Le serveur, l'OS, et le logiciel et le tout déjà préinstallé.
- Louer un serveur avec les systèmes de virtualisation déjà préinstallés.

Ce type de cloud dédie aux : développeurs

Remarque : On pourrait aussi parler du DaaS, (Desktop as a Service). Il s'agit d'une nouvelle définition car elle est plus récente que les autres. Le DaaS permet de mettre à disposition un bureau. Dans ce bureau, on va retrouver des logiciels installés et des espaces de stockage. L'avantage du DaaS est que vous pouvez vous connecter à ce bureau avec un ordinateur, un smartphone, une tablette, une télévision, etc.. et que vous retrouvez sur l'écran un bureau qui permet de faire la même chose que si vous étiez devant un ordinateur.



II. Le concept d'Ansible.

✚ Ansible est outil (ou bien logiciel) qui permet la configuration des systèmes (serveurs), le déploiement des logiciels (au niveau des serveurs), l'orchestration des tâches développées (comme : continuous deployments or zero downtime rolling updates) à distance, autrement dit ansible permet les fonctionnalités suivantes :

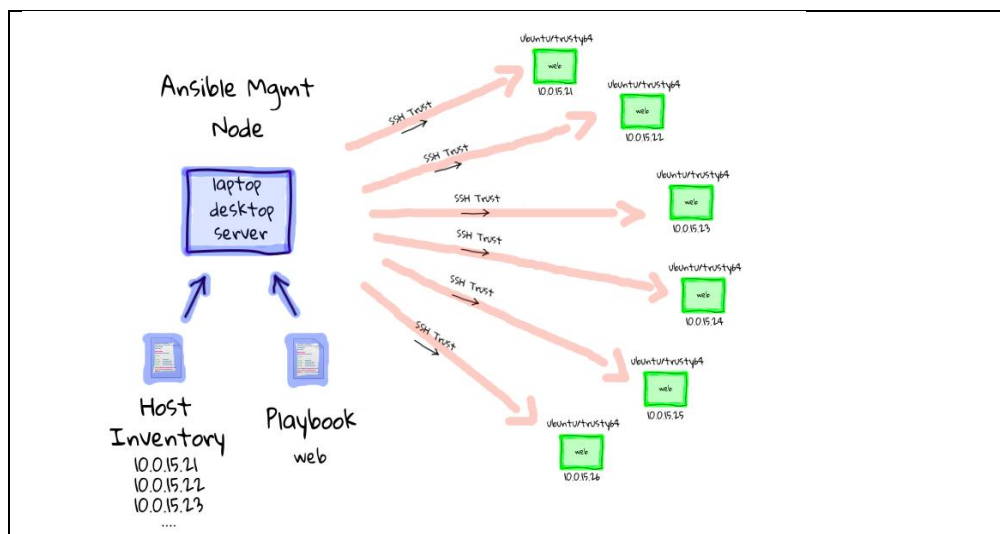
- ✓ d'installer des paquets
- ✓ d'installer des fichiers de configuration
- ✓ de configurer les utilisateurs systèmes
- ✓ de configurer les services
- ✓ de réaliser n'importe quelle tâche d'administration que l'on pourrait réaliser par ssh.

Il existe autre outils de gestion de configuration tels que Chef et Puppet, Mais ansible n'est pas similaire à ces outils de gestion de configuration (Chef et Puppet). Ansible est écrit en langage Python et il contient fichiers qui sont :

- ✓ inventory (hosts)
- ✓ roles
- ✓ playbook

mais pour effectuer la configuration ansible a besoin juste de deux fichiers de configuration le premier "host inventory" et le deuxième "playbook". A savoir que le fichier "host inventory" contient une liste des noms des machines ou bien des @IP des machines que l'on veut gérer, pour le fichier "playbook" contient une tâche ou bien liste des tâches à effectuer au niveau d'un ou des plusieurs serveurs. Ces tâches sont de la forme des commandes qui représente la configuration ou bien déploiement à effectuer au niveau du serveur ciblé. le fichier de configuration "playbook" est écrit en langage YAML.

✚ Ansible se connecte en parallèle et simultanément à tous les serveurs qu'on veut gérer.



Les avantages d'ansible:

- ✚ Ansible utilise des simples syntaxes YAML, qui est facile et compréhensible par tous (les développeurs, les administrateurs systèmes et les managers).
- ✚ Il est rapide à apprendre, et l'installation du ansible n'est pas complexe, il est facile et rapide.
- ✚ on peut gagner du temps si on a configuré plusieurs serveurs en même temps et par une simple commande sans avoir besoin d'accéder à chaque serveur et le configurer.
- ✚ sécurisé: ansible utilise SSH, alors il ne demande pas un port supplémentaire d'où la protection du serveur.
- ✚ Ce qui rend ansible plus avantageux c'est le langage facile utilisé comparablement à Puppet et chef ils demandent connaître le langage Ruby et/ ou autre langage juste pour commencer.
- ✚ Ansible ne demande pas un logiciel supplémentaire à ajouter comparablement aux autres outils de gestion de configuration il est efficace.
- ✚ Une fois ansible effectue les commandes pour configurer les machines, il arrête de s'intéresser à ces commandes encore fois si tu ne lui as demandé contrairement aux autres outils de gestion de configuration qui a un agent installer au niveau de chaque machine qui vérifie ces configurations effectuées.
- ✚ Ansible est caractérisé par Idempotence c'est à dire la capacité d'obtenir les mêmes résultats si on a fait une opération plus qu'une fois. si on a, par exemple, effectué une configuration au niveau d'un serveur et on a fait cette opération plus qu'une fois aucun changement ne sera fait au niveau du serveur après la première configuration.

Comparaison entre ansible, chef et puppet.

- ✚ Puppet et chef sont écrits en Ruby, ansible écrit en python et exige que le serveur contient des bibliothèques de python, les librairies de python sont par défaut installées au niveau du linux sinon il faut les installer.
- ✚ On a besoin de savoir Ruby car ansible utilise le langage YAML qui est un langage facile.
- ✚ Ansible est disponible seulement en linux et UNIX, et ne pas sur Windows ou MAC os.
- ✚ l'interface graphique (GUI) n'est pas encore développée.
- ✚ chef est disponible sur Linus, Unix, Windows.
- ✚ chef très difficile à maîtriser et à déployer.

- ✚ chef et puppet exige que un agent doit être installé au niveau la machine qu'on veut gérer, qui sert pour le control qui n'est le cas du ansible.
- ✚ une fois ansible est installé le master server peut communiquer avec les serveurs à gérer via SSH.
- ✚ Ansible utilise paramiko, qui est un standard ssh pour la communication, avec un mode accéléré qui permet une rapide et large communication.
- ✚ La configuration est écrite en YAML dans un fichier de configuration dite playbook.
- ✚ Ansible a une collection des modules qui peuvent être utilisé pour gérer différent système comme Amazon EC2 et Openstack.
- ✚ Ansible a une meilleure sécurité en utilisant ssh/ssh2.

NB : les serveurs qu'on peut trouver au niveau des machines clients à gérer: un Bind9, un Postfix (serveur de messagerie), un Nginx (serveur web ou http), un Apache(serveur http), un MySQL(serveur base de données), un PostgreSQL...

III. Le concept de vagrant.

➤ Introduction du vagrant.

qu'est-ce qu'une machine virtuelle ?

Le principe d'une machine virtuelle est de recréer un environnement indépendant, qu'il soit logiciel ou matériel, en utilisant les ressources d'un environnement hôte.

on peut par exemple demander à ma machine avec 2 processeurs et 4 go de RAM d'en émuler une avec 1 processeur et 1GO de RAM.

Dans les faits il existe 3 niveaux d'émulation:

- ✚ L'émulation complète, d'un ordinateur complet, dans ce cas on peut émuler une architecture différente du host. Par exemple on peut émuler un processeur ARM (qui est par exemple utilisé sur des consoles de jeu) alors que on a un processeur Intel. C'est très intéressant et ça peut être faite avec QEMU, par contre c'est lent.
- ✚ Ensuite on a la virtualisation, c'est elle qui nous intéresse. Il s'agit d'émuler un ordinateur du même type que l'hôte.
L'intérêt est par exemple de virtualiser un environnement linux alors qu'on utilise un environnement OSX.
C'est rapide et c'est sécurisé car l'hôte et le système virtuel sont distincts. Ça peut être fait avec Virtualbox par exemple.
- ✚ Enfin on a les containers qui consistent à émuler un ou plusieurs environnements sur un même hôte.
Ce sont des environnements applicatifs plutôt que des systèmes complets. Cela pourrait par exemple être utile pour faire tourner une application ruby on rails qui

accède à une base de données. Les environnements sont cloisonnés entre eux mais on est dépendant de l'hôte. Par exemple on ne peut pas faire tourner un programme windows si l'hôte est un linux. C'est par contre encore plus rapide qu'une virtualisation. À noter quand même qu'il n'existe pas de container sous Windows. Cette solution peut être mise en place avec docker.

➤ Vagrant.

- ✚ Vagrant est un outil permettant de générer des machines virtuelles, il permet notamment de faciliter la création des machines virtuelles que l'on peut utiliser dans notre environnement de développement. Autrement dit vagrant permet d'automatiser la création et la gestion de mes machines virtuelles. Il écrit en Ruby

➤ Les avantages de vagrant :

- ✚ Création et la configuration rapide des environnements de développements.
- ✚ Automatiser l'installation des machines virtuelles.
- ✚ Ne demande pas que l'utilisateur sache comment se fait l'installation des machines virtuelles.
- ✚ Les principaux avantages de cet outil sont la simplicité, la portabilité des VM, la légèreté (comparé à une machine virtuelle installée avec l'un des outils cités précédemment) et également la facilité que l'on a à reproduire ou dupliquer un même environnement plusieurs fois.
- ✚ Vagrant permet de configurer et contrôler plusieurs VMs avec un seul Vagrantfile.
- ✚ Lorsqu'on exécute `vagrant up` la première fois, vagrant automatiquement provisionne la nouvelle machine virtuelle créée utilisant l'outil de provisionnement configuré au niveau du fichier `vagrantfile`. On peut exécuter la commande `vagrant provision` après qu'on a créé la machine virtuelle pour expliciter l'exécution encore une fois l'outil de provisionnement dans notre cas l'outil de provisionnement utilisé est Ansible.

➤ Essai de création des machines virtuelles par vagrant.

Pour ce fait la procédure est :

- ✚ D'installer vagrant et virtualbox au niveau de la machine et on essaie de créer les machines virtuelles grâce à vagrant, en utilisant les commandes suivantes :
Dans un premier lieu la commande à exécuter est :
`vagrant init ubuntu/precise32` : cette commande va créer le fichier `vagrantfile`, et va spécifier le chemin vers la machines virtuelles à importer.
Une fois le fichier `vagrantfile` est créé on exécute la commande : `vagrant up` qui sert à importer et configurer la machine virtuelle. Après la création de la machine virtuelle on peut y accéder via virtualbox ou bien en utilisant la commande `Vagrant ssh`.
- ✚ On a intérêt à créer plus qu'une machine virtuelle, et c'est une des avantages du vagrant qui nous permet de créer et configurer plusieurs machines dans un seul fichier `vagrantfile`. On va créer 3 machines virtuelles, en procédant comme indique le fichier de configuration `vagrantfile`

```

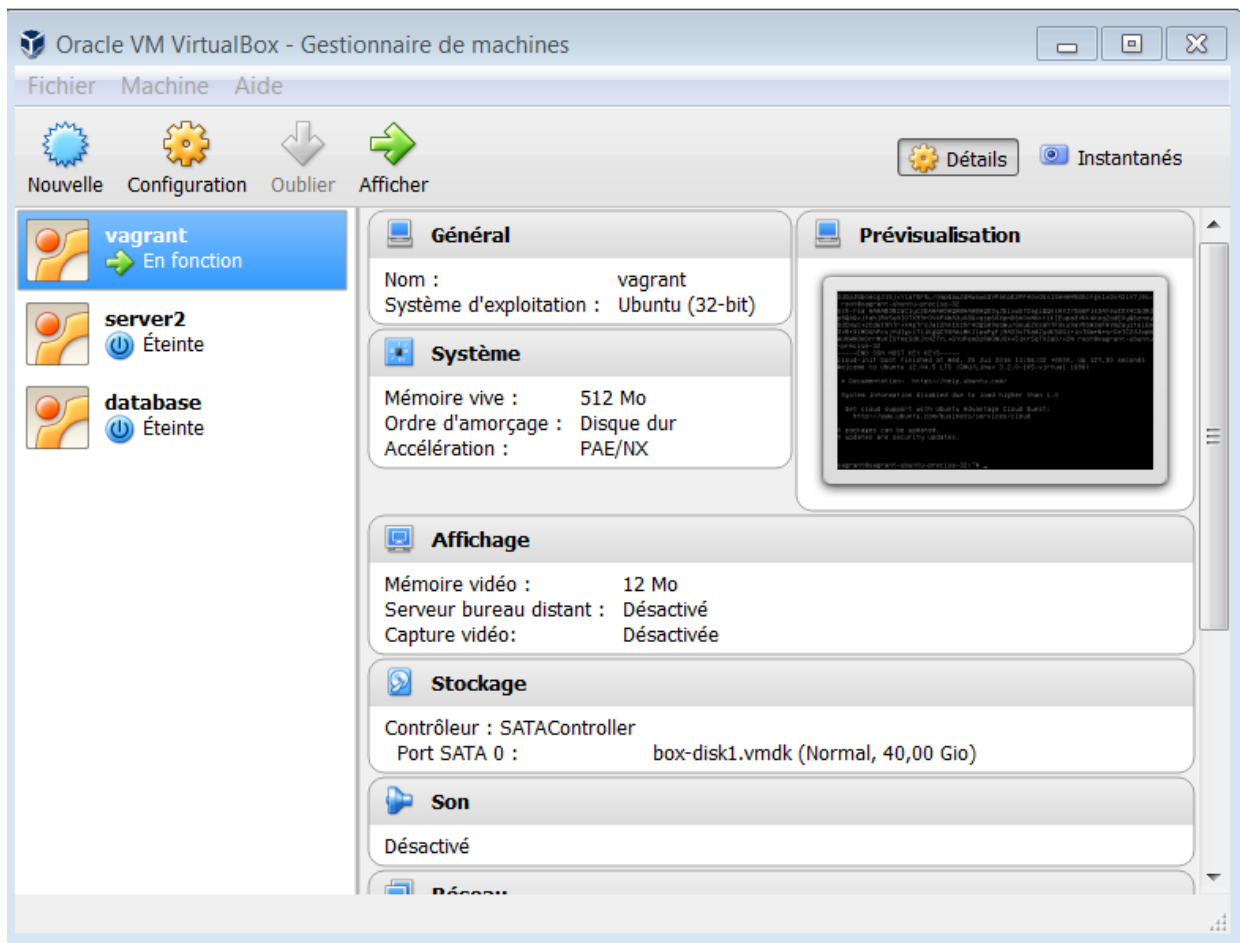
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/precise32"
  config.ssh.insert_key = false
# Application server 1.
  config.vm.define "vagrant" do |vagrant|
    vagrant.vm.hostname = "vagrant"
    vagrant.vm.box = "ubuntu/precise32"
#configuration du réseau de VM ,la machine appartient à un réseau privé :192.168.224.0 et on a attribué à
cette machine l'adresse 192.168.224.6
    vagrant.vm.network :private_network, ip: "192.168.224.6"
    config.vm.provider :virtualbox do |v|
      v.gui = true
      v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
      v.customize ["modifyvm", :id, "--ioapic", "on"]
    end
  end
# Application server 2.
  config.vm.define "server2" do |server2|
    server2.vm.hostname = "server2"
    server2.vm.box = "ubuntu/precise32"
#configuration du réseau de VM ,la machine appartient à un réseau privé :192.168.224.0 et on a attribué à
cette machine l'adresse 192.168.224.4
    server2.vm.network :private_network, ip: "192.168.224.4"
    config.vm.provider :virtualbox do |v|
      v.gui = true
      v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
      v.customize ["modifyvm", :id, "--ioapic", "on"]
    end
  end
# Database server.
  config.vm.define "database" do |database|
    database.vm.hostname = "database"
    database.vm.box = "ubuntu/precise32"
#configuration du réseau de VM ,la machine appartient à un réseau privé :192.168.224.0 et on a attribué à
cette machine l'adresse 192.168.224.5
    database.vm.network :private_network, ip: "192.168.224.5"
    config.vm.provider :virtualbox do |v|
      v.gui = true
      v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
      v.customize ["modifyvm", :id, "--ioapic", "on"]
    end
  end
end
end

```

Donner un nom à la machine et spécifier qu'elle va utiliser image de ubuntu 32



D'après la configuration il faut juste exécuter la commande : `vagrant up`. Et les 3 machines sont créées juste utilisant une simple ligne de commande : `vagrant up`.



IV. Ansible et vagrant.

Comme on a déjà défini que ansible sert à automatiser la configuration à distance d'une ou des plusieurs machines en même temps, alors que vagrant sert à automatiser la création des machines virtuelles par le biais d'un fichier dite vagrantfile au niveau duquel on consacre un bloc de code pour la définition de la machine (ou plusieurs blocs de code pour définir plusieurs machines virtuelles), et un bloc de code pour la configuration de l'outil de provisionnement qui dans notre cas ansible, et ce bloc de code doit être dans le bloc du serveur qui va s'occuper de la gestion des configurations des autres machines créées autrement dit on a créé une machine et on a spécifié son outil de provisionnement qui est ansible.

Pour installer ansible, il y a plusieurs méthodes :

- ✓ On peut l'installer par les commandes d'installation.
- ✓ On peut l'installer à partir du code source.
- ✓ On peut l'installer par Gestionnaire de paquets (paquets management system)

Remarque

Un gestionnaire de paquets est un (ou plusieurs) outil(s) automatisant le processus d'installation, désinstallation, mise à jour de logiciels installés sur un système informatique. Le terme est surtout utilisé pour les systèmes d'exploitation basés sur Unix, tels GNU/Linux.

- Dans notre exemple on veut que le serveur « vagrant » se charge de la gestion de configuration, c'est pourquoi on configure ansible en ajoutant dans vagrantfile une partie consacrée pour l'exécution du ansible une fois la machine est créée et lancée.

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/precise32"
  config.ssh.insert_key = false
# Application server 1.
  config.vm.define "vagrant" do |vagrant|
    vagrant.vm.hostname = "vagrant"
    vagrant.vm.box = "ubuntu/precise32"
    vagrant.vm.network :private_network, ip: "192.168.224.6"
    config.vm.provider :virtualbox do |v|
      v.gui = true
      v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
      v.customize ["modifyvm", :id, "--ioapic", "on"]
    end
    vagrant.vm.provision :ansible do |ansible|
      ansible.verbose = "v"
      ansible.playbook = "playbook.yml"
      ansible.limit = 'all'
      ansible.sudo = true
      ansible.verbose = true
    end
  end
end
# Application server 2.
  config.vm.define "server2" do |server2|
    server2.vm.hostname = "server2"
    server2.vm.box = "ubuntu/precise32"
    server2.vm.network :private_network, ip: "192.168.224.4"
    config.vm.provider :virtualbox do |v|
      v.gui = true
      v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
      v.customize ["modifyvm", :id, "--ioapic", "on"]
    end
  end
end
# Database server.
  config.vm.define "database" do |database|
    database.vm.hostname = "database"
```

Ce bloc destiné pour la configuration de l'ansible

```

        database. vm. box = "ubuntu/precise32"
#configuration du réseau de VM ,la machine appartient à un réseau privé :192.168.224.0 et on a attribué à cette
machine l'adresse 192.168.224.5
        database. vm. network :private_network, ip: "192.168.224.5"
        config.vm.provider :virtualbox do |v|
            v.gui = true
            v.customize ["modifyvm", :id, "--natdnshostresolver1", "on"]
            v.customize ["modifyvm", :id, "--ioapic", "on"]
        end
    end
end
end

```

Il y a plusieurs options de l'ansible qu'on peut configurer dans vagrantfile telles que :

- ansible.ask_sudo_pass:
- ansible.ask_vault_pass:
- ansible.verbose:
- ansible.playbook:
- ansible.extra_vars: ajoute des variables additionnelles au playbook.
- ansible.host_vars: ajoute un ensemble des variables des machines pour être incluent dans le fichier inventory qui va être généré automatiquement
- ansible.groups:
- ansible.sudo_user:
- ansible.tags:
- ansible.sudo:
- ansible.skip_tags:
- ansible.limit:

Sont nombreuses les options de l'ansible mais chaque une a impacte sur la configuration de l'ansible, chaque option a une fonctionnalité.

Une fois la machine est lancée par la commande vagrant up,

Une fois la machine est créée et ouverte on installe ansible par

La commande suivante : **sudo apt-get install ansible**

Généralement vagrant a une option pour ansible, mais il est nécessaire d'installer ansible au niveau de la machine local en dehors de l'environnement de vagrant, on choisit une machine pour être un nœud responsable de la gestion de configuration où forcément ansible doit être installé pour assurer la fonction de management de configuration.

Il possible d'automatiser l'installation d'ansible dans la machine virtuelle responsable de management via vagrantfile, à l'aide d'un fichier.sh où on définit l'ensemble des opérations à effectuer pour installer ansible dans la machine local, généralement le fichier.sh contient l'ensemble des commandes shell successives ordonnées de telle sorte d'installer vagrant automatiquement, autrement dit c'est comme lorsqu'on exécute vagrantfile il fait appel au fichier.sh où on a décrit le scénario pour installer ansible et voilà comment on automatise l'installation de ansible dans le fichier vagrantfile


```
# create mgmt node
config.vm.define :mgmt do |mgmt_config|
  mgmt_config.vm.box = "ubuntu/trusty64"
  mgmt_config.vm.hostname = "mgmt"
  mgmt_config.vm.network :private_network, ip: "10.0.15.10"
  mgmt_config.vm.provider "virtualbox" do |vb|
    vb.memory = "256"
  end
  mgmt_config.vm.provision :shell, path: "bootstrap-mgmt.sh"
end
```

Cette partie qui automatise l'installation de ansible par l'exécution du bootstrap-mgmt.sh où le scénario de l'installation est décrit

Et le fichier qui sert pour l'automatisation de l'installation d'ansible sera de la forme suivante :

```
#!/usr/bin/env bash

# install ansible (http://docs.ansible.com/intro\_installation.html)

apt-get -y install software-properties-common
apt-add-repository -y ppa:ansible/ansible
apt-get update
apt-get -y install ansible

# copy examples into /home/vagrant (from inside the mgmt node)

cp -a /vagrant/examples/* /home/vagrant
chown -R vagrant:vagrant /home/vagrant

# configure hosts file for our internal network defined by Vagrantfile

cat >> /etc/hosts

# vagrant environment nodes

10.0.15.10 mgmt
10.0.15.11 lb
10.0.15.21 web1
10.0.15.22 web2
10.0.15.23 web3
10.0.15.24 web4
10.0.15.25 web5
10.0.15.26 web6
10.0.15.27 web7
10.0.15.28 web8
10.0.15.29 web9
```

Ce fichier.sh peut contenir autre installation et n'est pas seulement celle de ansible autrement dit il peut contenir une commande sert pour l'installation du Mysql-server, d'Apache, ou bien de PHP5

Et on peut configurer le fichier **hosts** où on configure le réseau internet définit par **vagrantfile** ou bien ce que l'on appelle inventory qui est le fichier au niveau duquel on configure une liste des noms des machines ou bien des @IP des machines que l'on veut gérer par la machine de management.

Le fichier host ou bien inventory file est de la forme suivante :

Remarque : On peut configurer plusieurs machine ont les mêmes caractéristiques juste en utilisant boucle qui génère un nombre demander des machines au lieu de dupliquer les blocs de code de configuration.

Exemple :

```
(1..2).each do |i|
  config.vm.define "web#{i}" do |node|
    node.vm.box = "ubuntu/trusty64"
    node.vm.hostname = "web#{i}"
    node.vm.network :private_network, ip: "10.0.15.2#{i}"
    node.vm.network "forwarded_port", guest: 80, host: "808#{i}"
    node.vm.provider "virtualbox" do |vb|
      vb.memory = "256"
    end
  end
end
```

V. les commandes Ad-hoc :

ansible permet d'exécuter des commandes ad-hoc au niveau d'une ou des plusieurs machines en même temps

Il existe plusieurs commandes Ad-hoc au niveau de ansible, il y des commandes qui servent pour :

✓ File Transfer

🚦 Transférer directement un fichier à plusieurs utilisateurs :

```
$ ansible atlanta -m copy -a "src=/etc/hosts dest=/tmp/hosts"
```

Avec Atlanta est un serveur

🚦 Changer le propriétaire et les droits d'un fichier :

```
$ ansible webservers -m file -a "dest=/srv/foo/a.txt mode=600"
$ ansible webservers -m file -a "dest=/srv/foo/b.txt mode=600 owner=mdehaan group=mdehaan"
```

✚ Créer des répertoires :

```
$ ansible webservers -m file -a "dest=/path/to/c mode=755 owner=mdehaan group=mdehaan state=directory"
```

✚ Supprimer des répertoires :

```
$ ansible webservers -m file -a "dest=/path/to/c state=absent"
```

✓ Managing Packages

Pour la gestion des paquets, il y a des commandes qui servent pour assurer cette fonctionnalité.

✚ Pour s'assurer qu'un paquet est installé et n'est mis à jour on utilise la commande Ad-hoc suivante :

```
$ ansible webservers -m apt-get -a "name=acme state=present"
```

✚ Pour s'assurer qu'un paquet est installé à une spécifique version on utilise la commande Ad-hoc suivante :

```
$ ansible webservers -m apt-get -a "name=acme-1.5 state=present"
```

✚ Pour s'assurer qu'un paquet est installé à la dernière version on utilise la commande :

```
$ ansible webservers -m apt-get -a "name=acme state=latest"
```

✚ Pour s'assurer qu'un paquet n'est pas installé on utilise la commande Ad-hoc suivante :

```
$ ansible webservers -m apt-get -a "name=acme state=absent"
```

✓ Deploying From Source Control

✚ Déployer une application Web directement du git

```
$ ansible webservers -m git -a "repo=git://foo.example.org/repo.git dest=/srv/myapp version=HEAD"
```

✓ Managing Services

✚ S'assurer qu'un service est lancé au niveau de tous les serveurs web :

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

🚦 Relancer un service au niveau de tous les serveurs web :

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

🚦 S'assurer qu'un service est stoppé au niveau de tous les serveurs web :

```
$ ansible webservers -m service -a "name=httpd state=stopped"
```

✓ Découvrir la nature de ansible qui est l'exécution en parallèle des fonction

On applique cette commande au niveau de tous serveurs grâce à l'argument « a » qui permet d'inclure tous les serveurs

```
$ ansible multi -a "hostname"
```

Cette commande retourne les noms de chaque machine virtuelle inscrite au niveau du fichier hosts, si ansible ne peut pas accéder à un serveur il affiche une erreur pour ce serveur et continuer l'exécution de la commande pour les autres serveurs
Voilà les résultats de la commande :

```
# run results:
192.168.60.5 | success | rc=0 >>
orc-app2.dev
192.168.60.6 | success | rc=0 >>
orc-db.dev
192.168.60.4 | success | rc=0 >>
orc-app1.dev
```

```
root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -a "hostname"
192.168.230.54 | SUCCESS | rc=0 >>
database

192.168.230.88 | SUCCESS | rc=0 >>
server2

127.0.0.1 | SUCCESS | rc=0 >>
vagrant-ubuntu-precise-32
```

✓ Configurer des serveurs d'application

```
$ ansible app -s -m apt-get -a "name=MySQL-python state=present"
$ ansible app -s -m apt-get -a "name=python-setuptools state=present"
$ ansible app -s -m easy_install -a "name=django"
```

✓ Configurer des serveurs de base de données

```
$ ansible db -s -m apt-get -a "name=mariadb-server state=present"
$ ansible db -s -m service -a "name=mariadb state=started enabled=yes"
```

```
$ ansible db -s -a "iptables -F"
```

```
$ ansible db -s -a "iptables -A INPUT -s 192.168.60.0/24 -p tcp \
-m tcp --dport 3306 -j ACCEPT"
```

✓ Savoir des informations à propos d'un fichier

```
$ ansible multi -m stat -a "path=/etc/environment"
```

```
root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -m stat -a "path=/etc/environment"
192.168.230.54 | SUCCESS => {
  "changed": false,
  "stat": {
    "atime": 1469605879.605661,
    "checksum": "f0765c471ff7eb2ef43553bbbfcf0daf50ce0656",
    "ctime": 1467330214.082637,
    "dev": 2049,
    "exists": true,
    "gid": 0,
    "gr_name": "root",
    "inode": 1197,
    "isblk": false,
    "ischr": false,
    "isdir": false,
    "isfifo": false,
    "isgid": false,
    "islnk": false,
    "isreg": true,
    "issock": false,
    "isuid": false,
    "md5": "44ad415fac749e0c39d6302a751db3f2",
    "mode": "0644",
    "mtime": 1467330093.906637,
    "nlink": 1,
    "path": "/etc/environment",
    "pw_name": "root",
    "rgrp": true,
    "roth": true,
    "rusr": true,
    "size": 79,
    "uid": 0,
    "wgrp": false,
    "woth": false,
    "wusr": true,
    "xgrp": false,
    "xoth": false,
    "xusr": false
  }
}
```

```
192.168.230.88 | SUCCESS => {
  "changed": false,
  "stat": {
    "atime": 1469604672.3104918,
    "checksum": "f0765c471ff7eb2ef43553bbbfcf0daf50ce0656",
    "ctime": 1467330214.082637,
    "dev": 2049,
    "exists": true,
    "gid": 0,
    "gr_name": "root",
    "inode": 1197,
    "isblk": false,
    "ischr": false,
    "isdir": false,
    "isfifo": false,
    "isgid": false,
    "islnk": false,
    "isreg": true,
    "issock": false,
    "isuid": false,
    "md5": "44ad415fac749e0c39d6302a751db3f2",
    "mode": "0644",
    "mtime": 1467330093.906637,
    "nlink": 1,
    "path": "/etc/environment",
    "pw_name": "root",
    "rgrp": true,
    "roth": true,
    "rusr": true,
    "size": 79,
    "uid": 0,
    "wgrp": false,
    "woth": false,
    "wusr": true,
    "xgrp": false,
    "xoth": false,
    "xusr": false
  }
}
```

```

127.0.0.1 | SUCCESS => {
  "changed": false,
  "stat": {
    "atime": 1469604644.101721,
    "checksum": "f0765c471ff7eb2ef43553bbbfcf0daf50ce0656",
    "ctime": 1467330214.082637,
    "dev": 2049,
    "exists": true,
    "gid": 0,
    "gr_name": "root",
    "inode": 1197,
    "isblk": false,
    "ischr": false,
    "isdir": false,
    "isfifo": false,
    "isgid": false,
    "islnk": false,
    "isreg": true,
    "issock": false,
    "isuid": false,
    "md5": "44ad415fac749e0c39d6302a751db3f2",
    "mode": "0644",
    "mtime": 1467330093.906637,
    "nlink": 1,
    "path": "/etc/environment",
    "pw_name": "root",
    "rgrp": true,
    "roth": true,
    "rusr": true,
    "size": 79,
    "uid": 0,
    "wgrp": false,
    "woth": false,
    "wusr": true,
    "xgrp": false,
    "xoth": false,
    "xusr": false
  }
}

```

✓ Savoir les informations de chaque machines virtuelles

```
$ ansible multi -a "df -h"
```

Les résultats comme suivant :

```
192.168.60.6 | success | rc=0 >>
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/mapper/centos-root	19G	1014M	18G	6%	/
devtmpfs	111M	0	111M	0%	/dev
tmpfs	120M	0	120M	0%	/dev/shm
tmpfs	120M	4.3M	115M	4%	/run
tmpfs	120M	0	120M	0%	/sys/fs/cgroup
/dev/sda	1497M	124M	374M	25%	/boot
none	233G	217G	17G	94%	/vagrant

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -a "df -h"
192.168.230.88 | SUCCESS | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       40G   927M   37G   3% /
udev            246M   12K   246M   1% /dev
tmpfs           50M    240K   50M   1% /run
none            5.0M     0   5.0M   0% /run/lock
none            248M     0   248M   0% /run/shm

192.168.230.54 | SUCCESS | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       40G   965M   37G   3% /
udev            246M   12K   246M   1% /dev
tmpfs           50M    236K   50M   1% /run
none            5.0M     0   5.0M   0% /run/lock
none            248M     0   248M   0% /run/shm

127.0.0.1 | SUCCESS | rc=0 >>
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       40G   1.2G   37G   4% /
udev            246M   12K   246M   1% /dev
tmpfs           50M    240K   50M   1% /run
none            5.0M     0   5.0M   0% /run/lock
none            248M   152K   248M   1% /run/shm

```

✓ Vérifier la connectivité de ansible.

Pour vérifier la connectivité de ansible on utilise la commande suivante, sachant qu'un fichier d'inventaire (inventory file) contient toutes les machines cibles.

```
ansible all -m ping
ou bien la commande suivante qui fait la même chose
ansible -i hosts -m ping all
```

Les résultats favorables de cette commande est comme suivant :

```
web1 | success >> {
  "changed": false,
  "ping": "pong"
}

web2 | success >> {
  "changed": false,
  "ping": "pong"
}

web5 | success >> {
  "changed": false,
  "ping": "pong"
}

web4 | success >> {
  "changed": false,
  "ping": "pong"
}

web3 | success >> {
  "changed": false,
  "ping": "pong"
}

web6 | success >> {
  "changed": false,
  "ping": "pong"
}

lb | success >> {
  "changed": false,
  "ping": "pong"
}
```

Exemple pour notre cas :

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -m ping
192.168.230.88 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
192.168.230.94 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}

```

Une autre commande :

```
ansible all -a "whoami" --sudo -K
```

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -a "whoami" --sudo -K
SUDO password:
192.168.230.54 | SUCCESS | rc=0 >>
root
192.168.230.88 | SUCCESS | rc=0 >>
root

```

```
Ansible all -a "/bin/echo hello " -k
```

```

root@vagrant-ubuntu-precise-32:/etc/ansible# nano hosts
root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -a "/bin/echo hello " -k
SSH password:
192.168.230.88 | SUCCESS | rc=0 >>
hello
192.168.230.54 | SUCCESS | rc=0 >>
hello
127.0.0.1 | SUCCESS | rc=0 >>
hello

```

```
ansible all -a "date"
```

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible all -a "date"
192.168.230.54 | SUCCESS | rc=0 >>
Wed Jul 27 13:25:58 UTC 2016
192.168.230.88 | SUCCESS | rc=0 >>
Wed Jul 27 13:25:58 UTC 2016
127.0.0.1 | SUCCESS | rc=0 >>
Wed Jul 27 13:26:10 UTC 2016
root@vagrant-ubuntu-precise-32:/etc/ansible# █

```


VI. Les modules d'ansibles.

Les modules de ansible sont des scripts prédéfinis assurent une fonctionnalité spécifique, il suffit les appeler lors d'une exécution d'une opération. Ces scripts dites les modules sont sans extension et ils peuvent être écrits n'importe quel langage tel que : bash, C++, clojure, Python, Ruby.

au niveau de ansible il y a douzaine des modules tels que : copy module, file module, service module, git module, apt module, command module, fetch module, setup module, user module, cloud module, shell module, network module, database module, inventory module, Windows modules, web infrastructure module, et il y a aussi des modules qui permettent d'envoyer un mail.

- Copy module : permet de copier un fichier de la machine local vers la ou les machines cibles.

```
# Example from Ansible Playbooks
```

```
- copy: src=/srv/myfiles/foo.conf dest=/etc/foo.conf owner=foo group=foo mode=0644
```

```
# The same example as above, but using a symbolic mode equivalent to 0644
```

```
- copy: src=/srv/myfiles/foo.conf dest=/etc/foo.conf owner=foo group=foo mode="u=rw,g=r,o=r"
```

```
# Another symbolic mode example, adding some permissions and removing others
```

```
- copy: src=/srv/myfiles/foo.conf dest=/etc/foo.conf owner=foo group=foo mode="u=rw,g-wx,o-rwx"
```

```
# Copy a new "ntp.conf" file into place, backing up the original if it differs from the copied version
```

```
- copy: src=/mine/ntp.conf dest=/etc/ntp.conf owner=root group=root mode=644 backup=yes
```

```
# Copy a new "sudoers" file into place, after passing validation with visudo
```

```
- copy: src=/mine/sudoers dest=/etc/sudoers validate='visudo -cf %s'
```

Lorsqu'on fait appel à ce module on spécifie la source et la destination, et on peut aussi spécifier le propriétaire (owner) qui doit posséder ce fichier, le groupe qui aussi posséder ce fichier, et les droits d'exécution, de lecture et d'écriture donnés au utilisateur, au groupe, et aux autres. On peut spécifier un argument dite Backup qui peut prendre deux valeur booléen vrai ou faux et qui sert à Créez un fichier de sauvegarde incluant les informations d'horodatage donc vous pouvez récupérer le fichier (dossier) original si vous l'avez d'une façon ou d'une autre d'une destruction incorrectement.

- Fetch module : copie un fichier de la machine cible vers la machine local, il fonctionne comme copy module.

```
# Store file into /tmp/fetched/host.example.com/tmp/somefile
```

```
- fetch: src=/tmp/somefile dest=/tmp/fetched
```

```
# Specifying a path directly
```

```
- fetch: src=/tmp/somefile dest=/tmp/prefix-{{ inventory_hostname }} flat=yes
```

```
# Specifying a destination path
```

```
- fetch: src=/tmp/uniquefile dest=/tmp/special/ flat=yes
```

```
# Storing in a path relative to the playbook
```

```
- fetch: src=/tmp/uniquefile dest=special/prefix-{{ inventory_hostname }} flat=yes
```

- file module: permet de changer les droits, et les propriétaires d'un fichier ou supprimer des fichiers et des répertoires.

change file ownership, group and mode. When specifying mode using octal numbers, first digit should always be 0.

```
- file: path=/etc/foo.conf owner=foo group=foo mode=0644
- file: src=/file/to/link/to dest=/path/to/symlink owner=foo group=foo state=link
- file: src=/tmp/{ { item.src } } dest={ { item.dest } } state=link
with_items:
  - { src: 'x', dest: 'y' }
  - { src: 'z', dest: 'k' }
```

touch a file, using symbolic modes to set the permissions (equivalent to 0644)

```
- file: path=/etc/foo.conf state=touch mode="u=rw,g=r,o=r"
```

touch the same file, but add/remove some permissions

```
- file: path=/etc/foo.conf state=touch mode="u+rw,g-wx,o-rwx"
```

create a directory if it doesn't exist

```
- file: path=/etc/some_directory state=directory mode=0755
```

- git module
- service module: permet de vérifier l'état d'un service.
- apt module: c'est le module qui sert la gestion des fichiers

```
# Update repositories cache and install "foo" package
- apt: name=foo update_cache=yes

# Remove "foo" package
- apt: name=foo state=absent

# Install the package "foo"
- apt: name=foo state=present

# Install the version '1.00' of package "foo"
- apt: name=foo=1.00 state=present

# Update the repository cache and update package "nginx" to latest version using default release squeeze-backport
- apt: name=nginx state=latest default_release=squeeze-backports update_cache=yes

# Install latest version of "openjdk-6-jdk" ignoring "install-recommends"
- apt: name=openjdk-6-jdk state=latest install_recommends=no

# Update all packages to the latest version
- apt: upgrade=dist

# Run the equivalent of "apt-get update" as a separate step
- apt: update_cache=yes

# Only run "update_cache=yes" if the last one is more than 3600 seconds ago
- apt: update_cache=yes cache_valid_time=3600

# Pass options to dpkg on run
- apt: upgrade=dist update_cache=yes dpkg_options='force-confold,force-confdef'

# Install a .deb package
- apt: deb=/tmp/mypackage.deb

# Install the build dependencies for package "foo"
- apt: pkg=foo state=build-dep

# Install a .deb package from the internet.
- apt: deb=https://example.com/python-ppq_0.1-1_all.deb
```

VII. Playbook.

Playbook est une façon différente pour utiliser ansible autre que les commandes Ad-hoc, la différence entre playbooks et les commandes ad-hoc est :

Ad Hoc :

Exécuter des commandes rapides à partir du binaire ansible. Par exemple, exécuter une action simple qui consiste à redémarrer 10 machines de votre infrastructure (administré par Ansible).

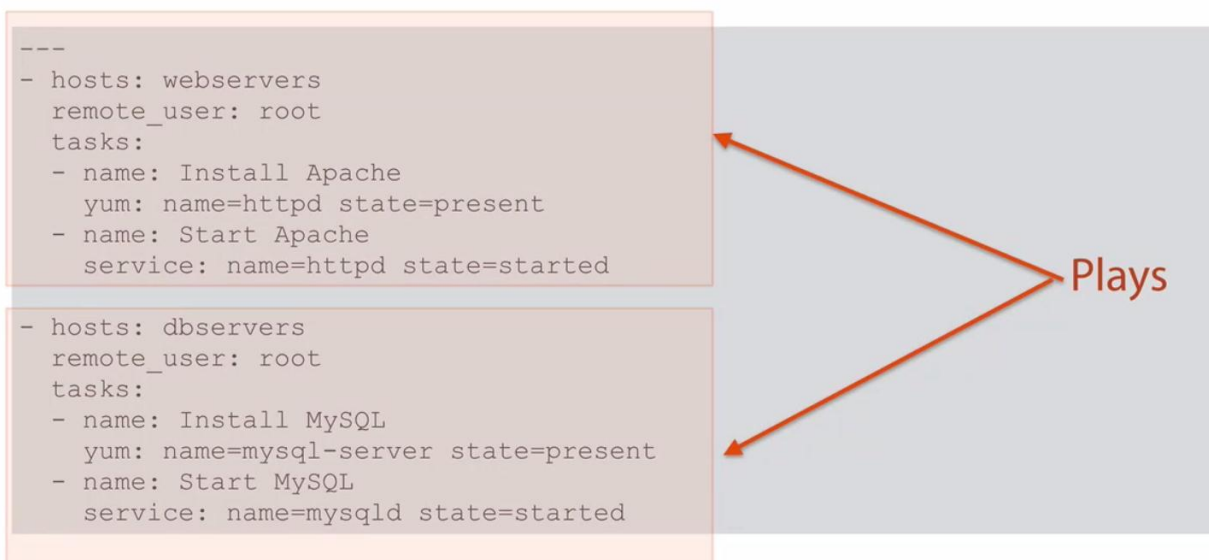
PlayBooks :

Exécuter des actions à partir du binaire ansible-playbook. Un Playbook est écrit en YAML et permet de réaliser des actions plus complexes.

Le fait que les playbooks soient écrit en YAML, rend le code facilement lisible et compréhensible, on parle généralement de code « human readable ».

Finalement, avec les « actions Ad-Hoc » vous pourrez faire des choses basiques alors qu'avec les « actions PlayBooks » vous pourrez effectuer de la véritable automatisation avancée. Les deux sont intéressants mais les PlayBooks sont la véritable force d'Ansible.

Les playbooks sont des scénarios dans lesquels sont décrits des actions que les « agents » doivent réaliser. Les fichiers playbooks sont des fichiers avec extension .yaml et ils sont très lisibles, chaque playbook constitue d'un ou des plusieurs 'plays', autrement dit, playbook est une liste des plays, plays reviennent au mapping entre un ensemble des hôtes choisis et les tâches qui fonctionnent sur ces hôtes pour définir le rôle que ces systèmes exécuteront, il peut être une ou plusieurs plays dans un playbook.



Pour utiliser le playbook pour l'automatisation des tâches, on crée un répertoire au niveau duquel on met tous les playbooks d'extension .yaml

Une fois le fichier playbook.yaml est configuré on applique au niveau du terminal de la machine serveur responsable de la gestion une commande qui déclenche l'exécution du playbook et qui :

```
Ansible-playbook hosts playbook.yaml -options
```

Handlers sont comme des tâches régulières dans le playbook ansible, mais ils sont exécutés seulement si la tâche les appelle, et au niveau du playbook on peut distinguer handler par 'notify', Handlers peuvent être définis dans une section ou bien dans un fichier appelé à la fin de chaque play.

Play est un bloc de configuration de ansible correspond à un groupe spécifique des machines.

Exemple :

```
---
- hosts: local
  tasks:
    - name: Install Nginx
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - Start Nginx

  handlers:
    - name: Start Nginx
      service: name=nginx state=started
```

```
hosts: webservers #spécification de la ou les machines cibles
vars:
  http_port: 80
  max_clients: 200
  remote_user: root
tasks: #description des caractéristiques des tâches à exécuter au niveau des machines cibles
  - name: ensure apache is at the latest version #le nom de la tâche
    yum: name=httpd state=latest # la commande de la tâche
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
      - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
```

L'ensemble des variables
nécessaire lors de la configuration

Autre exemple:

```
---
- hosts: local
  vars:
    - docroot: /var/www/serversforhackers.com/public
  tasks:
    - name: Add Nginx Repository
      apt_repository: repo='ppa:nginx/stable' state=present
      register: ppastable

    - name: Install Nginx
      apt: pkg=nginx state=installed update_cache=true
      when: ppastable|success
      register: nginxinstalled
      notify:
        - Start Nginx

    - name: Create Web Root
      when: nginxinstalled|success
      file: dest={{ '{{' }} docroot {{ '}}' }} mode=775 state=directory owner=www-data group=www-data
      notify:
        - Reload Nginx

  handlers:
    - name: Start Nginx
      service: name=nginx state=started

    - name: Reload Nginx
      service: name=nginx state=reloaded
```

Exemple: utiliser playbook pour créer un dossier sous le nom serveur dans le répertoire home.

```
root@vagrant-ubuntu-precise-32:/etc/ansible/playbooks# ansible-playbook serveur.yml -k -s
SSH password:

PLAY [all] *****

TASK [setup] *****
ok: [127.0.0.1]
ok: [192.168.230.26]
ok: [192.168.230.45]

TASK [create directory] *****
changed: [127.0.0.1]
[WARNING]: Consider using file module with state=directory rather than running mkdir
changed: [192.168.230.45]
changed: [192.168.230.26]

PLAY RECAP *****
127.0.0.1          : ok=2    changed=1    unreachable=0    failed=0
192.168.230.26    : ok=2    changed=1    unreachable=0    failed=0
192.168.230.45    : ok=2    changed=1    unreachable=0    failed=0
```

Et voilà le dossier serveur est créé au niveau des 3 serveurs

```
vagrant@database:/home$ ls
serveur  ubuntu  vagrant
vagrant@database:/home$
```

```
vagrant@server2:/home$ ls
serveur  ubuntu  vagrant
vagrant@server2:/home$
```

```
root@vagrant-ubuntu-precise-32:/home# ls
serveur  ubuntu  vagrant
root@vagrant-ubuntu-precise-32:/home#
```

Exemple2 : installer apach2 au niveau du server2

```
root@vagrant-ubuntu-precise-32:/etc/ansible/playbooks# ansible-playbook demo.yml -k -s
SSH password:
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo' (default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.

PLAY [192.168.230.26] *****

TASK [setup] *****
ok: [192.168.230.26]

TASK [Installs apatch server] *****
changed: [192.168.230.26]

PLAY RECAP *****
192.168.230.26      : ok=2    changed=1    unreachable=0    failed=0
```

demo.yml

```
--
- hosts: 192.168.230.26
  sudo: yes
  tasks:
    - name: Installs apach server
      apt: name=apach2 state=installed
```

Exemple3 : installer mysql server

```

root@vagrant-ubuntu-precise-32:/etc/ansible/playbooks# ansible-playbook demo.yml -k -s
SSH password:
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo' (default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.

PLAY [192.168.230.45] *****

TASK [setup] *****
ok: [192.168.230.45]

TASK [Installs mysql server] *****
changed: [192.168.230.45]

PLAY RECAP *****
192.168.230.45 : ok=2    changed=1    unreachable=0    failed=0

```

demo.yml

```

- --
- hosts: 192.168.230.45
  sudo: yes
  tasks:
    - name: Installs mysql server
      apt: name=mysql-server state=installed

```

Voilà mysql server est installé

```

root@database:/home# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.5.50-0ubuntu0.12.04.1 (Ubuntu)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █

```


VIII. Les rôles.

Un rôle est un playbook structuré pour répondre à un usage plus large et réutilisable, tel que l'installation d'une suite logicielle ou le paramétrage complet d'un système en fonction d'un usage.

La structure d'un répertoire rôle : le répertoire rôle contient des sous-répertoires suivants : files, handlers, meta, templates, tasks, vars.

- ✓ Files : où on met les fichiers qu'on veut ajouter à des machines virtuelles, la plupart du temps les fichiers dans files sont référencés par la tâche copy.
- ✓ Handlers : où on liste les handlers qui vont être ajoutés au play
- ✓ Vars : contient la liste des variables qui vont être ajoutés au play.
- ✓ Meta :
- ✓ Templates :
- ✓ Tasks : contient toutes les tâches qui vont être dans le playbook

on crée un dossier roles au niveau duquel on crée des sous-répertoires qu'on a mentionné précédemment, et on crée un fichier main.yml comme suivant :

```
./files:
main.yml

./handlers:
main.yml

./meta:
main.yml

./tasks:
main.yml

./templates:
main.yml

./vars:
main.yml
```

Et forme du playbook structuré sera comme le suivant :

```
—
- hosts: all
  roles: rôles
```

Après avoir configuré les rôles il suffit juste exécuter la commande :

```
ansible-playbook -s playbook.yml
```

Exemple illustratif:

Voilà un playbook basique à partir de ce playbook on va essayer de créer un playbook structure qui est le rôle.

```
---
- hosts: droplets
  tasks:
    - name: Installs nginx web server
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start nginx

    - name: Upload default index.php for host
      copy: src=static_files/index.php dest=/usr/share/nginx/www/ mode=0644
      register: php
      ignore_errors: True

    - name: Remove index.html for host
      command: rm /usr/share/nginx/www/index.html
      when: php|success

    - name: Upload default index.html for host
      copy: src=static_files/index.html dest=/usr/share/nginx/www/ mode=0644
      when: php|failed

  handlers:
    - name: start nginx
      service: name=nginx state=started
```

Etape 1 : créer un dossier rôles au niveau duquel on créer un dossier sous le nom nginx, et dans ce dossier on crée deux sous-dossiers un s'appelle tasks et autre handlers .

Etape 2 : dans le dossier tasks on crée un fichier main.yml qui va contenir juste les lignes en rouge.

```
---
- hosts: droplets
  tasks:
    - name: Installs nginx web server
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start nginx

    - name: Upload default index.php for host
      copy: src=static_files/index.php dest=/usr/share/nginx/www/ mode=0644
      register: php
      ignore_errors: True
```

```
- name: Remove index.html for host
  command: rm /usr/share/nginx/www/index.html
  when: php|success

- name: Upload default index.html for host
  copy: src=static_files/index.html dest=/usr/share/nginx/www/ mode=0644
  when: php|failed
```

handlers:

```
- name: start nginx
  service: name=nginx state=started
```

Le fichier tasks/main.yml est de la forme suivante:

```
---
- name: Installs nginx web server
  apt: pkg=nginx state=installed update_cache=true
  notify:
    - start nginx

- name: Upload default index.php for host
  copy: src=index.php dest=/usr/share/nginx/www/ mode=0644
  register: php
  ignore_errors: True

- name: Remove index.html for host
  command: rm /usr/share/nginx/www/index.html
  when: php|success

- name: Upload default index.html for host
  copy: src=index.html dest=/usr/share/nginx/www/ mode=0644
  when: php|failed
```

Etape 3: au niveau du dossier handlers on crée aussi un fichier main.yml où on met les lignes de handlers

```
---
- hosts: droplets
  tasks:
    - name: Installs nginx web server
      apt: pkg=nginx state=installed update_cache=true
      notify:
        - start nginx

    - name: Upload default index.php for host
      copy: src=static_files/index.php dest=/usr/share/nginx/www/ mode=0644
      register: php
      ignore_errors: True
```

```
- name: Remove index.html for host
  command: rm /usr/share/nginx/www/index.html
  when: php|success

- name: Upload default index.html for host
  copy: src=static_files/index.html dest=/usr/share/nginx/www/ mode=0644
  when: php|failed

handlers:
- name: start nginx
  service: name=nginx state=started
```

Le fichier handlers/main.yml est comme le suivant

```
---
- name: start nginx
  service: name=nginx state=started
```

Etape 4: crée un fichier général incluant les rôles et qui représente le playbook structuré en présence des rôles playbook.yml.

```
---
- hosts: droplets
  roles:
  - role: nginx
```

Demo:

- ✓ On crée un dossier exemple au sien duquel on crée les sous-dossiers suivants : templates, files, meta, handlers, tasks, vars.

```
root@vagrant-ubuntu-precise-32:/etc/ansible# cd roles/
root@vagrant-ubuntu-precise-32:/etc/ansible/roles# cd nginx/
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx# ls
files handlers meta tasks templates vars
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx#
```

- ✓ Au niveau du sous-dossier tasks, on crée un fichier main.yml dans lequel on va décrire les tâches qu'on veut effectuer.

```
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx# cd tasks/
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx/tasks# ls
main.yml
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx/tasks#
```

- ✓ Le fichier main.yml qui contient la configuration des tâches.

```
--
- name: Ensure nginx is installed
  apt: name=nginx state=present

- name: install nginx if not installed
  apt: name=nginx state=installed
  notify:
    - start nginx

- name: Ensure nginx is at the latest version
  apt: name=nginx state=latest
```

Les tâches configurées sont des tâches pour vérifier si nginx est installé et de l'installer sinon, puis vérifier que nginx installé est la dernière version.

- ✓ On écrit un fichier main.yml dans le dossier handlers et qui va commencer nginx.

```
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx# cd handlers/
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx/handlers# ls
main.yml
root@vagrant-ubuntu-precise-32:/etc/ansible/roles/nginx/handlers#
```

```
--
- name: start nginx
  service: name=nginx state=started
```

- ✓ Voilà le playbook de base qui introduit les rôles.

```
--
- hosts: 192.168.230.76
  sudo: yes
  roles:
    - nginx
```

- ✓ L'exécution de la commande `ansible-playbook main.yml -k -s`

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible-playbook nginx.yml -k -s
SSH password:
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo' (default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.

PLAY [192.168.230.76] *****

TASK [setup] *****
ok: [192.168.230.76]

TASK [nginx : Ensure nginx is installed] *****
ok: [192.168.230.76]

TASK [nginx : install nginx if not installed] *****
ok: [192.168.230.76]

TASK [nginx : Ensure nginx is at the latest version] *****
ok: [192.168.230.76]

PLAY RECAP *****
192.168.230.76 : ok=4 changed=0 unreachable=0 failed=0

root@vagrant-ubuntu-precise-32:/etc/ansible#

```

On doit vérifier si nginx est installé, si oui on saut cette tâche, sinon on l'installe pour ce fait voilà le fichier de tâche qui rend ce service.

```

- --
- name: Ensure nginx is installed
  command: dpkg-query -W nginx
  register: result

- name: install nginx if not installed
  apt: name=nginx state=installed
  when: result.rc == 1
  notify:
    - start nginx

- name: Ensure nginx is at the latest version
  apt: name=nginx state=latest

```

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible-playbook nginx.yml -k -s
SSH password:
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo' (default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.

PLAY [192.168.230.76] *****

TASK [setup] *****
ok: [192.168.230.76]

TASK [nginx : Ensure nginx is installed] *****
changed: [192.168.230.76]

TASK [nginx : install nginx if not installed] *****
skipping: [192.168.230.76]

TASK [nginx : Ensure nginx is at the latest version] *****
ok: [192.168.230.76]

PLAY RECAP *****
192.168.230.76 : ok=3 changed=1 unreachable=0 failed=0

root@vagrant-ubuntu-precise-32:/etc/ansible#

```

Cette fois en spécifiant la version.

```
--
- name: Ensure nginx is installed
  command: dpkg-query -W nginx
  register: result

- name: install nginx if not installed
  apt: name=nginx state=installed
  when: result.rc == 1
  notify:
    - start nginx

- name: Ensure nginx is at a specific version
  apt: name=nginx=1.1.19-1ubuntu0.8 state=present
```

```
root@vagrant-ubuntu-precise-32:/etc/ansible# ansible-playbook nginx.yml -k -s
SSH password:
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo' (default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.

PLAY [192.168.230.76] *****

TASK [setup] *****
ok: [192.168.230.76]

TASK [nginx : Ensure nginx is installed] *****
changed: [192.168.230.76]

TASK [nginx : install nginx if not installed] *****
skipping: [192.168.230.76]

TASK [nginx : Ensure nginx is at a specific version] *****
ok: [192.168.230.76]

PLAY RECAP *****
192.168.230.76 : ok=3 changed=1 unreachable=0 failed=0

root@vagrant-ubuntu-precise-32:/etc/ansible#
```

Les rôles peuvent être créés comme ils peuvent être téléchargé par le biais du ansible galaxy, alors c'est quoi ansible galaxy ?

Ansible galaxy est site web gratuit pour trouver, télécharger et même partager des rôles déjà développé.

Alors pour récupérer ou bien télécharger n'importe quel rôle, on effectue la commande suivante :

```
Ansible-galaxy install username.rolename -p /etc/ansible/roles
```

Cette commande suffit pour télécharger les rôles et les mettre dans le dossier dédié pour les rôles, lors de l'utilisation du playbook, il suffit juste indiquer le nom du rôle qu'on a téléchargé.

IX. Include statement.

Cela revient à dire qu'un fichier playbook peut inclure un ou plusieurs playbooks, autrement dit on externalise les tâches et/ou les variables, les handlers dans des fichiers d'extension yml et on fait appel à ces fichiers dans le fichier playbook global

Exemple :

```
---
- hosts: all
  vars_files:
    - vars.yml
  tasks:
    - include: included-playbook.yml
```

Avec le fichier included-playbook.yml est de la forme suivante:

```
---
- name: Add profile info for user.
  copy:
    src: example_profile
    dest: "/home/{{ username }}/.profile"
    owner: "{{ username }}"
    group: "{{ username }}"
    mode: 0744
- name: Add private keys for user.
  copy:
    src: "{{ item.src }}"
    dest: "/home/.ssh/{{ item.dest }}"
    owner: "{{ username }}"
    group: "{{ username }}"
    mode: 0600
  with_items: ssh_private_keys
- name: Restart example service.
  service: name=example state=restarted
```

Exemple: handler includes

```
handlers:
  - include: included-handlers.yml
```

Exemple: playbook includes

```
- hosts: all
  remote_user: root
  tasks:
    - include: web.yml
    - include: db.yml
```


Exemple complet includes

```
---  
- hosts: all  
vars_files:  
  - vars.yml  
handlers:  
  - include: handlers/handlers.yml  
  
tasks:  
  - include: tasks/common.yml  
  - include: tasks/apache.yml  
  - include: tasks/php.yml  
  - include: tasks/mysql.yml  
  - include: tasks/composer.yml  
  - include: tasks/drush.yml  
  - include: tasks/drupal.yml
```




Automatisation d'une architecture avec Nginx et Wakanda

L'architecture est constituée de deux serveurs Wakanda et un serveur Nginx, et une application réalisée par l'outil Wakanda Studio, et on va essayer d'accéder à l'application via le serveur Nginx.

I. C'est quoi Wakanda ?

Wakanda c'est une Platform open développement intégrée et complète, permet de construire et livrer des applications Web orientée business, les applications Web orientée business sont des applications sur un serveur et accessible via Web browser, tablette ou les Smartphones.

La Platform Wakanda constitue de 3 composants principaux :

-  Wakanda server
-  Wakanda studio
-  Java-script framework

La Platform Wakanda fournit les outils nécessaires pour créer et exécuter une application Web. Les autres environnements de développement demandent différentes technologies et les outils tels que un serveur d'application, serveur http, un serveur de base de données, framework, etc qui travaillent ensemble. Wakanda fournit tous dans un seul environnement.

II. Présentation de l'application créée sur Windows.

L'application créée avec Wakanda Studio est pour la gestion de l'inscription dans une école et qui a comme une interface, l'interface suivante :

A Propos de l'école !



L'Institut National des Postes et Télécommunications - INPT Rabat - est une école d'ingénieurs spécialisée dans le domaine des technologies d'information. Depuis sa création, l'INPT Rabat formait chaque année une centaine d'ingénieurs dans des filières de télécoms. Il est à noter aussi que INPT Rabat fait partie du réseau des écoles d'ingénieurs au Maroc, et qui assurent la formation en métier d'ingénieur.

A Propos des cours assurés.

ID	nom_cours	discription
1	mathématique	la maîtrise des outils ...
2	administration système	la familiarisation avec...
3	réseau d'entreprise	la maîtrise des concep...
5	conception des réseaux	Etudes et l'optimisatio...
6	marketing	savoir tous les concep...
7	cloud computing	la familiarisation avec...
8	programmation orien...	la maîtrise de la progr...
9	algorithmique	acquisition d'esprit de...
10	micro-economie	
11	système d'information	
12	management des projet	acquisition des compé...

19 item(s)



[Accueil](#)
[Cours](#)
[Professeur](#)
[Etudiant](#)
[Inscription](#)

recherche professeur:

nom_cours

0

système d'information

0 item(s)



[Accueil](#)
[Cours](#)
[Professeur](#)
[Etudiant](#)
[Inscription](#)

rechercher etudiant:

nom_cours

0 item(s)

Accueil Cours Professeur Etudiant **Inscription**

S'inscrire aux cours libres.

Create

nom_etudiant

Lesson

nom_cours

Save

ID	nom_cours
1	mathématique
2	administration système
3	réseau d'entreprise
5	conception des réseaux
6	marketing
7	cloud computing
8	programmation orien...
9	algorithmique
10	micro-economie
11	système d'information

19 item(s)

Le module de base de données de cette application web.

Lesson			
Attributes			
ID	232	long	
nom_cours	T	string	
discription	T	string	
studentCollection	1-n	StudentCollection	cours_choisit
professeurCollection	1-n	ProfesseurCollection	cours_assurés
Methods			

Student			
Attributes			
ID	232	long	
nom_etudiant	T	string	
cours_choisit	n-1	Lesson	
Methods			

Professeur			
Attributes			
ID	232	long	
nom_professeur	T	string	
contact	T	string	
cours_assurés	n-1	Lesson	
Methods			

III. L'installation du Wakanda server sur Linux.

Pour installer Wakanda serveur on va utiliser les rôles :

La première des étapes est de créer un répertoire wakanda-server dans le répertoire dédié pour les rôles, au niveau duquel on crée un répertoire consacré pour les tâches que ce rôle va effectuer, dans un fichier main.yml

```
--
- name: download wakanda server package
  get_url: url=https://github.com/Wakanda/wakanda-digital-app-factory/releases/download/1.0.3/wakanda-server-community_1.0.3$

- name: install libcap-ng-utils
  command: apt-get install libcap-ng-utils

- name: install wakanda-server
  apt: deb=/tmp/wakanda-server-community_1.0.3-197492_amd64.deb

- name: update
  apt: update_cache=yes
```

Et pour le fichier.yml global est le suivant :

```
--
- hosts: 192.168.224.11
  sudo: yes
  roles:
    - wakanda-server
```

Et après on exécute de la commande `ansible-playbook wakanda.yml -k -s`

```

root@vagrant-ubuntu-precise-32:/etc/ansible# ansible-playbook wakanda.yml --ask-sudo-pass
SUDO password:
[DEPRECATION WARNING]: Instead of sudo/sudo_user, use become/become_user and make sure become_method is 'sudo' (default).
This feature will be removed in a future release. Deprecation warnings can be disabled by setting deprecation_warnings=False
in ansible.cfg.

PLAY [192.168.224.11] *****

TASK [setup] *****
ok: [192.168.224.11]

TASK [wakanda-server : download wakanda server package] *****
changed: [192.168.224.11]

TASK [wakanda-server : install libcap-ng-utils] *****
changed: [192.168.224.11]
[WARNING]: Consider using apt module rather than running apt-get

TASK [wakanda-server : install wakanda-server] *****
changed: [192.168.224.11]

TASK [wakanda-server : update] *****
ok: [192.168.224.11]

PLAY RECAP *****
192.168.224.11      : ok=5    changed=3    unreachable=0    failed=0

root@vagrant-ubuntu-precise-32:/etc/ansible# █

```

III. C'est quoi alors Nginx ?

Nginx est un serveur web ou bien un serveur http qui sert à héberger des sites Web, nginx est un serveur http qui est capable de comprendre les requêtes faites par le navigateur et de renvoyer les bonnes informations en fonction.

Ce qui diffère Nginx des autres serveurs Web est sa performance lors d'un trafic important, nginx est capable de répondre à plus de 10000 requêtes simultanées.

Dans l'architecture à réaliser, on installe nginx au niveau du serveur en utilisant les rôles et le playbook comme suivant.

```

--
- name: install nginx
  apt: name=nginx state=installed
  notify:
    - start nginx
- name: update
  apt: update_cache=yes

```

Conclusion

Ansible est une plate-forme logicielle libre pour la configuration et la gestion des ordinateurs. Elle combine le déploiement de logiciels multi-nœuds, l'exécution des tâches ad-hoc, et la gestion de configuration. Elle gère les différents nœuds par-dessus SSH et ne nécessite l'installation d'aucun logiciel supplémentaire à distance sur eux. Les modules fonctionnent grâce à JSON et à la sortie standard et peuvent être écrits dans n'importe quel langage de programmation. Le système utilise YAML pour exprimer des descriptions réutilisables de systèmes.

La plate-forme a été créée par Michael DeHaan, l'auteur de l'application serveur de provisioning Cobbler et coauteur du cadre de développement Func pour l'administration à distance. Les utilisateurs de Ansible comprennent le Fedora Project, Hewlett-Packard Allemagne, Basho Technologies et l'Université Aristote de Thessalonique Il est inclus dans le cadre de la distribution Linux Fedora, propriété de Red Hat inc., et est également disponible pour Red Hat Enterprise Linux, CentOS et Scientific Linux via les paquets supplémentaires "Extra Packages for Enterprise Linux"