Universite des Antilles

TP : Génie logiciel

TP : Installation d'Ansible et écriture de son 1er rôle

29/10/2018

Attributs du document	
Objet du document	**Rapport TP M2 INFO en Génie Logiciel**
Référence du document	**Réf :**
Nom du client	**Public**

Auteur		
Nom	**Fonction**	**Contact**
JEAN-CHARLES Loïc	**Etudiant en M2 Informatique**	

Destinataires principaux			
Nom	**Fonction**	**Société**	
Mr WATTÉ	**Professeur**	**Université des Antilles**	Université des Antilles

Critères de diffusion		
Confidentiel UA	**Confidentiel Client**	**Public**
[ ]	**[ ]**	**[X]**

TP : Installation d'Ansible et écriture de son 1er rôle

1. Question

L'export de clé, permet une connexion automatisée, sans fournir de mot de passe au prompt. Comment va-t-on faire pour les invocations de $sudo$?

$user@machine$: ~$ $sudo$ $< command >$

$[sudo]$ $Mot\ de\ passe\ de\ user$:

Comment exécuter une commande root avec un utilisateur normal, à distance, en SSH entre deux machines Linux ?

1. Il faut sur la machine distante, éditer en tant que $root$, le fichier de configuration de $sudo$: $vi\ /etc/sudoers$
2. Ajouter une ligne, pour $user$ en lui donnant tous les droits si cette ligne n'est pas déjà présente : $user\quad ALL = (ALL)\qquad ALL$
3. On se place sur la machine de lancement, et on saisit la commande suivante : $ssh\quad user@machine\quad sudo\quad /usr/sbin/< command >$

2. Préparation

a) Qu'est-ce qu'un $role$?

Le $role$ va permettre de diviser de lourdes et longues actions en tâches plus petites et simplifiées. Ce dernier, va répartir un $playbook$ en plusieurs dossiers et fichiers dont chacun aura une fonction bien spécifique. Par exemple, on mettra les variables dans une partie séparée, les $handlers$ dans une autre puis les templates dans une partie dédiée.

```
baseline/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Exemple de structure d'un $role$ qui se nomme $baseline$.

b) Qu'est-ce qu'un $playbook$?

L'élément central d'Ansible est le $playbook$. C'est un fichier en YAML qui décrit les tâches qu'Ansible doit accomplir sur nos serveurs. Il utilise une syntaxe très simple : on définit les hôtes, les variables éventuelles, puis on crée des tâches. Chaque tâche possède un nom et appelle des modules (les mêmes qu'en ligne de commande).

Exemple de $playbook$ créant un fichier avec le contenu hello-world

```
---
- name: Exemple hello - world
  hosts: ansibleclient01.local
  tasks:
    - name: Crée un fichier '/tmp/testfile.txt' avec 'hello world'.
      copy:
            src:  ./files/testfile.txt
            dest: /tmp/testfile.txt
...
```

3. Résumé de l'installation pour la configuration du serveur Ansible

1. Installation et configuration SSH

1) **Installation d'Ansible** sur la machine serveur Debian avec la commande : $sudo\ apt\ install\ ansible$
2) **Vérification de l'installation** sur la machine serveur Debian avec la commande : $ansible - galaxy\ --version$
3) **Génération d'une clé publique/privée rsa** sur la machine serveur Debian avec la commande : $ssh - keygen$
4) **Récupération des adresses ip** des machines **clientes** Debian et Ubuntu avec la commande : $ip\ address$ Exemple : inet 192.168.0.23
5) **Copie de la clé publique** du serveur vers la machine clientes avec la commande : $ssh - copy - id\ 192.168.0.23$
6) **Connexion à la machine cliente en SSH** pour tester avec la commande : $ssh\ 192.168.0.23$ puis $exit$ pour se déconnecter

2. Création du dossier projet pour placer les fichiers Ansible

$freshloic@debian$: ~$ $mkdir\ ua - m2 - ec914 - ansible$
$freshloic@debian$: ~$ $cd\ ua - m2 - ec914 - ansible$

3. Création du fichier hosts

$[nom\_de\_groupe\_pour\_designer\_les\_machines]$

$Ip\_de\_la\_machine\_cliente \quad type\_connection \;\; utiliseur\_utilisé \;\; méthode\_d\_acces\_aux\_privileges$

Exemple :

$[groupeMachine1]$

$192.168.0.26 \;\; ansible\_connection = ssh \;\; ansible\_user = freshloic$

$ansible\_become\_method = su \;\; ou \; sudo$

4. Test de la communication d'Ansible avec les machines clientes présentes dans le fichier hosts

```
freshloic@debian: ~/ua − m2 − ec914 − ansible$ ansible − m ping all − i hosts
192.168.0.26 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
192.168.0.29 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

4. Création de notre premier rôle

1. Création du *playbook*

```
− − −
− hosts: all
  become: yes
  roles:
    − role: installation − dotdeb − backport
```

On commence d'abord le fichier YAML avec 3 traits. $Hosts : all$ sert à indiquer que l'on veut lancer le $playbook$ sur tous les groupes présents dans le fichier hosts. $Become : yes$ sert à indiquer qu'on a besoin des droits sudo pour exécuter les prochaines tâches. Pour finir, on lance le rôle $installation - dotdeb - backport.$

La commande suivante sert à lancer le $playbook$:

$ansible - playbook \; - i \; hosts \; playbook.yml \; - -ask - become - pass$

$-i$: Pour choisir notre fichier hosts

$- - ask - become - pass$: Pour demander le mot de passe SUDO au lancement

2. Création du rôle $installation - dotdeb - backport$

1. J'ai utilisé cette commande pour générer mon rôle :
   $ansible - galaxy\ init - p\ roles/installation - dotdeb - backport$

   -p : pour préciser le chemin de création du rôle.

2. J'ai partagé les différentes tâches en sous-tâches que j'appelle dans le fichier $main.yml$ du $role$ grâce à des $includes$.

   $- include: add - dotdeb - key.yml$
   $- include: install - dotdeb - repo.yml$
   $- include: include - specifics - variables.yml$
   $- include: install - backports - repo.yml$
   $- include: update - cache.yml$

   1. $On\ importe\ DotDeb\ GPG\ key.$
   2. $On\ installe\ DotDeb\ repo\ pour\ Debian\ et/ou\ Ubuntu.$
   3. $Ajout\ des\ variables\ spécifiques\ pour\ Debian\ et/ou\ Ubuntu.$
   4. $On\ installe\ le\ repo\ de\ \ Backports\ pour\ Debian\ et/ou\ Ubuntu.$
   5. $On\ met$ à $jour\ le\ cache\ apt\ si\ un\ repo\ ou\ une$ clé a été $ajouté.$

   Ma méthode pour gérer les différents systèmes, est de jouer sur la modification de variables en fonction de la distribution. En effet, c'est plus rapide pour Ansible de modifier quelques variables que de chercher des taches différentes pour chaque distribution.

   Exemple :

   $- - -$
   $- name:\ \ 2.Installer\ DotDeb\ repo\ pour\ Debian\ et/ou\ Ubuntu.$
   $apt\_repository:\ \ repo = "\{\{\ item\ \}\}"$
   $with\_items:\ \ "\{\{\ dotdeb\_sources\_Debian\ if\ ansible\_distribution\ =$
   $= \ 'Debian'\ else\ dotdeb\_sources\_Ubuntu\ \}\}"$
   $register: dotdeb\_apt\_repo\_result$

   Apt_key : $pour\ installer\ la$ clé
   $Apt\_repository$: $pour\ ajouter\ un\ repository\ au\ fichier\ sources\ list$
   $Register$: $pour\ obtenir\ un\ retour\ sur\ le$ résultat $de\ la$ tâche