



A N S I B L E

Travaux Pratiques

Introduction

Ce guide contient l'ensemble des TP qui seront pratiqués lors de la formation Ansible.

Nous allons agrémenter cette formation Ansible de plusieurs TP qui vont successivement vous permettre de constituer un déploiement Ansible assez élaboré.

D'abord assez simples, les TP se basent à chaque fois sur le résultat du TP précédent.

Si vous avez eu du mal lors d'un TP, à chaque étape, un joker vous permet de repartir d'un état correct pour continuer l'aventure sur le TP suivant.

Au fur et à mesure de TP, des questions vous seront posées sous la forme suivante :

Question

- Ceci est une question d'exemple, l'avez-vous bien lue ?

Une zone de réponse vous permet de noter votre réponse :

Réponse

Les réponses seront vérifiées en groupe à la fin de chaque TP.

Durant certains exercices, des zones d'aides ou de suggestion vous permettront d'aller au delà des concepts étudiés dans la partie théorique de la formation. Libre à vous de relever les défis qui vous seront proposés !!

Démarrage

Votre formateur vous a normalement transmis une information vous permettant de vous connecter à une première machine qui vous servira tout au long de votre formation.

Vérifiez que vous avez bien noté l'adresse IP de votre VM :

Adresse IP de ma VM de contrôle :

C'est cette machine qui servira de **machine de contrôle Ansible**.

Cette machine vous est dédiée et vous avez les droits d'administrateur, ce qui implique notamment la capacité de tout y détruire. De grands pouvoirs impliquant de grandes responsabilités, prenez-en grand soin.

Connexion Web en ligne de commande

Nous recommandons de passer par le service **gotty**, qui offre un terminal en ligne de commande à travers votre navigateur web. Pour vous connecter de cette manière, ouvrez un navigateur et tapez dans la barre d'adresse l'URL suivante:

```
http://<ip de la machine>:9090
```

Utilisez le login/mot de passe suivant:

```
student / password
```

Connexion Web graphique

Vous pouvez également utiliser l'interface graphique à distance à travers **guacamole**.

Pour vous connecter à votre machine, ouvrez un navigateur et tapez dans la barre d'adresse l'URL suivante :

```
http://<ip de la machine>:8080/guacamole
```

Une bannière de login doit apparaître.

Saisissez le login / mot de passe suivant :

```
student / password
```

Vous devez arriver sur une fenêtre qui vous permet d'ouvrir une session. Sélectionnez "Default".

À la première connexion, choisir le login ubuntu

Avant d'entrer le mot de passe, cliquer sur l'icône Gnome (Avec le pied) pour choisir la session gnome-flashback (Metacity)

Saisissez le mot de passe password avant d'appuyer sur entrée.

Connexion par Remote Desktop

Lancez mstsc

Mettez l'IP de la machine pour lancer la connexion.

Choisissez la connexion par défaut et appuyez sur OK

À la première connexion, choisir le login ubuntu

Avant d'entrer le mot de passe, cliquer sur l'icône Gnome (Avec le pied) pour choisir la session gnome-flashback (Metacity)

Saisissez le mot de passe password avant d'appuyer sur entrée.

TP #1 - Installation d'Ansible

Objectifs du TP

- Vérifier que votre accès à votre VM de contrôle Ansible est correct
- Installer ansible !!

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle
- Vous êtes familier de Linux, des commandes shells

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Installation d'Ansible

Notre premier objectif est d'installer Ansible. Une fois installé, Ansible est accessible notamment au travers de la commande :

```
$ ansible
```

Question 1.1

- Comment peut-on vérifier si la commande est déjà installée dans le système ?

Réponse 1.1

Votre formateur a déjà installé un certain nombre de pré-requis à Ansible, notamment python et pip. Vous n'avez par conséquent pas besoin de procéder à leur installation.

Une fois connecté sur votre VM, ouvrez une console et taper la commande suivante :

```
$ sudo pip install ansible==2.4
```

Vérifiez que la commande se termine bien par deux lignes semblables à :

```
...
Successfully installed MarkupSafe-1.0 PyYAML-3.12 ansible-2.4.0.0
asn1crypto-0.23.0 bcrypt-3.1.4 cffi-1.11.2 cryptography-2.1.3
enum34-1.1.6 idna-2.6 ipaddress-1.0.18 jinja2-2.10 paramiko-2.3.1
pyasn1-0.3.7 pycparser-2.18 pynacl-1.2.0 six-1.11.0
.
.
.
InsecurePlatformWarning
```

Astuce

Vous pouvez noter qu'un certain nombre de dépendances python ont été installées également, par exemple :

- jinja2, que l'on reverra plus tard dans la formation dans les templates
- paramiko qui est un des modes de connexion pouvant être utilisé par Ansible

Attention

Une partie de la difficulté d'installation a été cachée par des scripts d'initialisation fournis lors de l'instanciation des machines dans AWS.

Cette complexité est en partie due à notre volonté de fixer la version d'Ansible à 2.4, ce qui n'est pas permis par le ppa (repository ubuntu additionnel) d'Ansible.

L'installation complète par le biais du ppa est relativement simple :

```
sudo add-apt-repository -y ppa:ansible/ansible
sudo apt-get update -y
sudo apt-get install ansible
```

En revanche elle offre moins de souplesse :

- ppa:ansible/ansible pour la dernière version
- ppa:ansible/ansible-1.9 pour la dernière mise à jour d'ansible 1.9

Vérification de la version d'Ansible installée

Lancez la commande suivante :

```
$ ansible --version
```

Question 1.2

- Quelle est la version d'Ansible qui a été installée ?

Réponse 1.2

TP #2 - Inventaires et commandes simples

Objectifs du TP

- Écrire un premier inventaire
- lancer des commandes sur cet inventaire

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle
- Vous êtes familier de Linux, des commandes shell
- Vous disposez d'Ansible installé

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Création d'une structure pour notre code Ansible

Nous allons commencer par créer un répertoire pour y stocker le code Ansible que nous allons écrire durant les TP. Ce répertoire sera géré dans Git.

Ouvrez une console.

Commençons par vérifier que nous sommes bien dans le répertoire `/home/ubuntu` :

```
$ pwd
```

La commande doit retourner `/home/ubuntu`. Si ce n'est pas le cas, simplement tapez « `cd` ».

Créez par la suite un répertoire :

```
$ mkdir tp-ansible  
$ cd tp-ansible  
$ pwd
```

Vérifiez que votre répertoire courant est bien `/home/ubuntu/tp-ansible`. À partir de ce moment, et sauf indication particulière, nous allons toujours supposer que c'est le répertoire courant de votre shell. Si vous fermez et ré-ouvrez votre console, pensez bien à y retourner !

Créer ensuite la structure git pour ce répertoire :

```
$ git init .
```

La commande « `ls -a` » doit vous montrer la présence désormais d'un répertoire `.git` dans ce dossier.

Création d'un fichier d'inventaire

Nous allons maintenant créer un nouveau répertoire qui va contenir notre fichier d'inventaire :

```
$ mkdir inventories
```


Ouvrez à présent un éditeur de texte pour saisir le contenu du fichier d'inventaire. Sublime Text est un éditeur qui existe sous Windows, Linux et Mac. Il a été pré-installé sur votre VM. Il est disponible dans les applications sous la rubrique « Programmation ».

Nous allons créer un fichier `inv.ini` avec comme contenu 3 machines que nous allons contrôler avec Ansible. Votre formateur doit vous fournir, en plus de l'adresse IP de la première machine 3 nouvelles adresses IP. Au final, vous devez produire le fichier suivant :

```
# inventories/inv.ini
[load-balancers]
<ip machine1>

[app-servers]
<ip machine2>
<ip machine3>

[vms:children]
load-balancers
app-servers
```

N'oubliez pas de remplacer `<ip machine1>`, `<ip machine2>` et `<ip machine3>` par les adresses IP des machines qui vous ont été transmises.

Question 2.1

- Comment peut-on vérifier sans Ansible que les 3 machines sont joignables ?
- Comment vérifier que nous y avons accès ?

Réponse 2.1

Assurez-vous au moment de sauvegarder le fichier qu'il se trouve bien dans `/home/ubuntu/tp-ansible/inventories/` et qu'il se nomme bien `inv.ini`.

Dans une console, lancer à présent la commande suivante, pour vérifier que le fichier d'inventaire est correct :

```
$ ansible -i inventories/inv.ini all --list-hosts
```

Cette commande devrait vous afficher la liste des 3 machines.

Dans la ligne de commande précédente, remplacez «all» par successivement :

- load-balancers
- app-servers
- vms
- my-group

Question 2.2

- Combien de machines sont remontées avec l'argument load-balancers ?
- Avec app-servers ?
- Avec vms ?
- Avec my-group ?

Réponse 2.2

-
-
-
-

Première commande simple

Une fois que vous avez vérifié que l'inventaire était correctement écrit et compris par Ansible, nous allons modifier la commande pour lancer (enfin) une vraie commande sur les machines :

```
$ ansible -i inventories/inv.ini all -u ubuntu -a "hostname"
```

Vous devriez voir la première sortie de l'appel à la commande hostname sur toutes les machines. Vérifiez que les retours sont tous en succès avec un code de retour (rc) égal à 0.

Modifiez la commande précédente pour compter sur chaque machine le nombre de lignes dans le fichier /etc/passwd :

```
$ ansible -i inventories/inv.ini all -u ubuntu -a "wc -l /etc/passwd"
```

Prenez garde à bien encadrer la commande entre des guillemets !!

Vérifier que la commande distante est bien exécutée en tant qu'utilisateur ubuntu avec la commande suivante :

```
$ ansible -i inventories/inv.ini all -u ubuntu -a "whoami"
```

Recommencez la même commande sans l'option « -u ubuntu ».

```
$ ansible -i inventories/inv.ini all -a "whoami"
```

Question 2.3

- Est-ce que la commande fonctionne ?
- Pourquoi ?

Réponse 2.3

-
-

Relancer la commande en ajoutant l'option « -b » :

```
$ ansible -i inventories/inv.ini all -b -a "whoami"
```

La commande a été exécutée en tant que **root** en utilisant *sudo* pour passer administrateur. Ceci est possible car les comptes **ubuntu** utilisés dans notre TP sur les machines cibles sont autorisés à lancer n'importe quelle commande sans même à avoir à fournir un mot de passe.

le terme « -b » signifie « *become* » pour signifier qu'après la connexion, on souhaite devenir un autre utilisateur. Sans précision particulière, c'est l'utilisateur **root** qui est utilisé.

Lancez la commande `ansible --help` pour chercher l'option à ajouter pour devenir un autre utilisateur. une fois trouvée, utilisez cette option pour lancer la commande « **id** » en tant que l'utilisateur **nobody**.

Question 2.4

- Quel est l'uid de l'utilisateur nobody ?

Réponse 2.4

-

La commande ansible permet de lancer plusieurs types d'opérations (appelé **modules**), et sans précision, c'est le module `command` qui est utilisé. Ainsi, les commandes

```
$ ansible -i inventories/inv.ini all -a "whoami"
```

et

```
$ ansible -i inventories/inv.ini all -m command -a "whoami"
```

Sont équivalentes.

Regardons quelques autres modules disponibles et utilisables facilement en ligne de commande.

Pour simplement vérifier la connectivité (connexion ssh OK, compte OK, python installé), le module ping est utile :

```
$ ansible -i inventories/inv.ini all -m ping
```

Un autre module intéressant est le module setup.

Lancez le module setup sur le load-balancer :

```
$ ansible -i inventories/inv.ini load-balancers -m setup
```

Vous devriez obtenir une sortie très verbeuse composée d'une structure JSON plutôt conséquente.

Création d'un ansible.cfg simple

Ajoutez un fichier à présent un fichier ansible.cfg dans le répertoire courant (/home/ubuntu/tp-ansible) avec le contenu suivant :

```
[defaults]
inventory = inventories/inv.ini
```

Assurez-vous que désormais vous pouvez lancer les mêmes commandes que précédemment sans avoir à préciser le fichier d'inventaire à utiliser :

```
$ ansible all -a "pwd"
```

Profitions-en pour lancer une resynchronisation des caches apt qui nous servira plus tard :

```
$ ansible all -b -a "apt-get update"
```

Commit Git

C'est la fin de ce premier TP réellement productif, bravo.
Il est temps d'enregistrer votre travail dans Git.

Si vous n'êtes pas familier avec Git, voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .  
$ git commit -m "initial commit"  
$ git tag tp2
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp2
```

TP #3 - Playbooks Ansible

Objectifs du TP

- Écrire un premier playbook
- Déployer des vrais logiciels sur des machines

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle et aux VMs cibles
- Vous êtes familier de Linux, des commandes shells
- Vous disposez du code fonctionnel produit lors du TP précédent

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Répertoire de travail

Nous allons commencer par nous assurer que nous sommes bien dans notre répertoire de travail `/home/ubuntu/tp-ansible`.

```
$ pwd
```

Vérifiez que votre répertoire courant est bien `/home/ubuntu/tp-ansible`.

Premier playbook

Nous allons commencer l'écriture d'un premier playbook que nous allons nommer `install.yml`. Le fichier devra se trouver dans le répertoire `/home/ubuntu/tp-ansible`.

Nous allons créer un premier play qui installe tomcat sur les machines du groupe `app-servers`.

```
# install.yml
---

- hosts: app-servers
  gather_facts: false
  tasks:
    - name: install tomcat
      package:
        name: tomcat7
```

Nous allons explicitement préciser que l'on ne souhaite pas collecter les facts des machines, car *a priori*, nous n'avons pas besoin de cette fonction.

Lançons la commande en mode test (dry-run) pour voir ce qu'ansible-playbook souhaite réaliser :

```
$ ansible-playbook install.yml --check
```

Une erreur doit se produire avec une sortie de la forme :

```
PLAY [app-servers]
*****

TASK [install tomcat]
*****
changed: [18.194.210.238]
changed: [18.194.135.36]

PLAY RECAP
*****
18.194.135.36      : ok=1    changed=1    unreachable=0    failed=0
18.194.210.238    : ok=1    changed=1    unreachable=0    failed=0
```

Dans les versions antérieures d'Ansible, le module package nécessitait de collecter les faits (gather_facts) : Ansible devait savoir sur quelle distribution le code s'exécutait pour savoir quelle était la commande à lancer pour installer un paquet (yum, apt-get...).

Relancez la commande une seconde fois.

Question 3.1

- La sortie est-elle identique ?
- Pourquoi a-t-on toujours des lignes **changed** ?

Réponse 3.1

-
-

Relançons le playbook, mais en modifiant les options sur la ligne de commande

```
$ ansible-playbook install.yml
```

Nous devrions cette fois-ci avoir une sortie de la forme

```
ubuntu@ip-172-31-21-203:~/tp-ansible$ ansible-playbook install.yml

PLAY [app-servers]
*****

TASK [install tomcat]
*****
fatal: [18.194.210.238]: FAILED! => {"cache_update_time": 1510241596,
"cache_updated": false, "changed": false, "failed": true, "msg": "'/usr/bin/apt-
get -y -o \"Dpkg::Options::=--force-confdef\" -o \"Dpkg::Options::=--force-
confold\"      install 'tomcat7' failed: E: Could not open lock file
/var/lib/dpkg/lock - open (13: Permission denied)\nE: Unable to lock the
administration directory (/var/lib/dpkg/), are you root?\n", "rc": 100,
"stderr": "E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission
denied)\nE: Unable to lock the administration directory (/var/lib/dpkg/), are
you root?\n", "stderr_lines": ["E: Could not open lock file /var/lib/dpkg/lock -
open (13: Permission denied)", "E: Unable to lock the administration directory
(/var/lib/dpkg/), are you root?"], "stdout": "", "stdout_lines": []}
fatal: [18.194.135.36]: FAILED! => {"cache_update_time": 1510241596,
"cache_updated": false, "changed": false, "failed": true, "msg": "'/usr/bin/apt-
get -y -o \"Dpkg::Options::=--force-confdef\" -o \"Dpkg::Options::=--force-
confold\"      install 'tomcat7' failed: E: Could not open lock file
/var/lib/dpkg/lock - open (13: Permission denied)\nE: Unable to lock the
administration directory (/var/lib/dpkg/), are you root?\n", "rc": 100,
"stderr": "E: Could not open lock file /var/lib/dpkg/lock - open (13: Permission
denied)\nE: Unable to lock the administration directory (/var/lib/dpkg/), are
you root?\n", "stderr_lines": ["E: Could not open lock file /var/lib/dpkg/lock -
open (13: Permission denied)", "E: Unable to lock the administration directory
(/var/lib/dpkg/), are you root?"], "stdout": "", "stdout_lines": []}
      to retry, use: --limit @/home/ubuntu/tp-ansible/install.retry

PLAY RECAP
*****
18.194.135.36      : ok=0    changed=0    unreachable=0    failed=1
18.194.210.238    : ok=0    changed=0    unreachable=0    failed=1
```

On obtient une erreur, mince.

Cette fois-ci, au milieu du message rouge, on distingue notamment **Permission denied ... are you root?**. En effet, pour installer un paquet sous Linux, il faut des droits d'administrateur. Nous allons donc modifier le playbook comme suit :

```
# install.yml
---

- hosts: app-servers
  gather_facts: false
  become: true
  tasks:
    - name: install tomcat
      package:
        name: tomcat7
```

Refaisons une tentative. Cette fois-ci, la commande a dû mettre plus de temps à s'exécuter, signe que quelque chose s'est vraiment exécuté.

Relançons la commande une seconde fois. La commande a dû rendre la main bien plus vite cette fois-ci.

Question 3.2

- La sortie est-elle identique ?
- Pourquoi ?

Réponse 3.2

-
-

Aller plus loin dans notre playbook : gestion du service

Il est temps d'enrichir notre playbook en y ajoutant désormais le démarrage de Tomcat.

```
# install.yml
---

- hosts: app-servers
  gather_facts: false
  become: true
```

```
tasks:
- name: install tomcat
  package:
    name: tomcat7
- name: start / enable tomcat
  service:
    name: tomcat7
    enabled: true
    state: started
```

En lançant la commande, vous devriez voir le résultat suivant :

```
PLAY *****
TASK [setup] *****
ok: [52.50.186.151]
ok: [52.30.177.148]

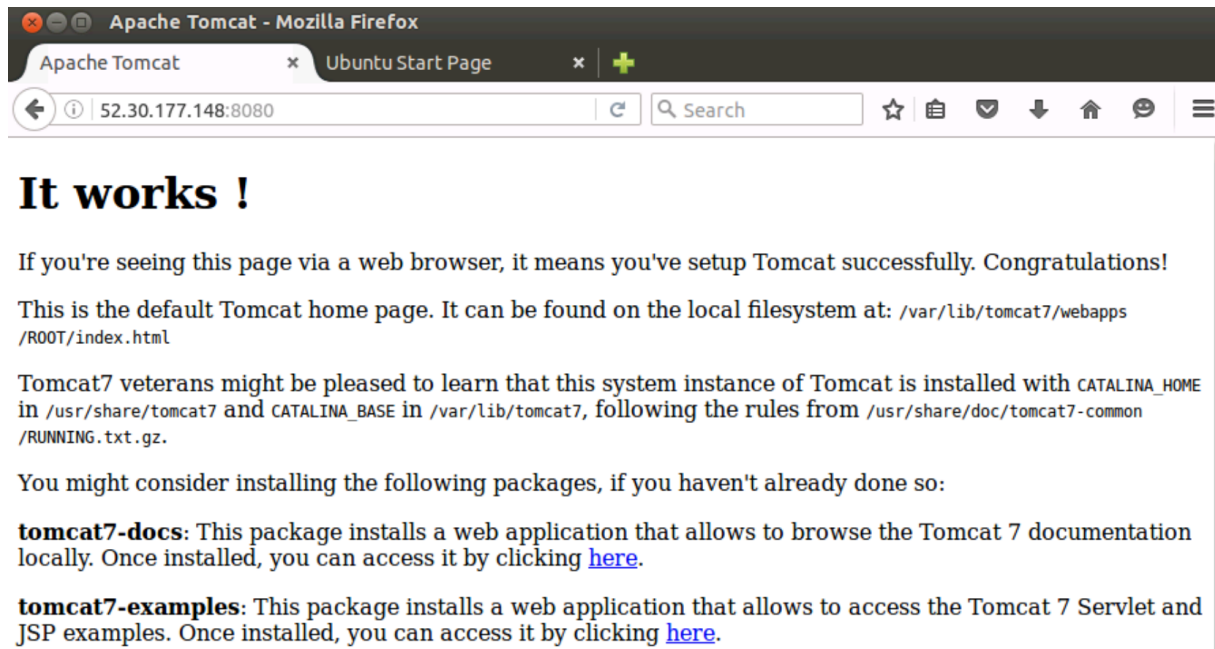
TASK [install tomcat] *****
ok: [52.30.177.148]
ok: [52.50.186.151]

TASK [start / enable tomcat] *****
ok: [52.30.177.148]
ok: [52.50.186.151]

PLAY RECAP *****
52.30.177.148      : ok=3    changed=0    unreachable=0    failed=0
52.50.186.151    : ok=3    changed=0    unreachable=0    failed=0
```

La sortie étant toute verte, aucun changement n'a eu lieu. La raison en est simple : l'installation du paquet avait déjà déclenché l'activation et le démarrage du service.

Nous pouvons nous assurer que le service est effectivement opérationnel en saisissant dans un navigateur l'URL d'une des deux machine sur le port 8080 (<http://52.30.177.148:8080> dans cet exemple) :



Tests du service

Il est important d'ajouter le maximum de tests pour valider que le service est effectivement fonctionnel, et ce, sans avoir à pratiquer de vérifications manuelles. Ces tests, écrits avec Ansible seront systématiquement exécutés et leur automatisation vous servira de harnais de non-régression.

Astuce

L'abus de tests est excellent pour la santé, à consommer sans modération

wait_for

Ajoutons un test d'écoute de Tomcat sur le port par défaut (8080) :

```
# install.yml
---
- hosts: app-servers
  gather_facts: false
  become: true
  tasks:
    - name: install tomcat
      package:
        name: tomcat7
```

```
- name: start / enable tomcat
  service:
    name: tomcat7
    enabled: true
    state: started
- name: wait for Tomcat to be up on port 8080
  wait_for:
    port: 8080
```

Pour forcer Ansible à travailler un peu, nous allons provoquer un arrêt de tomcat sur une des machines :

```
ubuntu@ip-172-31-20-24:~/tp-ansible$ ansible app-servers[0] -b -m
service -a "name=tomcat7 state=stopped"
```

Cela doit vous retourner un message de la forme suivante

```
18.194.210.238 | SUCCESS => {
  "changed": true,
  "failed": false,
  "name": "tomcat7",
  "state": "stopped",
  "status": {
    "ActiveEnterTimestamp": "Thu 2017-11-09 16:47:53 CET",
    "ActiveEnterTimestampMonotonic": "1352140195",
    "ActiveExitTimestamp": "Thu 2017-11-09 16:39:34 CET",
    "ActiveExitTimestampMonotonic": "853462928",
    "ActiveState": "active",
    "After": "remote-fs.target system.slice network-
online.target basic.target nss-lookup.target local-fs.target
sysinit.target systemd-journald.socket",
    "AllowIsolate": "no",
    "AmbientCapabilities": "0",
    "AssertResult": "yes",
    "AssertTimestamp": "Thu 2017-11-09 16:47:48 CET",
    "AssertTimestampMonotonic": "1347113300",
    "Before": "multi-user.target graphical.target
shutdown.target",
    "BlockIOAccounting": "no",
    "BlockIOWeight": "18446744073709551615",
    "CPUAccounting": "no",
    "CPUQuotaPerSecUsec": "infinity",
    "CPUSchedulingPolicy": "0",
    "CPUSchedulingPriority": "0",
    "CPUSchedulingResetOnFork": "no",
```

```

"CPUShares": "18446744073709551615",
"CPUUsageNSec": "2652918961",
"CanIsolate": "no",
"CanReload": "no",
"CanStart": "yes",
"CanStop": "yes",
"CapabilityBoundingSet": "18446744073709551615",
"ConditionResult": "yes",
"ConditionTimestamp": "Thu 2017-11-09 16:47:48 CET",
"ConditionTimestampMonotonic": "1347113300",
"Conflicts": "shutdown.target",
"ControlGroup": "/system.slice/tomcat7.service",
"ControlPID": "0",
"DefaultDependencies": "yes",
"Delegate": "no",
"Description": "LSB: Start Tomcat.",
"DevicePolicy": "auto",
"Documentation": "man:systemd-sysv-generator(8)",
"ExecMainCode": "0",
"ExecMainExitTimestampMonotonic": "0",
"ExecMainPID": "0",
"ExecMainStartTimestampMonotonic": "0",
"ExecMainStatus": "0",
"ExecStart": "{ path=/etc/init.d/tomcat7 ;
argv[]=/etc/init.d/tomcat7 start ; ignore_errors=no ;
start_time=[Thu 2017-11-09 16:47:48 CET] ; stop_time=[Thu 2017-11-
09 16:47:53 CET] ; pid=20319 ; code=exited ; status=0 }",
"ExecStop": "{ path=/etc/init.d/tomcat7 ;
argv[]=/etc/init.d/tomcat7 stop ; ignore_errors=no ;
start_time=[n/a] ; stop_time=[n/a] ; pid=0 ; code=(null) ;
status=0/0 }",
"FailureAction": "none",
"FileDescriptorStoreMax": "0",
"FragmentPath":
"/run/systemd/generator.late/tomcat7.service",
"GuessMainPID": "no",
"IOScheduling": "0",
"Id": "tomcat7.service",
"IgnoreOnIsolate": "no",
"IgnoreSIGPIPE": "no",
"InactiveEnterTimestamp": "Thu 2017-11-09 16:39:34 CET",
"InactiveEnterTimestampMonotonic": "853564525",
"InactiveExitTimestamp": "Thu 2017-11-09 16:47:48 CET",
"InactiveExitTimestampMonotonic": "1347113922",
"JobTimeoutAction": "none",
"JobTimeoutUSec": "infinity",
"KillMode": "process",
"KillSignal": "15",

```

```
"LimitAS": "18446744073709551615",
"LimitASSoft": "18446744073709551615",
"LimitCORE": "18446744073709551615",
"LimitCORESoft": "0",
"LimitCPU": "18446744073709551615",
"LimitCPUSoft": "18446744073709551615",
"LimitDATA": "18446744073709551615",
"LimitDATASoft": "18446744073709551615",
"LimitFSIZE": "18446744073709551615",
"LimitFSIZESoft": "18446744073709551615",
"LimitLOCKS": "18446744073709551615",
"LimitLOCKSSoft": "18446744073709551615",
"LimitMEMLOCK": "65536",
"LimitMEMLOCKSoft": "65536",
"LimitMSGQUEUE": "819200",
"LimitMSGQUEUESoft": "819200",
"LimitNICE": "0",
"LimitNICESoft": "0",
"LimitNOFILE": "4096",
"LimitNOFILESoft": "1024",
"LimitNPROC": "3901",
"LimitNPROCSoft": "3901",
"LimitRSS": "18446744073709551615",
"LimitRSSSoft": "18446744073709551615",
"LimitRTPRIO": "0",
"LimitRTPRIOSoft": "0",
"LimitRTTIME": "18446744073709551615",
"LimitRTTIMESoft": "18446744073709551615",
"LimitSIGPENDING": "3901",
"LimitSIGPENDINGSoft": "3901",
"LimitSTACK": "18446744073709551615",
"LimitSTACKSoft": "8388608",
"LoadState": "loaded",
"MainPID": "0",
"MemoryAccounting": "no",
"MemoryCurrent": "67137536",
"MemoryLimit": "18446744073709551615",
"MountFlags": "0",
"NFDescriptorStore": "0",
"Names": "tomcat7.service",
"NeedDaemonReload": "no",
"Nice": "0",
"NoNewPrivileges": "no",
"NonBlocking": "no",
"NotifyAccess": "none",
"OOMScoreAdjust": "0",
"OnFailureJobMode": "replace",
"PermissionsStartOnly": "no",
```

```
"PrivateDevices": "no",
"PrivateNetwork": "no",
"PrivateTmp": "no",
"ProtectHome": "no",
"ProtectSystem": "no",
"RefuseManualStart": "no",
"RefuseManualStop": "no",
"RemainAfterExit": "yes",
"Requires": "system.slice sysinit.target",
"Restart": "no",
"RestartUsec": "100ms",
"Result": "success",
"RootDirectoryStartOnly": "no",
"RuntimeDirectoryMode": "0755",
"RuntimeMaxUsec": "infinity",
"SameProcessGroup": "no",
"SecureBits": "0",
"SendSIGHUP": "no",
"SendSIGKILL": "yes",
"Slice": "system.slice",
"SourcePath": "/etc/init.d/tomcat7",
"StandardError": "inherit",
"StandardInput": "null",
"StandardOutput": "journal",
"StartLimitAction": "none",
"StartLimitBurst": "5",
"StartLimitInterval": "1000000",
"StartupBlockIOWeight": "18446744073709551615",
"StartupCPUShares": "18446744073709551615",
"StateChangeTimestamp": "Thu 2017-11-09 16:47:53 CET",
"StateChangeTimestampMonotonic": "1352140195",
"StatusErrno": "0",
"StopWhenUnneeded": "no",
"SubState": "running",
"SyslogFacility": "3",
"SyslogLevel": "6",
"SyslogLevelPrefix": "yes",
"SyslogPriority": "30",
"SystemCallErrorNumber": "0",
"TTYReset": "no",
"TTYVHangup": "no",
"TTYVTDisallocate": "no",
"TasksAccounting": "no",
"TasksCurrent": "21",
"TasksMax": "18446744073709551615",
"TimeoutStartUsec": "5min",
"TimeoutStopUsec": "5min",
"TimerSlackNsec": "50000",
```

```

    "Transient": "no",
    "Type": "forking",
    "UMask": "0022",
    "UnitFilePreset": "enabled",
    "UnitFileState": "bad",
    "UtmpMode": "init",
    "WantedBy": "multi-user.target graphical.target",
    "Wants": "network-online.target",
    "WatchdogTimestamp": "Thu 2017-11-09 16:47:53 CET",
    "WatchdogTimestampMonotonic": "1352140178",
    "WatchdogUsec": "0"
  }
}

```

Par la suite, si l'on relance le playbook, on doit obtenir enfin la preuve qu'Ansible a fait du (bon) boulot :

```

PLAY *****

TASK [setup] *****
ok: [52.50.186.151]
ok: [52.30.177.148]

TASK [install tomcat] *****
ok: [52.30.177.148]
ok: [52.50.186.151]

TASK [start / enable tomcat] *****
ok: [52.30.177.148]
changed: [52.50.186.151]

TASK [wait for Tomcat to be up on port 8080] *****
ok: [52.30.177.148]
ok: [52.50.186.151]

PLAY RECAP *****
52.30.177.148      : ok=4    changed=0    unreachable=0    failed=0
52.50.186.151    : ok=4    changed=1    unreachable=0    failed=0

```

On constate que seule une des deux machines du groupe a été modifiée. En effet, l'arrêt du service tomcat7 n'a eu lieu que sur `app-servers[0]`, ce qui correspond à la première machine du groupe `app-servers`. Il est donc logique de l'autre machine soit restée inchangée.

Astuce

On constate que le module `wait_for` ne provoque pas de `changed` (et n'en provoquera jamais). Ce n'est pas un module qui effectue de changement sur les machines cibles. Ce module attend (par défaut 300s) qu'un port soit dans un état attendu (en écoute par

défaut). La documentation vous détaillera plusieurs autres cas d'utilisation très intéressants.

test HTTP

Pour aller encore plus loin dans notre validation de l'application, nous allons ajouter l'appel à un nouveau module `uri` :

```
# install.yml
---
- hosts: app-servers
  gather_facts: false
  become: true
  tasks:
    - name: install tomcat
      package:
        name: tomcat7
    - name: start / enable tomcat
      service:
        name: tomcat7
        enabled: true
        state: started
    - name: wait for Tomcat to be up on port 8080
      wait_for:
        port: 8080
    - name: ensure home page returns 200
      uri:
        url: http://127.0.0.1:8080/
```

L'exécution du playbook se termine avec succès.

Dans les versions précédentes d'Ansible à 2.1, il y avait l'erreur "`httplib2 >= 0.7 is not installed`" ainsi écrit provoque à nouveau une erreur :

```
TASK [ensure home page returns 200]
*****
fatal: [52.30.177.148]: FAILED! => {"changed": false, "failed":
true, "msg": "httplib2 >= 0.7 is not installed"}
fatal: [52.50.186.151]: FAILED! => {"changed": false, "failed":
true, "msg": "httplib2 >= 0.7 is not installed"}
```

La raison de cette erreur est que le module `uri` qui est utilisé nécessitait une dépendance python qui devait être installée sur les machines cibles. Ce n'est plus vrai depuis Ansible 2.1 mais la démarche pour gérer des modules avec dépendances est à connaître.

Astuce

Le module uri utilisé est très puissant et peut être utilisé dans de nombreux cas. Dans le cas présent nous avons principalement utilisé son comportement par défaut :

- Interroger une URL avec la méthode HTTP GET
- Retourner une erreur sur le code HTTP de retour est différent de 200

Nous aurions pu préciser une autre méthode, injecter des headers HTTP spécifiques voire même capturer la page retournée. Mais nous verrons cela plus tard...

Commit Git

Il est temps d'enregistrer votre travail dans Git.

Voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .  
$ git commit -m "fin tp3"  
$ git tag tp3
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp3
```

TP #4 - Déploiement dynamique

Objectifs du TP

- Écrire un playbook avec un template dynamique
- Apprendre et manipuler Jinja2

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle et aux VMs cibles
- Vous êtes familier de Linux, des commandes shells
- Vous disposez du code fonctionnel produit lors du TP précédent

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Répertoire de travail

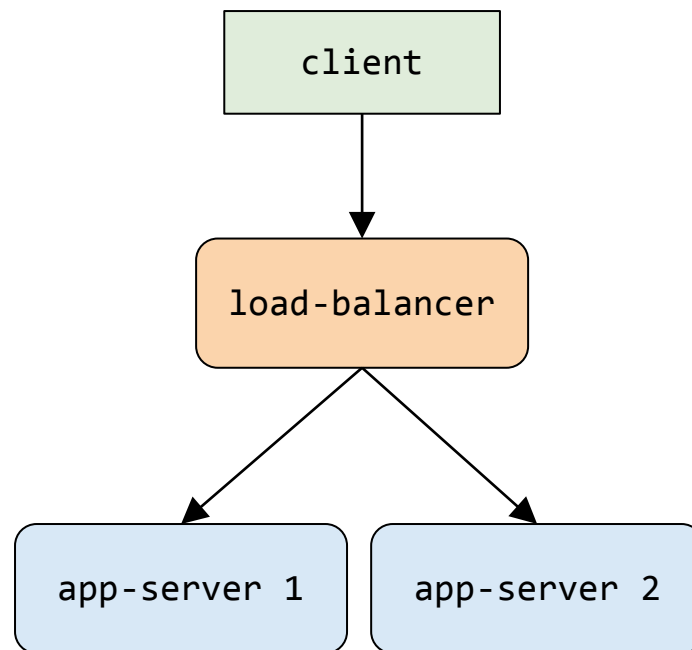
Nous allons commencer par nous assurer que nous sommes bien dans notre répertoire de travail `/home/ubuntu/tp-ansible`.

```
$ pwd
```

Vérifiez que votre répertoire courant est bien `/home/ubuntu/tp-ansible`.

Architecture de notre système

Nous sommes en train de mettre en place une architecture web qui ressemblera (à terme) à ça :



Cette architecture, assez classique, même si simplifiée permet de mettre en place de la répartition de charge sur plusieurs serveurs d'application.

Nous allons nous attaquer au composant de load-balancing. Pour les besoins de nos TP, nous allons utiliser HAProxy, qui est un logiciel libre très efficace pour ce genre de fonctions.

Nous allons continuer à enrichir le playbook du TP précédent pour mettre en œuvre HAProxy

Installation de HAProxy

L'installation de HAProxy ressemble à l'installation de Tomcat que nous avons vu précédemment. Commençons à ajouter un nouveau play :

```
# install.yml
---
- hosts: load-balancers
  gather_facts: false
  become: true
  tasks:
    - name: install HAProxy
      package:
        name: haproxy
- hosts: app-servers
```

```
gather_facts: false
become: true
tasks:
- name: install tomcat
  package:
    name: tomcat7
- name: start / enable tomcat
  service:
    name: tomcat7
    enabled: true
    state: started
- name: wait for Tomcat to be up on port 8080
  wait_for:
    port: 8080
- name: ensure home page returns 200
  uri:
    url: http://127.0.0.1:8080/
```

Remarque

Ce TP ne va modifier que le play sur le groupe load-balancers. Dans la suite du TP, seule cette section du fichier `install.yml` est décrite, mais le reste du fichier (play sur le groupe `all` et play sur le groupe `app-servers`) doivent être conservés inchangés

Le lancement d'`ansible-playbook` peut-être réalisé. Pour gagner du temps, nous allons restreindre son exécution uniquement aux machines du groupe `load-balancers` grâce à l'option `-l <subset>/--limit=<subset>` :

```
$ ansible-playbook install.yml -l load-balancers
```

Le retour de l'exécution donne ceci :

```
PLAY [load-balancers]
*****

TASK [install HAProxy]
*****
changed: [18.195.91.219]

PLAY [app-servers]
*****
skipping: no hosts matched

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

L'exécution se passe comme prévu. Notez que puisque l'on a réduit le champ d'application du playbook à une machine, on découvre qu'un play a été passé ([skipping](#)) car aucune machine du groupe utilisé n'appartient au critère (groupe) pour lancer ce play.

Configuration de HAProxy

Pour démarrer le service, il va être nécessaire de modifier des fichiers de configuration :

- /etc/default/haproxy, pour armer le service
- /etc/haproxy/haproxy.cfg pour paramétrer le démon avec une configuration applicative qui implémente l'architecture décrite précédemment.

Armer le service

Nous allons simplement procéder à un changement d'une ligne du fichier /etc/default/haproxy en utilisant le module `lineinfile`.

Le fichier contient initialement le contenu suivant :

```
# Set ENABLED to 1 if you want the init script to start haproxy.
ENABLED=0
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

Nous allons le modifier pour qu'il contienne :

```
# Set ENABLED to 1 if you want the init script to start haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

Le play des load-balancers devient alors :

```
- hosts: load-balancers
  become: true
  tasks:
    - name: install HAProxy
      package:
        name: haproxy
    - name: allow service to start
      lineinfile:
        dest: /etc/default/haproxy
        regexp: ENABLED=.*
        line: ENABLED=1
```

Le lancement d'ansible-playbook (toujours limité au groupe load-balancers) doit procéder à un changement au premier lancement et pas au second.

Vérifiez que le fichier est bien modifié via ansible :

```
ubuntu@ip-172-31-20-24:~/tp-ansible$ ansible load-balancers -a \
"cat /etc/default/haproxy"
52.48.56.113 | SUCCESS | rc=0 >>
# Set ENABLED to 1 if you want the init script to start haproxy.
ENABLED=1
# Add extra flags here.
#EXTRAOPTS="-de -m 16"
```

Si vous avez fait une bêtise, connectez-vous en ssh sur la VM pour remettre le fichier dans son état initial avant de refaire une tentative.

Nous allons à présent pouvoir armer le service haproxy :

```
- hosts: load-balancers
  gather_facts: false
  become: true
  tasks:
    - name: install HAProxy
      package:
        name: haproxy
    - name: allow service to start
      lineinfile:
        dest: /etc/default/haproxy
        regexp: ENABLED=.*
        line: ENABLED=1
    - name: start / enable HAProxy
      service:
        name: haproxy
        enabled: true
        state: started
```

Le lancement du playbook se termine correctement.

Configuration applicative d'HAProxy

Nous allons utiliser le module `template` pour créer un fichier de configuration de HAProxy. Créez un nouveau fichier dans le répertoire courant (`/home/ubuntu/tp-ansible`) du nom de `haproxy.cfg.j2`. Le contenu à y insérer est le suivant :

```
global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    user haproxy
    group haproxy
    daemon
    stats socket /var/run/haproxy.sock

defaults
    log          global
    mode          http
    option        httplog
    option        dontlognull
    option                redispatch
    retries                3
    timeout http-request   1s
    timeout http-keep-alive 1s
    timeout check         1s
    timeout client        50s
    timeout server        50s
    timeout connect       1s
    maxconn               3000

listen stats
    bind 0.0.0.0:9000
    stats enable
    stats uri /

frontend my_frontend
    bind *:80
    default_backend be_app_servers

backend be_app_servers
    balance roundrobin
    option httpchk
{% for server in groups['app-servers'] %}
    server      vm_{{ server|replace('.', '_') }} {{ server }}:8080 check
{% endfor %}
```

Tips

Pour éviter des copier-coller depuis le fichier PDF, nous avons pré-ajouté ce fichier dans le dossier /home/ubuntu/haproxy.cfg.j2

Nous avons un fichier qui contient deux types d'utilisation de Jinja2 :

- une boucle sur les membres d'un groupes {% for server... endfor %}
- une variable à transformer : {{ server|replace('.', '_') }}, à l'intérieur de la boucle précédente

Si l'on venait à ajouter une nouvelle machine dans l'inventaire dans le groupe app-servers, un nouveau passage d'ansible-playbook aurait pour conséquence de mettre à jour le fichier produit.

Reste ensuite à tester notre template en le référençant dans le playbook :

```
- hosts: load-balancers
  gather_facts: false
  become: true
  tasks:
    - name: install HAProxy
      package:
        name: haproxy
    - name: allow service to start
      lineinfile:
        dest: /etc/default/haproxy
        regexp: ENABLED=.*
        line: ENABLED=1
    - name: Install configuration file
      template:
        src: haproxy.cfg.j2
        dest: /etc/haproxy/haproxy.cfg
    - name: start / enable HAProxy
      service:
        name: haproxy
        enabled: true
        state: started
```

Pensez bien à placer ce bout de code avant le démarrage du service HAProxy.

Si le service ne démarre toujours pas, l'erreur affichée doit pouvoir vous aider à détecter la coquille dans le template.

Astuce

Lorsque vous mettez au point des templates, utilisez systématiquement l'option `--diff` au lancement d'ansible-playbook. Vous verrez ainsi passer les modifications du fichier sous forme de diff UNIX dans la sortie de l'exécution

Une fois que le service est démarré, vous pouvez faire des tests en consultant deux URLs dans le navigateur. Notez bien l'adresse IP de notre load-balancer (52.48.56.113 dans cet exemple).

Page de statistiques de HAProxy, qui montre la configuration du service, les frontends, les backends et le trafic :

<http://52.48.56.113:9000>

← → ↻ ⓘ 18.195.91.219:9000 ☆ 🔒 📄 📱 📧 ⌵ ⋮

HAProxy version 1.6.3, released 2015/12/25

Statistics Report for pid 18120

> General process information

pid = 18120 (process #1, nbproc = 1)
 uptime = 0d 0h03m07s
 system limits: memmax = unlimited; ulimit-n = 4034
 maxsock = 4034; maxconn = 2000; maxpipes = 0
 current conns = 2; current pipes = 0/0; conn rate = 2/sec
 Running tasks: 1/9; idle = 100 %

active UP
 active UP, going down
 active DOWN, going up
 active or backup DOWN
 active or backup DOWN for maintenance (MAINT)
 active or backup SOFT STOPPED for maintenance

backup UP
 backup UP, going down
 backup DOWN, going up
 not checked
 active or backup DOWN for maintenance (MAINT)

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
 • Scope :
 • Hide 'DOWN' servers
 • Refresh now
 • CSV export

External resources:
 • [Primary site](#)
 • [Updates \(v1.5\)](#)
 • [Online manual](#)

stats		Queue		Session rate		Sessions				Bytes		Denied		Errors		Warnings		Server														
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend					2	2	-	2	2	3 000	2			0	0	0	0	0					OPEN									
Backend	0	0			0	0		0	0	300	0	0	0s	0	0	0	0	0	0	0	0	0	3m7s UP		0	0	0			0		

my_frontend		Queue		Session rate		Sessions				Bytes		Denied		Errors		Warnings		Server														
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend					0	0	-	0	0	3 000	0			0	0	0	0	0	0	0	0	0	OPEN									

be_app_servers		Queue		Session rate		Sessions				Bytes		Denied		Errors		Warnings		Server													
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
vm_18_194_210_238	0	0	-	0	0	0	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	0	3m7s UP	L7OK/200 in 2ms	1	Y	-	0	0	0s	-
vm_18_194_135_36	0	0	-	0	0	0	0	0	0	0	-	0	0	?	0	0	0	0	0	0	0	3m7s UP	L7OK/200 in 1ms	1	Y	-	0	0	0s	-	
Backend	0	0			0	0		0	0	300	0	0	?	0	0	0	0	0	0	0	0	0	3m7s UP		2	2	0		0	0s	

Le service (frontend) load-balancé :

<http://52.48.56.113>

Statistics Report for H... x Apache Tomcat x +

← → ⓘ 52.48.56.113 🔒 📄 📱 📧 ⌵ ⋮

It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat7/webapps/ROOT/index.html`

Tomcat7 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat7` and `CATALINA_BASE` in `/var/lib/tomcat7`, following the rules from `/usr/share/doc/tomcat7-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat7-docs: This package installs a web application that allows to browse the Tomcat 7 documentation locally. Once installed, you can access it by clicking [here](#).

tomcat7-examples: This package installs a web application that allows to access the Tomcat 7 Servlet and JSP examples. Once installed, you can access it by clicking [here](#).

À chaque fois que vous rechargez cette dernière URL, vous devriez voir des compteurs augmenter dans la pages de statistiques :

be_app_servers																									
	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings						
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk			
vm_52_50_186_151	0	0	-	0	1		0	1	-	5	5	2 053	10 738		0		0	0	0	0	2m8s UP	L7OK/200 in 3ms			
vm_52_30_177_148	0	0	-	0	1		1	1	-	5	5	3 357	9 107		0		0	0	0	0	2m8s UP	L7OK/200 in 2ms			
Backend	0	0		0	2		1	2	0	10	10	5 410	19 845	0	0		0	0	0	0	2m8s UP				

Ajout des tests

Ajoutez les tests sur HAProxy : celui-ci doit répondre sur les port 80 et 9000. Le port 9000 doit en outre répondre en HTTP/200

```
- hosts: load-balancers
gather_facts: false
become: true
tasks:
- name: install HAProxy
  package:
    name: haproxy
- name: allow service to start
  lineinfile:
    dest: /etc/default/haproxy
    regexp: ENABLED=.*
    line: ENABLED=1
- name: Install configuration file
  template:
    src: haproxy.cfg.j2
    dest: /etc/haproxy/haproxy.cfg
- name: start / enable HAProxy
  service:
    name: haproxy
    enabled: true
    state: started
- name: wait for HAProxy to be up on port 9000
  wait_for:
    port: 9000
- name: wait for HAProxy to be up on port 80
  wait_for:
    port: 80
- name: ensure stat page returns 200
  uri:
    url: http://127.0.0.1:9000/
```

Relancez ansible-playbook, vérifiez que vous obtenez bien la sortie suivante :

```
PLAY *****

TASK [setup] *****
ok: [52.48.56.113]

TASK [install http lib2 required by uri module] *****
ok: [52.48.56.113]

PLAY *****

TASK [install HAProxy] *****
ok: [52.48.56.113]

TASK [allow service to start] *****
```

```

ok: [52.48.56.113]

TASK [Install configuration file] *****
ok: [52.48.56.113]

TASK [start / enable HAProxy] *****
ok: [52.48.56.113]

TASK [wait for HAProxy to be up on port 9000] *****
ok: [52.48.56.113]

TASK [wait for HAProxy to be up on port 80] *****
ok: [52.48.56.113]

TASK [ensure stat page returns 200] *****
ok: [52.48.56.113]

PLAY *****
skipping: no hosts matched

PLAY RECAP *****
52.48.56.113      : ok=9    changed=0    unreachable=0    failed=0
  
```

Question 4.1

- Pourquoi la mise en place d'un test HTTP avec le module `uri` sur le port 80 n'est a priori pas une bonne idée ?

Réponse 4.1

-

Question Bonus pour départager les plus rapides

Question 4.2

- Sauriez-vous modifier le code ci-dessus pour utiliser les `@IP` internes et les noms des machines (hostname) à la place des données de l'inventaire ?
- Pourquoi faut-il passer ansible-playbook sur toutes les machines pour cela ?

L'objectif étant d'obtenir :

Avant :

```

backend be_app_servers
    balance roundrobin
    option httpchk
  
```

```
server      vm_52_50_186_151 52.50.186.151:8080 check
server      vm-52_30_177_148 52.30.177.148:8080 check
```

Après:

```
backend be_app_servers
  balance roundrobin
  option httpchk
  server      vm-ip-172-31-23-39 172.31.23.39:8080 check
  server      vm-ip-172-31-23-40 172.31.23.40:8080 check
```

Réponse 4.2

-
-

Commit Git

Il est temps d'enregistrer votre travail dans Git.

Voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .
$ git commit -m "fin tp4"
$ git tag tp4
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp4
```

TP #5 - Modularisation du code

Objectifs du TP

- Réorganiser notre code pour utiliser des rôles
- Ajouter une gestion «propre» des handlers

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle et aux VMs cibles
- Vous êtes familier de Linux, des commandes shells
- Vous disposez du code fonctionnel produit lors du TP précédent

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Répertoire de travail

Nous allons commencer par nous assurer que nous sommes bien dans notre répertoire de travail `/home/ubuntu/tp-ansible`.

```
$ pwd
```

Vérifiez que votre répertoire courant est bien `/home/ubuntu/tp-ansible`.

Organisation des rôles

Nous allons réorganiser le code présent dans `install.yml` sous forme de 3 rôles :

- Un rôle commun qui contiendra les éléments communs à toutes les machines
- Un rôle haproxy
- un rôle tomcat

Pour ce faire, nous allons créer l'arborescence suivante :

```
roles
+- common
|   +- meta
|   |   +- main.yml
|   |
|   +- tasks
|   |   +- main.yml
|   |
+- haproxy
|   +- handlers
|   |   +- main.yml
|   |
|   +- meta
|   |   +- main.yml
|   |
|   +- tasks
|   |   +- main.yml
|   |   +- tests.yml
|   |
|   +- templates
|   |   +- haproxy.cfg.j2
|   |
+- tomcat
|   +- meta
|   |   +- main.yml
|   |
|   +- tasks
|   |   +- main.yml
|   |   +- tests.yml
```

Le refactoring consiste donc à n'utiliser plus que des rôles dans le playbook qui doit ressembler à ça :

```
# install.yml
---
- hosts: all
  gather_facts: true
  become: true
  roles:
    - role: common
      tags: common

- hosts: load-balancers
  become: true
  roles:
    - role: haproxy
```

```
tags: haproxy

- hosts: app-servers
  gather_facts: false
  become: true
  roles:
    - role: tomcat
      tags: tomcat
```

Le playbook a bien maigri et c'est une bonne nouvelle, la complexité a été déplacée dans les rôles.

Le travail de refactoring est un travail classique de développement qui arrive très régulièrement au cours de la vie du code Ansible. N'ayez pas peur de casser le code pour le rendre meilleur. Le commit Git de l'état précédent est votre garde-fou pour revenir à un état fonctionnel. Les tests sont également là pour vous assurez que vous n'avez pas cassé tout votre travail.

Écriture des rôles

Nous allons réorganiser le code en suivant les principes suivants :

- Les tests (`wait_for`, `uri`) doivent être déplacés dans les fichiers `tests.yml` dans les répertoires `tasks` des rôles correspondant. Leur inclusion dans le fichier `main.yml` doit se faire en fin de fichier, et doit être taggé avec le label `tests`.
- Laissez pour le moment les fichiers `meta/main.yml` (dans tous les rôles) et `handlers/main.yml` (dans le rôle `haproxy`) vides
- Travaillez rôle par rôle et lancez souvent Ansible pour savoir si vous avez cassé quelque chose

```
roles
+- common
|   +- meta
|   |   +- main.yml -> (vide)
|   |
|   +- tasks
|       +- main.yml -> (récupérer une partie de install.yml)
|
+- haproxy
|   +- handlers
|   |   +- main.yml -> (vide)
|   |
|   +- meta
|   |   +- main.yml -> (vide)
|   |
|   +- tasks
|       +- main.yml -> (récupérer une partie de install.yml)
|       +- tests.yml -> (récupérer une partie de install.yml)
```



```
| |
| +- templates
|   +- haproxy.cfg.j2 -> déplacer haproxy.cfg.j2 ici
+- tomcat
  +- meta
  |   +- main.yml -> (vide)
  |
  +- tasks
    +- main.yml -> (récupérer une partie de install.yml)
    +- tests.yml -> (récupérer une partie de install.yml)
```

Refactoring de Tomcat

Commencez par refactorer la partie Tomcat comme amuse-bouche

Avant :

```
# install.yml
---
# ... Les partie précédentes du code ont été masquées
- hosts: app-servers
  gather_facts: false
  become: true
  tasks:
    - name: install tomcat
      package:
        name: tomcat7
    - name: start / enable tomcat
      service:
        name: tomcat7
        enabled: true
        state: started
    - name: wait for Tomcat to be up on port 8080
      wait_for:
        port: 8080
    - name: ensure home page returns 200
      uri:
        url: http://127.0.0.1:8080/
```

Après:

```
# install.yml
---
# ... Les partie précédentes du code ont été masquées
- hosts: app-servers
```

```
gather_facts: false
become: true
roles:
- role: tomcat
  tags: tomcat
```

```
# roles/tomcat/tasks/main.yml
---
- name: install tomcat
  package:
    name: tomcat7

- name: start / enable tomcat
  service:
    name: tomcat7
    enabled: true
    state: started

- name: include tests
  include_tasks: tests.yml
  tags: tests
```

```
# roles/tomcat/tasks/tests.yml
---

- name: wait for Tomcat to be up on port 8080
  wait_for:
    port: 8080

- name: ensure home page returns 200
  uri:
    url: http://127.0.0.1:8080/
```

Exemple de lancement d'ansible-playbook après refactoring de la partie Tomcat :

```
$ ansible-playbook install.yml -t tomcat
```

Produit le résultat suivant :

```
PLAY [app-servers]
*****

TASK [tomcat : install tomcat]
*****
ok: [18.194.210.238]
ok: [18.194.135.36]
```

```

TASK [tomcat : start / enable tomcat]
*****
ok: [18.194.210.238]
ok: [18.194.135.36]

TASK [tomcat : include tests]
*****
included: /home/ubuntu/tp-ansible/roles/tomcat/tasks/tests.yml for
18.194.210.238, 18.194.135.36

TASK [tomcat : wait for Tomcat to be up on port 8080]
*****
ok: [18.194.210.238]
ok: [18.194.135.36]

TASK [tomcat : ensure home page returns 200]
*****
ok: [18.194.210.238]
ok: [18.194.135.36]

PLAY RECAP
*****
18.194.135.36      : ok=5    changed=0    unreachable=0    failed=0
18.194.210.238    : ok=5    changed=0    unreachable=0    failed=0
  
```

Vous noterez que les tâches exécutées dans le rôle tomcat sont toutes préfixées par tomcat:.

Refactoring des deux autres rôles

Faites de mêmes pour le rôle haproxy. N'oubliez pas de déplacer le template haproxy.cfg.j2 dans le répertoire roles/haproxy/templates/, sans quoi vous allez faire connaissance avec de nouveaux messages d'erreur d'Ansible.

Quand vous aurez fini ce travail, le répertoire tp-ansible ne doit plus contenir que le playbook install.yml et les répertoires inventories et roles.

Une exécution d'ansible-playbook complète (sans limitation avec l'option -l ou -t) doit ressembler à ceci :

```

PLAY [load-balancers]
*****

TASK [Gathering Facts]
*****
ok: [18.195.91.219]

TASK [haproxy : install HAProxy]
*****
ok: [18.195.91.219]

TASK [haproxy : allow service to start]
  
```

```
*****
ok: [18.195.91.219]

TASK [haproxy : Install configuration file]
*****
ok: [18.195.91.219]

TASK [haproxy : start / enable HAProxy]
*****
ok: [18.195.91.219]

TASK [haproxy : Include tests for haproxy]
*****
included: /home/ubuntu/tp-ansible/roles/haproxy/tasks/tests.yml for
18.195.91.219

TASK [haproxy : wait for HAProxy to be up on port 9000]
*****
ok: [18.195.91.219]

TASK [haproxy : wait for HAProxy to be up on port 80]
*****
ok: [18.195.91.219]

TASK [haproxy : ensure stat page returns 200]
*****
ok: [18.195.91.219]

PLAY [app-servers]
*****

TASK [tomcat : install tomcat]
*****
ok: [18.194.210.238]
ok: [18.194.135.36]

TASK [tomcat : start / enable tomcat]
*****
ok: [18.194.135.36]
ok: [18.194.210.238]

TASK [tomcat : include tests]
*****
included: /home/ubuntu/tp-ansible/roles/tomcat/tasks/tests.yml for
18.194.210.238, 18.194.135.36

TASK [tomcat : wait for Tomcat to be up on port 8080]
*****
ok: [18.194.210.238]
ok: [18.194.135.36]

TASK [tomcat : ensure home page returns 200]
*****
ok: [18.194.135.36]
ok: [18.194.210.238]

PLAY RECAP
*****
```

```

18.194.135.36      : ok=5    changed=0    unreachable=0    failed=0
18.194.210.238    : ok=5    changed=0    unreachable=0    failed=0
18.195.91.219     : ok=9    changed=0    unreachable=0    failed=0
  
```

Note

Jusqu'à Ansible 2.1, la notification d'inclusion d'un fichier de tâches (en bleu clair dans l'exemple précédent) était affichée, mais ce comportement a changé dans la version 2.2. Pas de panique si vous ne voyez pas cette ligne dans la sortie ! Il a été rajouté en version 2.4

Ajout des handlers/notify pour HAProxy

Il nous reste un travail à réaliser pour être content de notre travail, faire en sorte que des modifications du template haproxy.cfg.j2 donne lieu au redémarrage du service haproxy.

Ça tombe bien, nous avons justement un changement à faire dans ledit fichier de configuration. En effet, la ligne :

```
stats socket /var/run/haproxy.sock
```

Doit devenir :

```
stats socket /var/run/haproxy.sock level admin
```

Pour faire ce changement, nous allons au préalable créer un handler qui va recharger le service haproxy en remplissant le fichier roles/haproxy/handlers/main.yml comme suit :

```

# roles/haproxy/handlers/main.yml
---

- name: reload HAProxy service
  service:
    name: haproxy
    state: reloaded
  
```

Attention !!

Jusqu'à maintenant le nom (name:) des tâches n'avaient pas de réelle importance. Dans le cas d'un handler, c'est la clé qui permet de faire le lien entre les notify et les handlers. Prenez garde à vérifier la casse notamment.

Une fois le handler écrit, nous pouvons mettre en place le notify :

```
# roles/haproxy/tasks/main.yml
---
- name: install HAProxy
  package:
    name: haproxy

- name: allow service to start
  lineinfile:
    dest: /etc/default/haproxy
    regexp: ENABLED=.*
    line: ENABLED=1

- name: Install configuration file
  template:
    src: haproxy.cfg.j2
    dest: /etc/haproxy/haproxy.cfg
  notify:
    - reload HAProxy service

- name: start / enable HAProxy
  service:
    name: haproxy
    enabled: true
    state: started

- name: include tests
  include: tests.yml
  tags: tests
```

Il ne reste plus qu'à faire la modification du fichier de template `roles/haproxy/templates/haproxy.cfg.j2` pour refléter le changement attendu et lancer `ansible-playbook` taggé sur `haproxy` pour gagner du temps :

```
$ ansible-playbook install.yml -t haproxy --diff
```

N'oubliez pas l'option `--diff` qui vous montrera précisément le changement effectué :

```
PLAY [all] *****

TASK [setup] *****
ok: [52.18.70.52]
ok: [52.18.41.52]
ok: [52.50.169.220]

PLAY [load-balancers] *****

TASK [haproxy : install HAProxy] *****
```

```

ok: [52.18.41.52]

TASK [haproxy : allow service to start] *****
ok: [52.18.41.52]

TASK [haproxy : Install configuration file] *****
changed: [52.18.41.52]
--- before: /etc/haproxy/haproxy.cfg
+++ after: dynamically generated
@@ -1,18 +1,18 @@
global
    log /dev/log      local0
    log /dev/log      local1 notice
    chroot /var/lib/haproxy
    user haproxy
    group haproxy
    daemon
-   stats socket /var/run/haproxy.sock
+   stats socket /var/run/haproxy.sock level admin

defaults
    log      global
    mode     http
    option   httplog
    option   dontlognull
    option                               redispatch
    retries                               3
    timeout  http-request 1s
    timeout  http-keep-alive 1s

TASK [haproxy : start / enable HAProxy] *****
ok: [52.18.41.52]

TASK [haproxy : include tests] *****
included: /home/ubuntu/tp-ansible/roles/haproxy/tasks/tests.yml for 52.18.41.52

TASK [haproxy : wait for HAProxy to be up on port 9000] *****
ok: [52.18.41.52]

TASK [haproxy : wait for HAProxy to be up on port 80] *****
ok: [52.18.41.52]

TASK [haproxy : ensure stat page returns 200] *****
ok: [52.18.41.52]

RUNNING HANDLER [haproxy : reload HAProxy service] *****
changed: [52.18.41.52]

PLAY [app-servers] *****

TASK [tomcat : include tests] *****
included: /home/ubuntu/tp-ansible/roles/tomcat/tasks/tests.yml for 52.18.70.52,
52.50.169.220

PLAY RECAP *****
52.18.41.52      : ok=10    changed=2    unreachable=0    failed=0
52.18.70.52     : ok=2     changed=0    unreachable=0    failed=0

```

```
52.50.169.220 : ok=2 changed=0 unreachable=0 failed=0
```

Question 5.1

- Si on relance ansible-playbook à l'identique, est-ce que la sortie est identique ?
- Si non, quels sont les changements ?

Réponse 5.1

-
-

Pour vous convaincre de l'utilité des tags, relancer ansible-playbook avec le tag tests :

```
$ ansible-playbook install.yml -t tests
```

Vous venez de trouver une commande simple et rapide pour vérifier que votre plateforme a ses composants de base en état de marche !

Commit Git

Il est temps d'enregistrer votre travail dans Git.

Voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .  
$ git commit -m "fin tp5"  
$ git tag tp5
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp5
```


TP #6 - Déploiement d'une application

Objectifs du TP

- Déployer une application avec Ansible.
- Redéployer l'application dans une nouvelle version.

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants :

- Vous avez validé votre accès à votre VM de contrôle et aux VMs cibles.
- Vous êtes familier de Linux, des commandes shells.
- Vous disposez du code fonctionnel produit lors du TP précédent.

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Répertoire de travail

Nous allons commencer par nous assurer que nous sommes bien dans notre répertoire de travail `/home/ubuntu/tp-ansible`.

```
$ pwd
```

Vérifiez que votre répertoire courant est bien `/home/ubuntu/tp-ansible`.

Plan de bataille

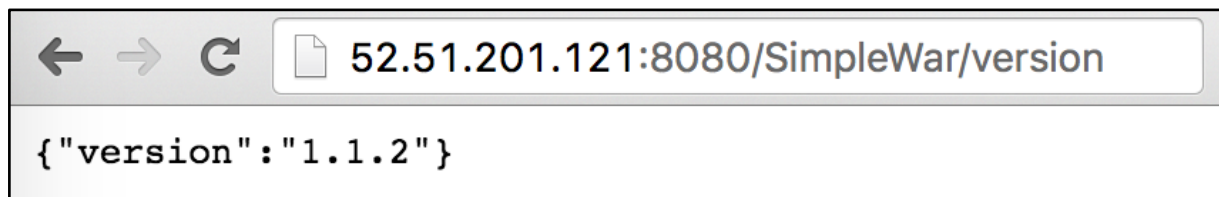
Nous nous sommes jusqu'à présent contentés d'installer et configurer un HAProxy et des Tomcats vides, il est grand temps de déployer une véritable application pour utiliser notre plateforme.

Pour les besoins de la formation, nous avons compilé une petite application java que nous allons installer à l'aide d'Ansible. Vous la trouverez sur <https://s3-eu-west-1.amazonaws.com/octo-formation-ansible/SimpleWar/SimpleWar-1.1.2.war>. Les versions 1.1.3 et 1.1.4 sont également disponibles.

Cette application affiche une simple phrase sur sa page d'accueil :



Elle renvoie également sa version sur l'URL `/version`, nous nous en servons plus loin dans ce TP :



Nous allons créer le rôle nécessaire au déploiement de cette application dans Tomcat, puis l'améliorer pour gérer le déploiement d'un fichier de configuration et la mise à jour de l'application.

Nous n'allons pas rentrer dans la configuration détaillée de Tomcat, nous y passerions des heures. Quelques notions suffiront pour déployer notre application :

- Les applications doivent être déposées dans `/var/lib/tomcat7/webapps`.
- Les fichiers de configuration doivent être déposés dans `/usr/share/tomcat7/lib`.
- Tomcat va exposer l'application sur `http://<ip de votre machine>:8080/SimpleWar`.

Ajout du rôle tomcat-app

Dans la continuité du TP#5, nous allons mettre notre code de déploiement dans un rôle dédié. Commencez-donc par ajouter le rôle tomcat-app dans votre arborescence :

```
roles
+- common
+- haproxy
+- tomcat
+- tomcat-app
    + defaults
    |   +- main.yml
    + handlers
    |   +- main.yml
    +- tasks
        +- main.yml
        +- deploy.yml
```

Notre premier déploiement va se contenter du minimum :

- Arrêt de Tomcat.
- Suppression de l'application existante.
- Téléchargement du SimpleWar dans le dossier Tomcat.
- Démarrage de Tomcat.

Avant de s'attaquer au code, préparons le terrain en déclarant quelques variables. Ajoutez le contenu suivant à tomcat-app/defaults/main.yml :

```
# roles/tomcat-app/defaults/main.yml
---

tomcat_app_repository: https://s3-eu-west-1.amazonaws.com/octo-formation-
ansible/SimpleWar
tomcat_app_name: SimpleWar
```

(Attention, il n'y a pas de retour à la ligne entre tomcat_app_repository et l'URL, mais la mise en page n'aime pas les lignes trop longues !)

Ces variables seront disponibles partout ailleurs dans le rôle puisque nous les avons déclarées dans defaults/main.yml. Il sera également facile de les surcharger lors de l'appel du rôle, ce qui est très utile pour réutiliser un rôle dans différents environnements !

Remarque

Prenez toujours le temps de réfléchir à ce qui peut être mis en variable dans le rôle. Les variables rendent le code plus lisible, et permettent d'adapter le rôle à son environnement.

Grâce aux variables déclarées plus haut, vous pourrez :

- Déployer d'autres applications que SimpleWar, simplement en changeant le nom.
- Modifier l'URL du dépôt pour chaque environnement, pour utiliser un dépôt différent en production, par exemple.

Ceci sans modifier le rôle, uniquement l'inventaire ou les `group_vars` !

Mais revenons au code. Remplissez maintenant le fichier `tasks/main.yml`, qui ne fait pour l'instant qu'inclure `deploy.yml` :

```
# roles/tomcat-app/tasks/main.yml
---
- include_tasks: deploy.yml
```

Puis écrivez le fichier `tasks/deploy.yml` :

```
# roles/tomcat-app/tasks/deploy.yml
---
- name: Stop tomcat
  service:
    name: tomcat7
    state: stopped

- name: Delete existing webapp
  file:
    name: /var/lib/tomcat7/webapps/{{ tomcat_app_name }}
    state: absent
  notify: Restart tomcat

- name: Get application war
  get_url:
    url: "{{ tomcat_app_repository }}/{{ tomcat_app_name }}-{{ tomcat_app_version }}.war"
    dest: /var/lib/tomcat7/webapps/{{ tomcat_app_name }}.war
    force: yes
  notify: Restart tomcat
```

Ce fichier implémente la procédure de déploiement expliquée plus haut. Il utilise le module `get_url` d'Ansible pour télécharger l'application et la placer à l'endroit attendu par Tomcat, et appelle un handler de redémarrage de Tomcat.

Question 6.1

- Cherchez la documentation du module `get_url` sur le site d'Ansible. Pourquoi avons-nous dû ajouter le paramètre `force` : yes ?

Réponse 6.1

-

Notez l'usage des variables déclarées précédemment : le fait d'avoir déclaré `tomcat_app_name` nous évite d'écrire "SimpleWar" partout dans les tasks.

Votre grande sagacité vous aura fait remarquer que le code utilise une variable qui n'est pas encore déclarée : `tomcat_app_version`. Nous fournirons cette variable lors de l'appel à Ansible, ce qui nous permettra de changer la version déployée à chaque appel à Ansible.

Le rôle utilise un handler pour redémarrer Tomcat, n'oubliez pas de le déclarer :

```
# roles/tomcat-app/handlers/main.yml
---

- name: Restart tomcat
  service:
    name: tomcat7
    state: restarted
```

Ajout du playbook de déploiement

Vous allez maintenant créer le playbook de déploiement qui va appeler le rôle tomcat-app. A la racine de votre projet, ajoutez le fichier `deploy.yml` avec le contenu suivant :

```
# deploy.yml
---

- hosts: all
  tasks:
    - name: check mandatory attributes
      fail:
        msg: "Missing arg: tomcat_app_version"
        when: tomcat_app_version is not defined
        run_once: true
        tags: [ always ]

- hosts: app-servers
  gather_facts: yes
  become: yes
  roles:
    - role: tomcat-app
      tags: [ app, tomcat, deploy ]
```

La deuxième partie du fichier ne devrait pas vous surprendre : nous appliquons le rôle tomcat-app sur tous les app-servers.

La première partie en revanche est nouvelle. Nous vérifions que la variable `tomcat_app_version` est définie avant de déployer. Souvenez-vous, cette variable est utilisée dans notre rôle, qui va échouer si nous oublions de la donner à Ansible. Le module `fail` est ici très utile pour expliquer à l'opérateur qui lance le playbook que cette variable est obligatoire.

Notez l'ajout de `tags: [always]`, afin de s'assurer que la vérification soit effectuée quels que soient les filtres utilisés lors de l'exécution d'Ansible.

Question 6.2

- Pourquoi vérifie-t-on cette variable dès le début du playbook ?
- Si le rôle tomcat-app est exécuté sans cette variable, dans quel état va-t-il laisser nos serveurs ?

Réponse 6.2

-

•

Lancement du playbook de déploiement

Nous sommes enfin prêts à déployer ! Lancez donc votre playbook `deploy.yml` :

```
$ ansible-playbook deploy.yml

PLAY [all]
*****

TASK [Gathering Facts]
*****
ok: [18.194.210.238]
ok: [18.195.91.219]
ok: [18.194.135.36]

TASK [check mandatory attributes]
*****
fatal: [18.195.91.219]: FAILED! => {"changed": false, "failed": true, "msg":
"Missing arg: tomcat_app_version"}

NO MORE HOSTS LEFT
*****
to retry, use: --limit @/home/ubuntu/tp-ansible/deploy.retry

PLAY RECAP
*****
*****
18.194.135.36      : ok=1    changed=0    unreachable=0    failed=0
18.194.210.238    : ok=1    changed=0    unreachable=0    failed=0
18.195.91.219     : ok=1    changed=0    unreachable=0    failed=1
```

Votre déploiement échoue : le module `fail` a bien détecté l'absence de la variable `tomcat_app_version`. Relancez le déploiement en ajoutant cette variable (avec la première version disponible, 1.1.2) :

```
$ ansible-playbook deploy.yml -e tomcat_app_version=1.1.2
```

```
PLAY [all] *****

TASK [setup] *****
ok: [52.49.207.114]
ok: [52.51.201.121]
ok: [52.50.101.154]

TASK [check mandatory attributes] *****
skipping: [52.51.201.121]

PLAY [app-servers] *****

TASK [setup] *****
ok: [52.50.101.154]
ok: [52.49.207.114]

TASK [tomcat-app : include_tasks]
*****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/deploy.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Stop tomcat] *****
changed: [52.50.101.154]
changed: [52.49.207.114]

TASK [tomcat-app : Delete existing webapp] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : Get application war] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

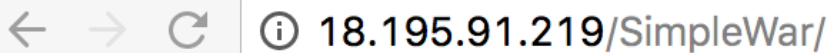
TASK [tomcat-app : Start tomcat] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

TASK [tomcat-app : include] *****

PLAY RECAP *****
52.49.207.114      : ok=7 changed=3    unreachable=0failed=0
52.50.101.154      : ok=7 changed=3    unreachable=0failed=0
52.51.201.121      : ok=1 changed=0    unreachable=0failed=0
```

Vérifiez maintenant le bon déploiement de l'application en ouvrant sa page d'accueil dans votre navigateur (remplacez par l'IP de votre load-balancer) :

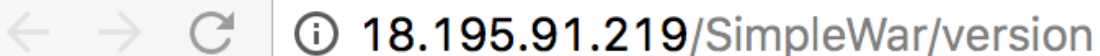
<http://18.195.91.219/SimpleWar/>

18.195.91.219/SimpleWar/

Cette application tourne sur l'environnement de DEV !

Vérifiez également la version déployée :

<http://18.195.91.219/SimpleWar/version>

18.195.91.219/SimpleWar/version

```
{"version": "1.1.2"}
```

Des tests, toujours des tests

Le module `uri` va une nouvelle fois nous être utile pour écrire des tests. SimpleWar renvoie sa version courante au format JSON sur l'URL `/version`, nous allons nous en servir.

Commencez par ajouter un fichier `tasks/tests.yml` à votre rôle :

```
# roles/tomcat-app/tasks/tests.yml
---

- name: Get running application version
  uri:
    url: http://localhost:8080/{{ tomcat_app_name }}/version
    return_content: yes
    timeout: 180
    register: tomcat_app_version_check_result

- name: Verify running version matches deployed version
  fail:
    msg: "Running version does not match deployed version"
    when: tomcat_app_version_check_result.json.version !=
tomcat_app_version
```

Nous avons ajouté le paramètre `return_content: yes`, qui va stocker le résultat de l'appel dans la variable `tomcat_app_version_check_result`.

La deuxième tâche va lever une erreur si la version renvoyée ne correspond pas à la version déployée par Ansible (`tomcat_app_version`).

Question 6.3

- Une fois de plus, la documentation va vous être utile. Le module `uri` a converti automatiquement le contenu de la page en structure JSON, ce qui nous permet de vérifier très facilement la version.
Sous quelle condition le module fait-il cette conversion ?
- Bonus : sans cette facilité, comment feriez-vous pour extraire la version depuis la page ?
Indice : Ansible fournit des filtres de conversion, voir... La documentation !

Réponse 6.3

-
-

Pour finir, rajoutez l'include nécessaire dans `tasks/main.yml` :

```
# roles/tomcat-app/tasks/main.yml
---
- include_tasks: deploy.yml

- name: Make sure tomcat is started
  meta: flush_handlers

- include_tasks: tests.yml
```

Nous introduisons au passage le module `meta: flush_handlers`. C'est un module spécial qui va obliger Ansible à exécuter les handlers en attente avant de passer à la suite de l'exécution.

Son usage est obligatoire ici : sans ça, Tomcat ne sera pas redémarré avant l'exécution des tests, qui échoueront systématiquement.

Comme toujours, pensez à tester vos ajouts en relançant le playbook. Vous devriez obtenir un résultat proche de ceci :

```
PLAY [all] *****

TASK [setup] *****
ok: [52.51.201.121]
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [check mandatory attributes] *****
skipping: [52.51.201.121]

PLAY [app-servers] *****

TASK [setup] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : include_tasks] *****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/deploy.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Stop tomcat] *****
changed: [52.50.101.154]
changed: [52.49.207.114]

TASK [tomcat-app : Delete existing webapp] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

TASK [tomcat-app : Get application war] *****
ok: [52.50.101.154]
ok: [52.49.207.114]

RUNNING HANDLER [tomcat-app : Restart tomcat] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

TASK [tomcat-app : include_tasks] *****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/tests.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Get running application version] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : Verify running version matches deployed version] *****
skipping: [52.49.207.114]
skipping: [52.50.101.154]

PLAY RECAP *****
52.49.207.114      : ok=9 changed=3    unreachable=0failed=0
52.50.101.154      : ok=9 changed=3    unreachable=0failed=0
52.51.201.121      : ok=1 changed=0    unreachable=0failed=0
```

Ajout d'un fichier de configuration

Pour l'instant, votre application dit qu'elle tourne sur l'environnement de "An Sible", mais ce n'est pas votre nom... Il est temps de la personnaliser.

Pour ce faire, vous allez déployer un fichier `app.properties` dans `/usr/share/tomcat7/lib/`, qui va définir la variable `env_name` pour votre application.

Commencez par créer un nouveau template dans votre rôle :

```
# roles/tomcat-app/templates/app.properties.j2

env_name={{ tomcat_app_env_name }}
```

Puis donnez une valeur par défaut à la variable `tomcat_app_env_name` :

```
# roles/tomcat-app/defaults/main.yml
---

tomcat_app_repository: https://s3-eu-west-1.amazonaws.com/octo-formation-
ansible/SimpleWar
tomcat_app_name: SimpleWar
tomcat_app_env_name: VOTRE_NOM
```

Enfin, créez le fichier `tasks/configure.yml` qui va déployer le fichier de configuration, et incluez-le dans `tasks/main.yml` :

```
# roles/tomcat-app/tasks/configure.yml
---

- name: Deploy configuration
  template:
    src: app.properties.j2
    dest: /usr/share/tomcat7/lib/app.properties
  notify: Restart tomcat
```

```
# roles/tomcat-app/tasks/main.yml
---

- include_tasks: deploy.yml
- include_tasks: configure.yml

- name: Make sure tomcat is started
```

```
meta: flush_handlers  
  
- include: tests.yml
```

Puis relancez un déploiement. La page d'accueil de SimpleWar devrait à présent afficher votre nom !

Ne déployer que quand il faut

Nous allons finir ce TP en améliorant le rôle pour le rendre idempotent.

Lancez le playbook `deploy.yml` deux fois de suite : vous remarquerez que notre rôle arrête Tomcat et redéploie l'application à chaque fois, même si rien n'a changé. Sur une application réelle, en production, ceci va générer des interruptions de service inutiles.

Heureusement, il nous est facile d'empêcher cela : il nous suffit de vérifier la version déployée avant de déployer. Et vous savez déjà le faire, puisque c'est exactement le test que vous avez écrit un peu plus tôt !

Adaptons-le donc pour cet usage. Créez un nouveau fichier `tasks/check-deploy-needed.yml` :

```
# roles/tomcat-app/tasks/check-deploy-needed.yml  
---  
  
- name: Get currently running application version  
  uri:  
    url: http://localhost:8080/{{ tomcat_app_name }}/version  
    return_content: yes  
    timeout: 60  
    failed_when: false  
    register: tomcat_app_predeploy_version  
  
- name: Compute whether deployment is needed  
  set_fact:  
    tomcat_app_needs_deploy: tomcat_app_predeploy_version.status  
    != 200 or (tomcat_app_predeploy_version.json.version != '{{  
tomcat_app_version }}')
```

Nous stockons le résultat de notre vérification dans le fact `tomcat_app_needs_deploy` pour pouvoir le réutiliser plus facilement par la suite.

Il ne nous reste qu'à nous en servir pour conditionner le déploiement :

```
# roles/tomcat-app/tasks/main.yml
---

- include_tasks: check-deploy-needed.yml

- include_tasks: deploy.yml
  when: tomcat_app_needs_deploy

- include_tasks: configure.yml

- name: Make sure tomcat is started
  meta: flush_handlers

- include_tasks: tests.yml
```

Question 6.4

- Pourquoi n'avons nous pas également mis la clause when sur l'include de configure.yml ?

Réponse 6.4

-

Lancez deploy.yml deux fois de suite avec la même version : la deuxième exécution ne devrait afficher que de belles couleurs verte et bleue, et ne rien changer sur votre environnement.

```
$ ansible-playbook deploy.yml -e tomcat_app_version=1.1.2

[...]

PLAY RECAP *****
52.49.207.114      : ok=14   changed=4 unreachable=0 failed=0
52.50.101.154     : ok=14   changed=4 unreachable=0 failed=0
52.51.201.121     : ok=1    changed=0 unreachable=0 failed=0

$ ansible-playbook deploy.yml -e tomcat_app_version=1.1.2

PLAY [all] *****

TASK [setup] *****
ok: [52.51.201.121]
ok: [52.49.207.114]
ok: [52.50.101.154]
```

```

TASK [check mandatory attributes] *****
skipping: [52.51.201.121]

PLAY [app-servers] *****

TASK [setup] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : include_tasks]
*****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/check-deploy-needed.yml
for 52.49.207.114, 52.50.101.154

TASK [tomcat-app : Get currently running application version] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : Compute whether deployment is needed] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : include_tasks]
*****
skipping: [52.49.207.114]
skipping: [52.50.101.154]

TASK [tomcat-app : include_tasks]
*****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/configure.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Deploy configuration] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : include_tasks]
*****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/tests.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Get running application version] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : Verify running version matches deployed version] *****
skipping: [52.49.207.114]
skipping: [52.50.101.154]

PLAY RECAP *****
52.49.207.114      : ok=9 changed=0    unreachable=0failed=0
52.50.101.154      : ok=9 changed=0    unreachable=0failed=0
52.51.201.121      : ok=1 changed=0    unreachable=0failed=0

```

Déployez à présent une autre version :

```
$ ansible-playbook deploy.yml -e tomcat_app_version=1.1.3

PLAY [all] *****

TASK [setup] *****
ok: [52.51.201.121]
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [check mandatory attributes] *****
skipping: [52.51.201.121]

PLAY [app-servers] *****

TASK [setup] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : include] *****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/check-deploy-needed.yml
for 52.49.207.114, 52.50.101.154

TASK [tomcat-app : Get currently running application version] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : Compute whether deployment is needed] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : include] *****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/deploy.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Stop tomcat] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

TASK [tomcat-app : Delete existing webapp] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

TASK [tomcat-app : Get application war] *****
changed: [52.49.207.114]
changed: [52.50.101.154]

TASK [tomcat-app : include] *****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/configure.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Deploy configuration] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

RUNNING HANDLER [tomcat-app : Restart tomcat] *****
changed: [52.49.207.114]
changed: [52.50.101.154]
```



```
TASK [tomcat-app : include] *****
included: /home/ubuntu/tp-ansible/roles/tomcat-app/tasks/tests.yml for
52.49.207.114, 52.50.101.154

TASK [tomcat-app : Get running application version] *****
ok: [52.49.207.114]
ok: [52.50.101.154]

TASK [tomcat-app : Verify running version matches deployed version] *****
skipping: [52.49.207.114]
skipping: [52.50.101.154]

PLAY RECAP *****
52.49.207.114      : ok=14    changed=4 unreachable=0 failed=0
52.50.101.154      : ok=14    changed=4 unreachable=0 failed=0
52.51.201.121      : ok=1     changed=0 unreachable=0 failed=0
```

Le rôle détecte que la version a changé, et lance bien le déploiement cette fois.

Question 6.5 (bonus)

- Il est courant sur des applications réelles de vouloir lancer un redéploiement même si la version n'a pas changé, en cas de crash ou pour recharger une configuration par exemple.

Un ajout de variable bien placé nous permettrait de forcer un déploiement dans notre rôle, en fournissant cette nouvelle variable à l'appel d'Ansible.

Voyez-vous comment faire cela ?

Réponse 6.5

-

Commit Git

Il est temps d'enregistrer votre travail dans Git.

Voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .  
$ git commit -m "fin tp6"  
$ git tag tp6
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp6
```

TP #7 - Déploiement “Blue-Green”

Objectifs du TP

- Redéployer à chaud l’application du TP précédent

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle et aux VMs cibles
- Vous êtes familier de Linux, des commandes shells
- Vous disposez du code fonctionnel produit lors du TP précédent.

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Répertoire de travail

Nous allons commencer par nous assurer que nous sommes bien dans notre répertoire de travail `/home/ubuntu/tp-ansible`.

```
$ pwd
```

Vérifiez que votre répertoire courant est bien `/home/ubuntu/tp-ansible`.

Plan de bataille

Nous avons mis en place le déploiement de notre application Tomcat avec une gestion intelligente des versions afin d’éviter les déploiements inutiles.

Cependant, lorsqu’il a lieu, le déploiement provoque une interruption du service car les deux Tomcat sont coupés simultanément.

Pour palier ce problème nous allons faire que ce déploiement ait lieu en deux temps afin de toujours avoir une application disponible.

Nous allons ensuite raffiner ce processus de déploiement afin de rendre explicite la mise en maintenance et la sortie de maintenance des applications. Ce qui permettra de s'assurer du bon fonctionnement de chaque application avant sa sortie de maintenance.

Pour effectuer cette gestion explicite de la maintenance nous allons exploiter la socket d'administration Haproxy avec un module Ansible prévu à cet effet.

Ajout de la directive serial

Dans le playbook `deploy.yml` nous allons ajouter la directive `serial` au play de déploiement afin d'effectuer ce déploiement en deux passes.

```
# deploy.yml

- hosts: all
  tasks:
    - name: check mandatory attributes
      fail:
        msg: "Missing arg: tomcat_app_version"
        when: tomcat_app_version is not defined
        run_once: true
        always_run: yes
        tags: [ always ]

- hosts: app-servers
  gather_facts: yes
  become: yes
  serial: 50%
  roles:
    - role: tomcat-app
      tags: [ app, tomcat, deploy ]
```

Nous pouvons alors lancer le déploiement avec une version différente de l'application :

```
ansible-playbook deploy.yml -e 'tomcat_app_version=1.1.4'
```

Et observer le résultat dans l'interface de statistique HAProxy.

be_app_servers																						
	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status	LastChk		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp			Retr	Redis
vm_52_17_219_69	0	0	-	0	0	1	0	0	2	-	15	13	25 646	13 494	0	0	0	0	2	1	2s DOWN	L4CON in 0ms
vm_52_31_25_5	0	0	-	0	0	1	0	0	1	-	13	11	37 621	14 447	0	0	1	0	2	1	34s UP	L7OK/200 in 2ms
Backend	0	0	-	0	0	2	0	0	2	0	22	24	63 267	27 941	0	0	1	0	4	2	1h37m UP	

be_app_servers																						
	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status	LastChk		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp			Retr	Redis
vm_52_17_219_69	0	0	-	0	1	-	0	2	-	15	13	25 646	13 494	0	0	0	1	0	2	1	2m20s UP	L7OK/200 in 2ms
vm_52_31_25_5	0	0	-	0	1	-	0	1	-	13	11	37 621	14 447	0	0	1	1	0	2	1	0s DOWN	L4CON in 0ms
Backend	0	0	-	0	2	-	0	2	0	22	24	63 267	27 941	0	0	1	1	0	4	2	1h40m UP	

Question 7.1

- Que se serait-il passé si nous avions eu 3 serveurs Tomcat ?
- Comment s'assurer de faire le déploiement en deux passes maximum ?

Réponse 7.1

-
-

Gestion active du pool de serveurs

Dans les premières secondes avant qu'il ne le marque down HAProxy pourrait laisser revenir des requêtes en erreur. Heureusement c'est un logiciel bien conçu qui en cas d'absence de réponse d'un backend tente d'en joindre un autre.

Mais comment réagirait HAProxy si un backend non fonctionnel était démarré suite à une mise à jour ?

À partir du moment où le Tomcat est de nouveau démarré et répond bien sur l'URL que vérifie HAProxy il est remis dans le pool actif. HAProxy est capable de gérer un backend qui ne répond pas, mais ne gère pas ceux qui répondent mal.

C'est à nous de nous assurer que nous ne remettons pas un Tomcat non fonctionnel dans le pool actif.

Il nous faut tester les réponses de ce Tomcat après son démarrage mais avant de le laisser rejoindre le pool actif.

Nous allons donc utiliser `delegate_to` et le module `haproxy` d'Ansible pour manipuler les backends actifs dans HAProxy.

```
# roles/tomcat-app/tasks/disable-in-haproxy.yml

- name: remove backend from haproxy
  haproxy:
    state: disabled
    host: "vm_{{ inventory_hostname|replace('.', '_') }}"
    backend: be_app_servers
    wait: true
    delegate_to: "{{ item }}"
    with_items: "{{ groups['load-balancers']|default([]) }}"
```

```
# roles/tomcat-app/tasks/enable-in-haproxy.yml
```

```
- name: add backend to haproxy
  haproxy:
    state: enabled
    host: "vm_{{ inventory_hostname|replace('.', '_') }}"
    backend: be_app_servers
    wait: true
  delegate_to: "{{ item }}"
  with_items: "{{ groups['load-balancers']|default([]) }}"
```

Il ne reste plus qu'à inclure ces fichiers aux bons endroits.

```
# roles/tomcat-app/tasks/deploy.yml

- include_tasks: disable-in-haproxy.yml

- name: Stop tomcat
  service:
    name: tomcat7
    state: stopped

- name: Delete existing webapp
  file:
    name: /var/lib/tomcat7/webapps/{{ tomcat_app_name }}
    state: absent
  notify: Restart tomcat

- name: Get application war
  get_url:
    url: "{{ tomcat_app_repository }}/{{ tomcat_app_name }}-{{ tomcat_app_version }}.war"
    dest: /var/lib/tomcat7/webapps/{{ tomcat_app_name }}.war
    force: yes
  notify: Restart tomcat
```

```
# roles/tomcat-app/tasks/configure.yml

- name: Deploy configuration
  template:
    src: app.properties.j2
    dest: /usr/share/tomcat7/lib/app.properties
  register: configuration_result
  notify: Restart tomcat

- include_tasks: disable-in-haproxy.yml
  when: configuration_result | changed
```

```
# roles/tomcat-app/tasks/main.yml

- include_tasks: check-deploy-needed.yml

- include_tasks: deploy.yml
  when: tomcat_app_needs_deploy

- include_tasks: configure.yml

- name: Make sure tomcat is started
  meta: flush_handlers

- include_tasks: tests.yml

- include_tasks: enable-in-haproxy.yml
```

Vous pouvez maintenant retester en relançant le playbook de déploiement.

Commit Git

Il est temps d'enregistrer votre travail dans Git.

Voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .
$ git commit -m "fin tp7"
$ git tag tp7
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp7
```

TP #8 - Tests ServerSpec

Objectifs du TP

- Mettre un harnais de test sur le code de déploiement de votre application avec "ServerSpec"

Prérequis

Avant de commencer ce TP, vous devez avoir satisfait les prérequis suivants

- Vous avez validé votre accès à votre VM de contrôle et aux VMs cibles
- Vous êtes familier de Linux, des commandes shells

Niveau de difficulté : Débutant

Connexion à la machine

Suivez la procédure de la première section du document pour vous connecter à votre VM.

Répertoire de travail

Nous allons commencer par nous placer dans un nouveau répertoire de travail afin de ne pas impacter le travail précédent : /home/ubuntu/.

```
$ cd /home/ubuntu/
```

Vérifiez que votre répertoire courant est bien /home/ubuntu/.

Vous pouvez ensuite extraire l'archive récupérée dans ce dossier.

```
$ unzip /home/ubuntu/tp8.zip
$ cd tp8
$ ls -a
```

Vous devriez voir un dossier tp-ansible et un dossier .git.

Plan de bataille

L'archive que vous avez récupérée contient une succession de solutions des TP 2 à 8.

```
$ git tag
tp2
```



```
tp3  
tp4  
tp5  
tp6  
tp7  
tp8
```

Pour pouvoir utiliser ServerSpec, il faut que ruby soit installé sur votre machine. En effet, ServerSpec est une gem (bibliothèque) ruby. Vous pouvez taper la commande suivante pour valider que ruby a bien été installé.

```
$ ruby --version  
ruby 2.4.2p198 (2017-09-14 revision 59899) [x86_64-linux-gnu]
```

Vous devriez voir la version de ruby installée sur la machine ($\geq 2.4.2p198$).

Pour pouvoir facilement installer plusieurs bibliothèques ruby, il est possible d'utiliser "[bundler](https://bundler.io)". C'est un gestionnaire de "gem" qui s'intègre avec le site de "gem" ruby principal : <https://rubygems.org/>. Pour vérifier que bundler a bien été installé, vous pouvez taper la commande suivante :

```
$ bundle --version
```

Vous devriez voir la version de bundle installée sur la machine ($\geq 1.16.0$).

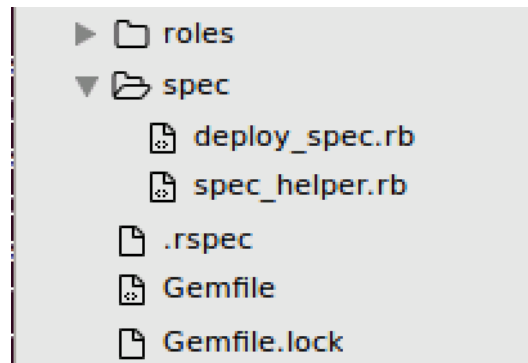
Vous pouvez maintenant vous positionner dans le répertoire projet et récupérer les gems qui vont vous être nécessaires.

```
$ cd tp-ansible  
$ bundle install
```

Il faut par ailleurs exporter les deux variables d'environnement suivantes :

```
$ export USER=ubuntu  
$ export USER_SSH_KEY_PATH=~/.ssh/id_rsa
```

Maintenant que vous avez les pré-requis pour faire du "ServerSpec", vous pouvez initialiser votre suite de tests. Pour cela, créez un fichier "deploy_spec.rb" dans le répertoire "spec".



Et en-tête du fichier, précisez :

- les dépendances requises à savoir “spec_helper.rb”
- la machine cible sur laquelle vous allez effectuer des assertions à savoir le Load Balancer (LB)

```
require_relative 'spec_helper.rb'  
on_linux_host '<IP LB>'
```

Enfin, pour lancer la suite, il vous faudra taper la commande suivante :

```
$ rspec spec/deploy_spec.rb
```

Il ne vous reste maintenant plus qu’à écrire les tests qui permettent d’arriver à ce résultat :

```
Service httpd  
  in LB  
    should be up and running  
      Service "haproxy"  
        should be running  
    should listen on port 80  
      Port "80"  
        should be listening  
    Command "curl http://localhost/SimpleWar/version"  
      stdout  
        should match /{"version":"\d.\d.\d"}/  
  
Finished in 0.66436 seconds (files took 0.28312 seconds to load)  
3 examples, 0 failures
```

BONNE CHANCE !

Question 8.1

- Quelle est l’utilité du fichier “GemFile” ?

- Quelle est l'utilité du fichier ".rspec" ?
- Où les variables "USER" et "USER_SSH_KEY_PATH" sont-elles utilisées ?
- Sur quelle machine sont exécutées les différentes assertions ?

Réponse 8.1

-
-
-
-

Commit Git

Il est temps d'enregistrer votre travail dans Git.

Voici les commandes à lancer pour enregistrer vos changements :

```
$ git add .  
$ git commit -m "fin tp8"  
$ git tag tp8
```

Plus tard, et si avez besoin, vous pourrez à tout moment revenir à cet état du code Ansible en tapant :

```
$ git checkout tp8
```