## Évaluation finale à caractère synthèse

Informations

**Évaluation** : 30 % de la session

**Type de travail** : en équipe de 3 et individuel

**Durée** : 2,5 heures/personne

**Système d’exploitation** : Ubuntu serveur

**Environnement** : virtuel

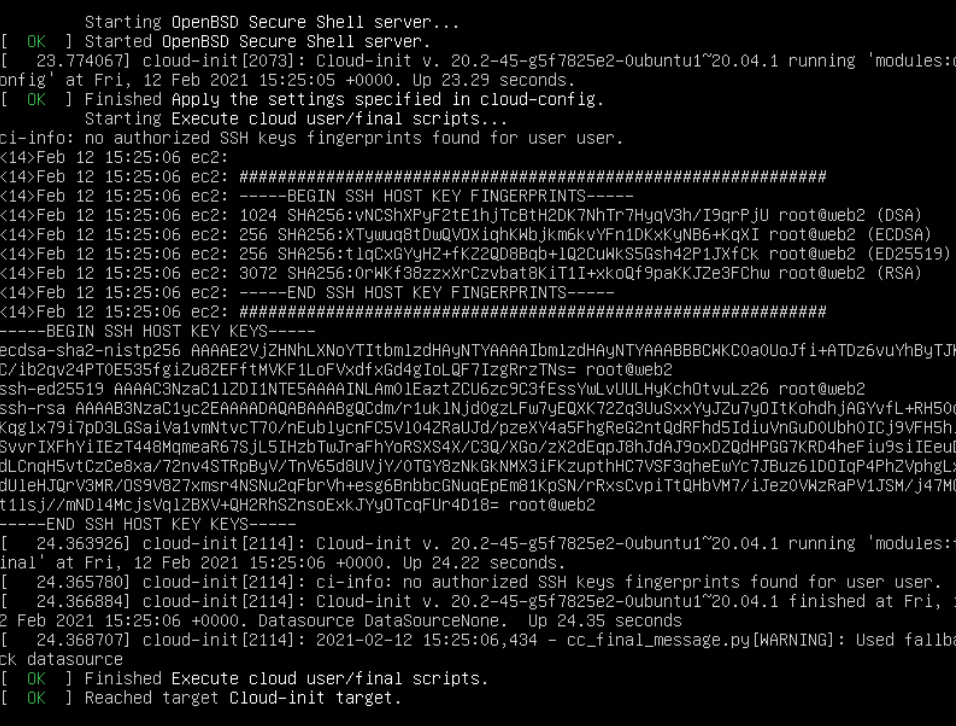
Objectifs

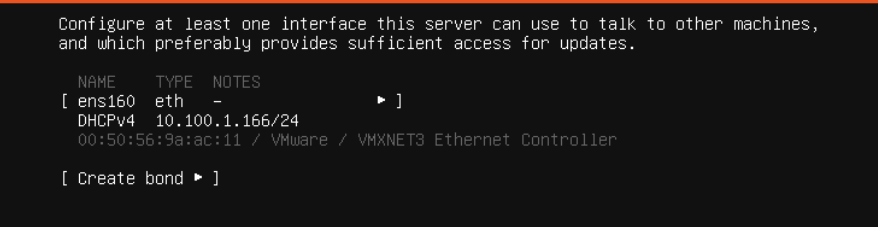
Cette évaluation a pour objectifs :

* Vérifier l’autonomie de la part de l’étudiant / étudiante qui doit chercher à résoudre les problèmes techniques par lui-même.
* Vérifier l’adaptabilité aux différents environnements et technologies.
* Vérifier les compétences du cours.

Dans cette évaluation finale à caractère synthèse, vous allez utiliser vos notions apprises dans ce cours pour installer des serveurs avec un outil d’automatisation, ansible.

Vous allez travailler en équipe de trois pour avoir 3 serveurs à configurer. Par contre, la majorité du travail se fera individuellement (les parties en équipe sont indiquées dans le travail) et vous devez remettre chacun un document personnel. Même si le travail se fait de manière individuelle, il vous est permis de solliciter l’aide des autres membres de l’équipe pour vous aider à accomplir vos tâches.





#### Partie 1 : Installation des serveurs et préparation d’Ansible

Dans cette partie, chaque membre de l’équipe va installer un serveur Ubuntu selon les spécifications demandées.

Étape 1 : installation des serveurs (en équipe)

Chaque membre de l’équipe doit installer un serveur Ubuntu, avec **seulement le serveur ssh d’installer**.

Voici les spécifications des serveurs :

* + - * Créer un sous-répertoire sous A2020\_4361\_InstServeurs\_CR ayant les initiales de chacun des membres de l’équipe : par exemple R\_L\_K. Installer les VMs des serveurs dans ce sous-répertoire.
      * 2 CPU
      * Mémoire : 4 Go
      * Espace disque : 5 Go
      * Installateur : ubuntu-20.04-live-server-amd64.iso
      * Quand demandé, utiliser le nouvel installateur.
      * Créer seulement une partition / avec tous l’espace disque (pas de swap).
      * Noms des serveurs :
        + web1\_1995717🡪 Keven,
        + web2\_1521797 🡪 Léakim
        + lb1\_1995544 🡪 Raoul
      * Utiliser le même utilisateur et le même mot de passe.
        + user
        + Soleil123!

Connectez-vous sur chacun des serveurs et vérifiez les adresses IPs. Vous aurez à vérifier les adresses IPs à chaque fois que vous allez lancer vos serveurs et modifier vos fichiers de configuration en conséquence.

Vous devez préparer vos serveurs pour la communication avec Ansible.

Vous allez copier vos clés SSH sur chacun des serveurs. Vérifiez que vous avez accès à vos serveurs en SSH.

Prenez un snapshot de chacune vos VMs serveurs. Chaque membre de l’équipe va faire le reste de l’EFCS individuellement et à la fin du travail, vous devrez revenir à ce snapshot. (juste 1)

//Dans Web2

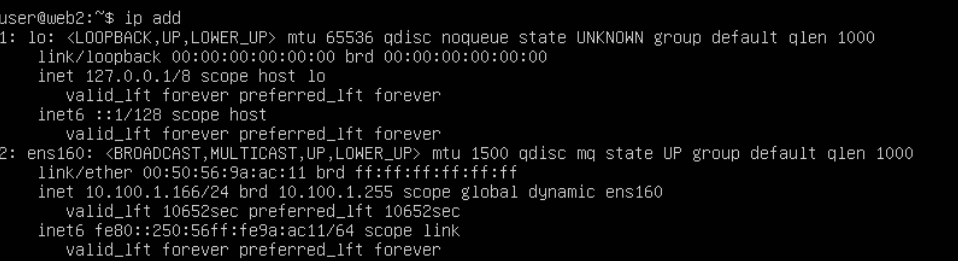
ssh-keygen -t ed25519 -C [“serveur web2”](mailto:1521797@csfoy.ca)

//Dans client

ssh-copy-id -i ~/.ssh/id\_ed25519 [user@10.100.1.157](mailto:user@10.100.1.157)

ssh-copy-id -i ~/.ssh/id\_ed25519 user@10.100.1.166

ssh-copy-id -i ~/.ssh/id\_ed25519 user@10.100.1.62



Étape 2 : Configuration d’Ansible pour vos serveurs (partie individuelle dans nos VMs personnelles)

Dans un nouveau répertoire sous le répertoire **ansible**, vous devez créer un fichier **hosts** qui inclut une section **[webservers]** et une section **[loadbalancers]**.

La section **[webservers]** va inclure les adresses IP des serveurs web1 et web2. La section **[loadbalancers]** va inclure l’adresse du dernier serveur lb1. Ajoutez également des alias à vos serveurs. Votre fichier **hosts** devrait ressembler à ça :

[webservers]

web1 ansible\_host=192.168.210.51

web2 ansible\_host=192.168.210.52

web1 ansible\_host=192.168.210.51

web2 ansible\_host=10.100.1.166

[loadbalancers]

lb1 ansible\_host=192.168.210.53

[local]

control ansible\_connection=local // Pour faire des commandes dans nos vm

La section **[local]** définit que notre contrôleur Ansible est local et nous permet d’automatiser des actions sur notre machine locale.

Vous devez également vous créer un fichier **ansible.cfg** qui indique quel fichier **hosts** utiliser. Comme on utilise le même utilisateur sur les trois serveurs, vous pouvez définir l’utilisateur dans la partie **[defaults]** avec le paramètre **remote\_user**. (il faut tout le monde avoir le même nomUtilisateur pour nous simplifier la tâche)

remote\_user = *VotreUtilisateur*

Étape 3 : Vérification

Vous allez vérifier votre configuration et la communication avec vos serveurs.

Exécutez les commandes suivantes.

ansible --list-hosts all

ansible --list-hosts webservers:loadbalancers

//à changer selon l’hôte

ansible --list-hosts web1

// Ping pour tester

ansible -m ping all

ansible -m shell -a "uname" webservers:loadbalancers

# La commande suivante a le même résultat que la précédente.

# Inclure le résultat d’une des deux commandes.

// le !local 🡪 on veut tout sauf local en excluant

ansible -m shell -a "uname" \!local

**Remarque** : l’état **changed** est normal, même si nous n’avons pas fait de changement. Ça indique que nous avons utilisé le shell.

**Question 1 :** vous devez inclure le résultat de chacune vos commandes. (screen shot)

Étape 4 : Mise à jour des serveurs

Nous allons automatiser la mise à jour de nos serveurs. Créez le playbook **apt\_update.yml** (fichier yaml) suivant :

# apt\_update.yml

---

- hosts: webservers:loadblancers

become: true

tasks:

- name: UPDATE PACKAGES

apt:

name: '\*'

state: latest

Lancez votre playbook avec ansible-playbook.

**Question 2 :** la mise à jour a-t-elle fonctionné ? (devrait pas fonctionner car on est become mais on a pas le mot de passe sudo)

Comme dans le TP, nous avons un problème avec le mot de passe sudo. On pourrait encore le mettre dans notre fichier hosts, mais il existe une meilleure solution (il est également possible de se créer une voûte chiffrer avec ansible, mais c’est en dehors de ce cours).

#### Partie 2 : Mot de passe sudo (tout le monde a le mot de passe sudo avec un bon firewall :D)

Nous pouvons configurer l'utilisateur distant que nous utilisons pour pouvoir exécuter toutes ou certaines commandes en utilisant sudo même sans fournir de mot de passe. Dans ce cas, nous devons protéger le compte utilisateur de la machine gestionnaire sur laquelle sa clé publique ssh est installée sur le serveur distant. Quiconque peut accéder à cette machine pourra contrôler les serveurs distants.

Qui peut exécuter la commande sudo, quelles sont ces commandes et si un mot de passe est requis est contrôlé dans le fichier /etc/sudoers. Il peut être modifié manuellement à l'aide de la commande visudo ou nous pouvons demander à Ansible de le modifier. Vous pouvez également modifier le fichier avec n'importe quel éditeur, mais si vous enregistrez une version mal formatée, vous pouvez facilement verrouiller l'utilisateur root. Par conséquent, il est fortement recommandé d'utiliser la commande visudo qui validera la syntaxe du fichier avant de l'enregistrer.

Étape 1 : Changer les sudoers manuellement

Établir une connexion ssh avec le serveur **web1**.

Sur ce serveur, lancez visudo.

sudo visudo

Nous voulons changer dans ce fichier :

# Allow members of group sudo to execute any command

%sudo ALL=(ALL:ALL) ALL

pour

# Allow members of group sudo to execute any command

%sudo ALL=(ALL:ALL) **NOPASSWD:** ALL

Pour vérifier que le tout fonctionne, vous devez fermer votre session ssh, en rétablir une nouvelle (ça permet de lire les modifications au fichier /etc/suduers) et d’essayer d’exécuter une commande demandant sudo : par exemple sudo grep root /etc/shadow. Le système ne devrait pas vous demander de mot de passe.

Vous pouvez maintenant essayer de nouveau le playbook pour mettre à jour ce serveur (ajouter le paramètre --limit à votre commande pour limiter la commande à seulement l’hôte modifier, ansible-playbook -v apt-update.yml --limit "web1").

**Question 3 :** la mise à jour a-t-elle fonctionné ?

Étape 2 : Changer les sudoers avec un playbook ansible

Créez un playbook **set\_sudoers.yml**.

# set\_sudoers.yml

---

- hosts: webservers:loadbalancers

tasks:

- lineinfile:

path: /etc/sudoers

state: present

regexp: '^%sudo'

line: '%sudo ALL=(ALL:ALL) NOPASSWD: ALL'

validate: 'visudo -cf %s'

Vous allez exécuter ce playbook avec cette commande :

ansible-playbook -v set\_sudoer.yml --limit 'all:!web1' -b -K

On enlève l’hôte déjà modifié.

Le -b nous permet de devenir root pour cette opération.

Le -K car sudo nécessite toujours un mot de passe aux hôtes non modifiés, donc vous devrez entrée le mot de passe de l’utilisateur des serveurs.

**Question 4 :** inclure la réponse de votre commande pour démontrer que le changement a fonctionné.

Maintenant, utiliser le playbook créé précédemment pour mettre à jour vos serveurs.

**Question 5 :** la mise à jour a-t-elle fonctionné ?

#### Partie 3 : Construire notre système

Dans cette partie, vous allez construire un système pour avoir 2 serveurs Web et un répartiteur de charge (loadbalancer). Vous aurez une topologie comme suit :

|--->web1

|Serveur Web 1

lb1-----------|

Load balancer |

|--->web2

Serveur Web 2

Voici l’ordre de travail :

* 1. Gestion des packages

Installation d’apache pour webservers

Installation de php pour webservers

Installation de nginx pour loadbalancers

* 1. Configurer l’application

Télécharger un fichier index.php

* 1. Configurer le répartiteur de charges

Étape 1 : Gestion des packages

Vous allez installer nginx sur les loadbalancers et Apache et PHP sur les webservers .

Consulter le TP4 pour l’installation d’apache, php et nginx, mais ne pas installer le module rewrite (ne pas mettre) pour apache.

Vous devez utiliser un fichier nommé **install\_services.yml**.

Faire une tâche pour loadbalancers avec nginx et une tâche pour webservers avec apache et php.

Pour s’assurer que le service web est actif au démarrage, ajouter aux deux tâches la commande suivante (ajuster pour nginx des loadbalancers) :

- name: Apache demarre au demarrage

service: name=apache2 state=started enabled=yes

Ou si vous utilisez la nomenclature yaml :

- name: Apache demarre au demarrage

service:

name: apache2

state: started

enabled: yes

Pour webservers, comme vous installez deux services, il est préférable d’utiliser la nomenclature yaml :

- hosts: webservers

become: true

tasks:

- name: INSTALLATION SERVEURS WEB

apt:

name:

- apache2

- php

state: present

- name: Apache demarre au demarrage

service:

name: apache2

state: started

enabled: yes

**Question 6 :** inclure la réponse de votre commande pour démontrer que le changement a fonctionné.

Étape 2 : Configurer l’application

Nous allons installer notre application.

Dans notre cas, l’application sera un simple fichier **index.php** que nous allons copier sur les deux serveurs Web.

Créer un fichier **index.php**.

<?php

echo "<h1>Mon application Web - *Votre nom*</h1>

?>

Utilisez un playbook nommer **setup\_app.yml**.

Les hôtes seront webservers.

Dans votre tâche, vous allez utiliser le module copy dont voici un exemple de son utilisation.

copy:

src: ./index.php

dest: /var/www/html/index.php

mode: 0755

Ouvrez un navigateur et accédez à vos deux serveurs Web pour vérifier que vous avez installé votre page index.php.

Malheureusement, l’ordre de recherche est **index.html** avant **index.php**. Vous devez faire un playbook (**change\_index.yml**) qui modifie le fichier **/etc/apache2/mods-available/dir.conf** sur vos serveurs Web. Vous devez faire le changement suivant.

La ligne :

DirectoryIndex index.html index.cgi index.pl **index.php** index.xhtml index.htm

devient :

DirectoryIndex **index.php** index.html index.cgi index.pl index.xhtml index.htm

Ouvrez un navigateur et accédez à vos deux serveurs Web pour vérifier que votre page index.php est chargée par défaut.

**Question 7 :** inclure une capture démontrant vos deux serveurs Web avec votre page index.php.

Étape 3 : Configurer le répartiteur de charge

Nous allons maintenant configurer le répartiteur de charge nginx.

Sous Ubuntu, nginx utilise le répertoire sites-enabled pour les sites Web actifs. Par contre, il est toujours possible d’utiliser un fichier default.conf dans le répertoire /etc/nginx/conf.d comme vous avez vu dans les exercices de Docker.

Vous allez créer un playbook **change\_nginx.yml** qui met en commentaire la ligne

include /etc/nginx/sites-enabled/\*;

dans le fichier **/etc/nginx/nginx.conf**.

Exécutez le playbook.

**Question 8 :** inclure la réponse de votre commande pour démontrer que le changement a fonctionné.

Vous allez copier un fichier **default.conf** dans le répertoire **/etc/nginx/conf.d**. Par contre, nous allons utiliser le langage **Jinja 2** pour rendre notre fichier dynamique. De cette façon, si on ajoute des hôtes à webservers, le fichier **default.conf** va automatiquement prendre en compte le changement.

Nous allons créer un fichier **lb\_config.j2** pour créer notre fichier **default.conf** à copier sur nos loadbalancers.

upstream monsite-servers {

{% for hosts in groups['webservers'] %}

server {{ hostvars[hosts]['ansible\_host'] }} max\_fails=2;

{% endfor %}

}

server {

listen 80;

location / {

proxy\_pass http://monsite-servers;

proxy\_redirect off;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Host $server\_name;

}

}

Une boucle for est définie avec **{%%}**. Ce qui est entre **{{}}** est utilisé comme template dans le fichier yaml que nous allons utiliser. La variable **hostvars[hosts]['ansible\_host']** référence l’adresse IP qu’ansible utilise pendant qu’il s’exécute. À l’exécution, ansible va remplacer cette variable par les adresses IP qu’il trouve dans le groupe webservers.

Il reste à créer le playbook **setup\_lb.yml**. Nous allons utiliser le module template.

---

- hosts: loadbalancers

become: true

tasks:

- name: CREATION DE TEMPLATE

template:

src: ./lb\_config.j2

dest: /etc/nginx/conf.d/default.conf

N’oubliez pas d’inclure dans votre playbook un handler pour relancer le service nginx.

Exécutez le playbook.

**Question 9 :** inclure la réponse de votre commande pour démontrer que le changement a fonctionné.

Ouvrez un navigateur et accéder à l’adresse du répartiteur de charge lb1, à chaque rafraîchissement vous accédez aux serveurs Web en round robin.

Étape 4 : Mettre ensemble la configuration

Il est possible de faire notre configuration complète avec un playbook qui regroupe les playbooks nécessaire à la configuration de notre infrastructure.

Nous utilisons le module import\_playbook. Donc, pour regrouper les deux premières actions nous aurions :

# all\_playbooks.yml

---

- import\_playbook: apt\_update.yml

- import\_playbook: install\_services.yml

Créez un playbook **all\_playbook.yml** qui regroupe les playbooks nécessaires à monter notre infrastructure (attention, ce ne sont pas tous les playbooks que nous avons créé qui sont nécessaire).

Exécutez le playbook (naturellement, il n’y aura pas beaucoup de changement, mais imaginer le temps économisé lorsque vous ajoutez 2 ou 3 serveurs Web).

**Question 10 :** inclure la réponse de votre commande pour démontrer que le changement a fonctionné.

Ouvrez un navigateur et accéder à l’adresse du répartiteur de charge lb1, à chaque rafraîchissement vous accédez aux serveurs Web en round robin.

**Question 11 :** inclure une capture démontrant votre page index.php dans un navigateur à l’adresse de lb1.

**Question 12 :** Vous devez remettre les fichiers de configuration : hosts, ansible.cfg, apt\_update.yml, set\_sudoers.yml, install\_services.yml, setup\_app.yml, change\_nginx.yml, lb\_config.j2, setup\_lb.yml, all\_playbook.yml.

#### Partie 4 : remettre les VMs à l’état initial (équipe)

Soyez certains d’avoir terminé l’EFCS.

Avec les membres de votre équipe, remettre la VM au snapshot initial pour permettre à un autre membre de l’équipe de faire son EFCS.

#### Partie 5 : Correction

|  |  |  |
| --- | --- | --- |
| **Item** | **Valeur (points)** | **Résultat (points)** |
| Question 1 | 2 |  |
| Question 2 | 2 |  |
| Question 3 | 2 |  |
| Question 4 | 2 |  |
| Question 5 | 2 |  |
| Question 6 | 3 |  |
| Question 7 | 3 |  |
| Question 8 | 3 |  |
| Question 9 | 3 |  |
| Question 10 | 3 |  |
| Question 11 | 2 |  |
| Question 12 | 3 |  |
| **Total** | **30** |  |

Compétences développées

|  |  |
| --- | --- |
| **00SJ** – Effectuer le déploiement de serveurs intranet. | **00SJ # 1** – Analyser le projet de déploiement.  **00SJ # 2** – Monter les serveurs intranet.  **00SJ # 3** – Installer les services intranet.  **00SJ # 5** – Participer à la mise en service des serveurs intranet. |
| **00SK** – Effectuer le déploiement de serveurs Internet. | **00SJ # 1** – Analyser le projet de déploiement.  **00SJ # 2** – Monter les serveurs Internet.  **00SJ # 3** – Installer les services Internet.  **00SJ # 5** – Participer à la mise en service des serveurs Internet. |

Références

<https://docs.ansible.com/ansible/latest/cli/ansible.html>

<https://ansible-tips-and-tricks.readthedocs.io/en/latest/ansible/commands/>

<https://docs.ansible.com/ansible/latest/collections/ansible/builtin/copy_module.html>

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_templating.html>

<https://docs.ansible.com/ansible/latest/user_guide/playbooks_handlers.html>

<https://jinja.palletsprojects.com/en/2.11.x/>

<https://code-maven.com/enable-ansible-passwordless-sudo>