Sujet TP – Linux

Table des matières

Vanessa David Avril 2021

Présentation du Contexte

Vous arrivez dans une entreprise qui doit complètement transformer son parc informatique. Le DSI a décidé de basculer l'ensemble de l'infrastructure sous Linux, et vous avez été recruté pour permettre le déploiement de cette nouvelle infrastructure.

Cette infrastructure est composée des éléments suivants :

- serveur web hébergeant le site de l'entreprise
- serveur d'intégration continue avec Jenkins, auquel les développeurs doivent se connecter en SSH
- 3 postes de développement sous Linux
- un serveur de fichier partagé en NFS qui permet de faire de la sauvegarde

Pour chacun de ces éléments d'infrastructure vous allez devoir écrire des scripts d'automatisation pour pouvoir automatiser le déploiement de ces éléments et ainsi faciliter leur redéploiement en cas de problème et aussi fournir une documentation pour expliquer l'utilisation de vos différents scripts.

Organisation

Cet exercice se fait en binôme et sera à rendre sous forme d'un dépôt Github commun au plus tard le vendredi 30 avril à 12h00 .

Le lien du dépôt github sera à mettre dans le fichier Excel de rendu qui se trouve sur Teams en face de votre binôme.

En plus de ce travail, vous devez évaluer un autre binôme, voir dans le même fichier, quel binôme doit être évaluer. Le travail d'évaluation se fait de manière individuelle, chacun devra rendre une grille d'évaluation rempli à partir de la grille vide fourni dans Teams. Pour vendredi 30 avril à 17h00 au plus tard.

Ce travail est à réaliser sur les VM initialiser à partir des vagrant file suivant :

```
# -*- mode: ruby -*-
# vi: set ft=ruby sw=2 st=2 et :

Vagrant.configure('2') do |config|
  config.vm.box = 'debian/buster64'
  # config.vm.box = "puppetlabs/debian-7.8-64-puppet"
  config.vm.box_check_update = false

  config.vm.network "public_network", ip: "192.168.0.17"

  # Mettre en place un cache pour APT
  # config.vm.synced_folder 'v-cache', '/var/cache/apt'

  config.vm.define 'server' do |machine|
    machine.vm.hostname = 'server'

    # Limiter la RAM de la VM
```

```
    machine.vm.provider 'virtualbox' do |vb|
      vb.memory = '3000'

      # UNCOMMENT FOR MORE DISKS
      disk2_vdi = 'disk2.vdi'


      # Creer les fichiers au bon format pour VBox s'ils n'existent pas
      unless File.exist?(disk2_vdi)
        vb.customize ['createhd', '--filename', disk2_vdi, '--size', 20 * 1024]
      end

      # On attache les fichiers 'disque' sur la VM
      vb.customize ['storageattach', :id, '--storagectl', 'SATA Controller',
                    '--port', 1, '--device', 0, '--type', 'hdd', '--medium',
                    disk2_vdi]
    end
  end

  # config.vm.provision 'shell', path: 'provision.sh'
end
```

Description détaillé des éléments attendus

Nous partons du principes que l'ensemble des machines est sous une distribution Debian Buster. L'ensemble des scripts seront donc à réaliser pour la distribution Debian (même si vous pouvez les rendre compatibles avec Red Hat / CentOs et détecter la distribution)

Serveur web hébergeant le site de l'entreprise

Voici les éléments attendus sur ce serveur :

- Le paquet apache2 installé
- Le dossier /var/www/html avec pour groupe www-data et pour le droit de lire / écrire pour le groupe (en plus du propriétaire) ainsi que pour tout ses sous-dossiers
- Dans ce même dossier, créer un fichier index.html avec le contenu suivant :

```
!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>Ma première page avec du style</title>
</head>

<body>

<!-- Menu de navigation du site -->
<ul class="navbar">
  <li><a href="index.html">Home page</a>
</ul>

<!-- Contenu principal -->
<h1>Ma première page avec du style</h1>

<p>Bienvenue sur ma page avec du style!

<p>Il lui manque des images, mais au moins, elle a du style. Et elle a des
liens, même s'ils ne mènent nulle part...
```

```
&hellip;

<p>Je devrais étayer, mais je ne sais comment encore.

<!-- Signer et dater la page, c'est une question de politesse! -->
<address>Fait le 22 avril 2021<br>
  par moi.</address>

</body>
</html>
```

- Installer et configurer le pare-feu (par exemple avec UFW) et ouvrir les ports 80 et 443

Pour vérifier, vérifier dans un navigateur que la page s'affiche bien

Serveur d'intégration continue avec Jenkins

- Créer une partition de type ext4 sur le disque vide .
- Installer la version stable de Jenkins et ses prérequis en suivant la documentation officielle : https://www.jenkins.io/doc/book/installing/linux
- Démarrer le service Jenkins
- Créer un utilisateur `userjob` son home devra être
- Lui donner les permissions (via le fichier sudoers) d'utiliser `apt` (et seulement apt pas l'ensemble des droits admin)
- Afficher à la fin de l'execution du script le contenu du fichier /var/jenkins_home/secrets/initialAdminPassword pour permettre de récupérer le mot de passe
- Installer un pare feu et ouvrir le port qu'utilise Jenkins ainsi que le port utilisé pour la connexion ssh
- Déployer sur ce serveur pour l'utilisateur userjob les clés publiques ssh de chacun des postes de développement

Pour vérifier accéder au jenkins via l'URL fournit à la fin de l'installation. Et vérifier si le mot de passe afficher vous permet de vous connecter.

Vérifier également depuis les postes de développement si vous arrivez bien à vous connecter en SSH sur ce serveur d'integration continue.

Postes de développement sous Linux

Sur chacun des postes effectuer les actions suivantes :

- Installer python3, python3-pip, python3-dev, git, visual studio code
- Généré une clé ssh privé et déployer la clé publique sur le serveur d'intégration continue.
- Installer vagrant
- Dans le home de l'utilisateur, cloner le dépôt suivant : https://github.com/vanessakovalsky/example-python.git

- Vérifier que la VM démarre avec un vagrant up puis connecter vous et exécuter le fichier main.py avec la commande python main.py
- Le script doit vous affiché un message « Bien joué : »
- Les développeurs ont besoin d'un utilitaire qui leur permettra d'obtenir dans le terminal le comportement d'une corbeille. Pour cela créer un script qui permettra d'éxecuter les 3 actions suivantes :
  - `RM` pour déplacer des fichiers dans la corbeille,
  - `TRASH` pour lister le contenu de la corbeille,
  - `RESTORE` pour restaurer un fichier de la corbeille vers son emplacement original.
- Il est nécessaire lors du déplacement d'un fichier dans la corbeille de pouvoir le restaurer et donc de conserver son chemin, pour cela nous proposons de créer un fichier nomdufichier.info qui contient le chemin original du fichier à déplacer dans la corbeille. En cas de restauration on utilise ce fichier .info pour savoir où l'on doit mettre le fichier à restaurer
- Ce script devra être packagé sous la forme d'un packet DEB pour pouvoir être installé plus facilement sur les différents postes . Créer le paquet et copier le sur chaque poste avant de l'installer avec DPKG

Serveur de fichier partagé en NFS qui permet de faire de la sauvegarde

- Installer un serveur NFS
- Depuis le serveur web et le serveur d'integration continue, monter un partage vers ce serveur NFS
- Puis faire tourner toutes les heures un script qui sauvegardera le dossier /var/www/html sur le serveur web et le positionnera sur le partage NFS dans un dossier nommé web et pour le serveur d'intégration continue sauvegarder le dossier /usr/local/jenkins qui contient les jobs et mettre la sauvegarde sur le partage nfs dans un dossier server_ic
- Chaque sauvegarde sera conservé 7 jours, sera au format tar.gz et aura pour nom serveur_web_jour_mois_annee.tar.gz pour le serveur web et serveur_ic_jour_mois_annee.tar.gz pour le serveur d'intégration continue

Documentation

Rédiger une documentation qui contiendra :

- comment exécuter chaque script (en détaillant la commande à lancer pour chaque script et les arguments à leur donner si nécessaire)
- Préciser pour chaque script un moyen de vérifier si l'installation est bien ok et la machine prête à être utilisée

- S’il est nécessaire d’obtenir l’adresse IP pour vérifier, expliquer comment récupérer l’adresse IP sur les VM concernées

Rendu

Le rendu se fait sous la forme d’un dépôt github contenant à minima :

- un script par serveur / type de poste (mais vous pouvez en faire plus sans problème) + un script de sauvegarde (donc au moins 5 script). Chaque script est au format shell avec l’interpreteur /bin/bash
- une documentation sous la forme d’un fichier markdown expliquant comment utiliser vos scripts pour chaque serveur / type de poste
- Vous pouvez ajouter autant de scripts / fichiers / documents que vous jugerez nécessaire pour réaliser cet exercice

Evaluation

L’évaluation se fait de deux manière :

- par un autre stagiaire selon le barême d’évaluation présenté ci-dessous (appelée évaluation entre pairs)
- par un formateur

Barème de notation

La notation est faite sur 50 points selon le barème suivant :

- 8 points par script (donc 40 points) découpé comme suit :
  - 3 points pour le fonctionnement du script : est ce qu’il fait tout ce qui est demandé et s’exécute sans erreur
  - 2 points pour la lisibilité du script : est ce qu’on comprend le script en le lisant ? Y ‘a t’il des commentaires lorsque cela est nécessaire ?
  - 3 points : bonnes pratiques : organisation du script (utilisation de fonction lorsque cela est nécessaire), tests des données en entrée, nommage des variables et des fonctions
- 10 point sur la documentation, découpé comme suit :
  - 4pts : arrive t’on a exécuté chaque script en suivant les indications de la documentation

- 4 pts : peut t’on vérifier en suivant la documentation la bonne exécution des scripts
- 2 pts sur la clarté et la lisibilité de la documentation

Notation entre pairs

Chacun s’est vu attribué un travail à évaluer.

Pour cela une grille d’évaluation reprenant les critères et barèmes ci-dessus est à votre disposition dans Teams. La colonne commentaire vous permet d’expliquer si nécessaire votre évaluation et / ou de faire des remarques.

Remplir individuellement la grille et la déposer dans Teams avec le nom de fichier suivant : evaluation_tp_linux_votrenom_n°dubinomeevalué.xls