TP1-Infra As Code-IAC – INFRASTRUCTURE SYSTEMES ET RESEAUX ECUE 3.6
vendredi 11 juin 2021
Amine ABDOUL-AZID
Kavirajan SARAVANANE
Martial SENE

Contenu des fichiers :

- Packer builder :

```
{
    "builders": [{
        "type": "virtualbox-ovf",
        "source_path": "/home/etudiant/debian-stretch.ova",
        "ssh_username": "root",
        "ssh_password": "vitrygtr",
        "vm_name": "fw",
        "vboxmanage": "************",
        "vboxmanage_post": "************"
    }
  ]
}
```

- Démo pour afficher l'utilisateur et une adresse IP via un provisionner :

```
{
  "type": "shell",
  "inline": ["whoami && ip a"]
}
```

- Construire une VM via Packer :
  - Démo avec fw.json
    - `packer validate fw/fw.json`
      `packer build fw/fw.json`

- Démo pour assigner adresse IP via un provisionner :

```
{
    "type": "file",
    "source": "/home/etudiant/app/fw/etc/network/interfaces",
    "destination": "/etc/network/interfaces"
}
```

- Démo pour Renommer la VM
  - 1ère étape : Édit avec hostnamectl
    ```
    {
        "type": "shell",
        "inline": ["hostnamectl set-hostname fw"]
    }
    ```
  - 2ème étape : Remplacement du fichier /etc/hosts
    ```
    {
        "type": "file",
        "source": "/home/etudiant/repo/fw/etc/hosts",
        "destination": "/etc/hosts"
    }
    ```
  - 3 ème étape : reboot
    ```
    {
        "type": "shell",
        "inline": ["reboot"]
    }
    ```
  - Installation des paquets :
    - Si fw :
      ```
      {
      "type": "shell",
      "inline": ["apt-get update -y && apt-get install -y tcpdump"]
      }
      ```

- Construire une VM
  - Démo avec fw.json
    - `packer validate fw/fw.json`
    - `packer build fw/fw.json`
    - `packer validate ./scripts/provisionner_ip_whoami.json`
    - `packer build ./scripts/provisionner_ip_whoami.json`
    - `packer validate ./scripts/fw/hostnamectl.json`
    - `packer build ./scripts/fw/hostnamectl.json`
    - `packer validate ./scripts/fw/hosts.json`
    - `packer build ./scripts/fw/hosts.json`

- `packer validate ./scripts/fw/reboot.json`
- `packer build ./scripts/fw/reboot.json`

- `packer validate ./scripts/fw/paquetages.json`
- `packer build ./scripts/fw/paquetages.json`

Synthèse 1 :

Tout d'abord, on a cloné un dépôt local avec git et configuré git avec notre nom et adresse email.

On a fait un commit, la gestion collaborative, la synchronisation avec git pull.

Ensuite, on a affiché nos contributions au dépôt.

Puis, on a compris la syntaxe d'un provisionner et le rôle d'un builder Packer.

Synthèse 2 :

Décommenter :

```
sed -E -i.SAVE \
's/^net.ipv4.ip_forward/#net.ipv4.ip_forward/' \
/etc/sysctl.conf
```

- copie du script (fichier) sur la VM dans */root*

```
{
    "type": "file",
    "source": "/home/etudiant/app/fw/routing.sh ",
    "destination": "/root/routing.sh"
}
```

- Exécution du script :

```
{
    "type": "shell",
    "inline": ["cd /root && chmod +x routing.sh && ./routing.sh"]
}
```

On a activé le routage persistant.

On a utilisé sed pour décommenter une ligne mais aussi pour commenter une ligne

On a automatisé avec Packer : la gestion d'un serveur SQL avec un serveur web, d'un collecteur.

Le fichier du load-balancer est dans le répertoire du proxy.

L'usage de git était pragmatique.

Synthèse 3 :

On a réalisé la recette de notre application avec une exception pour le proxy.

En testant sur l'URL du load-balancer.

Les messages sont bien affichés.

2.2. Serveur Git

Pour mettre votre travail en commun, vous utiliserez le serveur `Git` hébergé par votre prof. Pour l'utiliser, installez simplement le paquetage `git` sur votre PC et demandez l'adresse IP du serveur à votre prof.

Pour rappel, vous devez travailler uniquement en tant qu'utilisateur `etudiant`. *Travailler en `root` n'est pas nécessaire … et ne vous apportera que des ennuis.*

Téléchargez le fichier `id_rsa`[3] sur le serveur Web de votre prof. Copiez ce fichier dans votre dossier `.ssh` et vérifiez que vous pouvez maintenant vous connecter en SSH sans mot de passe.

Clonez le dépôt* `app.git` puis déplacez-vous dans le dossier créé. Observez son organisation. Il contient :

- Un dossier par élément de l'infrastructure (`fw`, `lb`, `www1`, etc.).

6.1. Git

Cloner le dépôt `perlimpinpin.git` depuis le serveur `198.51.100.42` :

```
git clone 198.51.100.42:perlimpinpin.git
```

L'ensemble des fichiers du dépôt est téléchargé et placé dans le dossier `perlimpinpin`, créé automatiquement dans le dossier courant.

Pour distinguer facilement vos contributions de celles de vos collègues, personnalisez la configuration* de Git en indiquant votre nom et adresse email.

Personnaliser la configuration :

```
git config --global --edit
```

Modifier notamment les paramètres `name` et `email`.

touch {fw,lb,www1,…}.json

Commencez par créer un fichier `{fw,lb,www1,…}.json` (*vide*) dans votre dossier, puis ajoutez-le* au dépôt Git. Indiquez, comme message de *commit* : "`création du fichier de conf Packer`".

Git add

Se placer dans le répertoire du dépôt local, puis :

```
git add proxy/proxy.json
git commit proxy/proxy.json
git push origin master
```

Git pull

Attendez que d'autres collègues parviennent à cette étape, puis faites un `pull`* sur le dépôt. Vous verrez alors apparaitre les fichiers `.json` de vos collègues dans votre dépôt local !

Faire un `pull` sur le dépôt du serveur :

```
git pull
```

Récupère toutes les mises à jour `push`ées par les autres utilisateurs depuis votre dernier `pull`.

Git log

Affichez les contributions du dépôt* pour vous rendre compte de qui a fait quoi depuis le démarrage du projet.

Afficher toutes les contributions du dépôt :

```
git log
```

Inclut les contributions de tous les collaborateurs.

Un fichier de configuration `Packer` doit contenir au moins :

- Un *builder* (directive `builder`*).

  Il est responsable de la création de la VM (à partir d'une image de base) et de son exportation. Il existe des *builders* pour `VirtualBox`, pour `VMware Workstation`, pour `Amazon EC2`, etc.

- Un ou plusieurs *provisionner(s)* (directive `provisionners`*).

  Chaque *provisionner* permet, par exemple, de copier un fichier ou d'exécuter une commande sur la VM

Ces informations permettent maintenant de comprendre le rôle du fichier `exemple/dns.json` :

Il se base sur une image OVA `debian-stretch.ova` située dans le répertoire personnel d'`etudiant` et dont le login est `root` / `vitrygtr`. Il s'en sert pour construire une VM `VirtualBox` nommée `DNS-GAU-FR-1` et équipée de deux cartes réseau. La seconde carte est en mode `Réseau interne` et branchée à un switch virtuel nommé `vSwitch42`.

Le provisionnement de la VM consiste à installer le paquetage `bind9`, copier quatre fichiers de configuration, et enfin exécuter le script `install.sh`.

A la fin de la construction de la VM, sa première carte réseau sera configurée en mode `Bridge` sur la carte physique `eth0`.

3.1. Packer

Prenez quelques minutes pour étudier le fichier de configuration `exemple/dns.json`, comprendre sa syntaxe*, ainsi que le rôle d'un builder* et des provisionners*.

Il s'agit de la clé de voûte de `Packer`. Vous devez maitriser cet aspect pour réaliser ce projet !

Informations requises par le builder :

Propriété	Rôle
`type`	`virtualbox-ovf` indique que l'on veut construire une VM à partir d'une image OVA fournie
`source_path`	Le chemin de l'image OVA servant de base; peut être relatif (par rapport à l'emplacement du fichier JSON) ou absolu
`ssh_username` `ssh_password`	Les identifiants root de l'image de base
`vm_name`	Le nom de la VM à construire
`vboxmanage`	Une liste de commandes `VBoxManage*` exécutées au début de la construction de la VM, pour la paramétrer
`vboxmanage_post`	Identique à `vboxmanage`, mais les commandes sont exécutées à la fin de la construction de la VM

Liste des *provisionners* utiles :

Provisionner	Rôle
`file`	Copier un fichier depuis l'hôte vers la VM
`shell`	Exécuter, soit une (ou plusieurs) commandes sur la VM (directive `inline`), soit un script présent sur l'hôte (directive `script`). Dans ce dernier cas, le script est d'abord copié sur la VM, rendu exécutable, puis exécuté

1. Écriture d'une partie du fichier de configuration :

   Par exemple, commencez avec une configuration contenant uniquement un builder, mais pas de provisionner (indiquez `"provisioners": []`).

2. Test en local :

   Vérifiez que vous pouvez construire une VM avec ce fichier. Corrigez les erreurs si ce n'est pas le cas. Si `Packer` parvient à construire la VM (pas de message d'erreur en rouge …), démarrez-la avec la GUI de `VirtualBox` et vérifiez que sa configuration est comme vous l'espérez.*

3. `Commit` et `push*` sur le serveur Git :

   Cette partie est importante pour garder une trace des modifications apportées à la configuration, et pour les partager avec vos collègues. N'oubliez pas d'indiquer un message de 'commit' précis, qui résume vos contributions.

4. Recommencez !

   Cette fois en ajoutant les provisionners nécessaires pour répondre au cahier des charges (adressage, paquetages, etc.).

   Attention, avant de passer au cycle suivant, vous devez faire un peu de nettoyage[4] :

```
# Supprimer la VM de VirtualBox
# et effacer tous ses fichiers
VBoxManage unregistervm fw --delete

# Supprimer le dossier créé par Packer
rm -rf fw/output-virtualbox-ovf/
```

6.2. Packer

Construire une VM :

`Packer` prend en entrée un fichier de configuration au format JSON, qui contient toutes les instructions de construction.

```
# Vérifier le fichier de conf
packer validate exemple/dns.json

# Construire une VM en utilisant les instructions
# du fichier de conf
packer build exemple/dns.json
```

Commencez par un *`pull`* si vous ne l'avez pas fait depuis un moment : entretemps, un collaborateur a peut-être fait un *`push`*, et vous devez nécessairement mettre à jour votre dépôt local avant d'envoyer vos contributions.

Ensuite :

```
git commit -a
```

Un éditeur de texte s'ouvre : vous devez alors résumer (en français) les modifications que vous avez apportées depuis votre dernier `push`.

Cette étape est importante, surtout lorsque vous travaillez en groupe sur les mêmes fichiers : il permet de garder une trace des contributions, de savoir qui a fait quoi et surtout pour quelle raison !

Enfin, synchroniser vos contributions avec le dépôt distant :

```
git push
```

`Bridge` pour permettre un accès à l'application depuis l'extérieur.

Les cartes réseau doivent être configurées dans le *builder* de votre fichier de configuration `Packer`

Les cartes doivent être configurées en adressage persistant*.

Soyez rigoureux ! Si vous n'utilisez pas les noms et adresses indiqués, les VM ne pourront pas communiquer ensemble lors de la phase d'intégration (phase 2).

Conseils :

Dans le dossier de votre dépôt local, créez un fichier `interfaces` avec la configuration IP adéquate. Utilisez un *provisionner** de type `file` pour envoyer ce fichier dans le répertoire `/etc/network` de la VM.

N'oubliez surtout pas d'ajouter ce fichier au dépôt Git* pour que vos collègues puissent en bénéficier !

Configurer `eth0` en adressage statique persistant (adresse `203.0.113.10/24`) :

Modifier `/etc/network/interfaces` :

```
auto eth0
iface eth0 inet static
     address 203.0.113.10/24
```

Liste des *provisionners* utiles :

Provisionner	Rôle
`file`	Copier un fichier depuis l'hôte vers la VM
`shell`	Exécuter, soit une (ou plusieurs) commandes sur la VM (directive `inline`), soit un script présent sur l'hôte (directive `script`). Dans ce dernier cas, le script est d'abord copié sur la VM, rendu exécutable, puis exécuté

```
{
  "type" : "file",
  "source" : "/home/etudiant/app/fw/etc/network/interfaces",
  "destination" : "/etc/network/interfaces"
}
```

Se placer dans le répertoire du dépôt local, puis :

```
git add proxy/proxy.json
git commit proxy/proxy.json
git push origin master
```

3.3. *Configuration système*

Votre VM doit porter le nom indiqué dans la Fig. 4. Votre configuration `Packer` doit donc renommer* la VM pendant sa construction.

> **Conseils** :
> Là encore, vous pouvez vous appuyer sur des *provisionners* de type `file` pour envoyer des fichiers déjà prêts, ou sur des *provisionners* de type `shell` pour écraser ou remplacer le contenu des fichiers d'origine. Vous êtes libres de choisir, mais n'oubliez pas que le renommage nécessite deux étapes …

```
{
    "type" : "shell",
    "inline" : ["sudo hostnamectl set-hostname fw "]
}
```

En trois étapes :

1. Changer le nom :

```
hostnamectl set-hostname web2
```

2. Éditer le fichier `/etc/hosts` et remplacer les deux occurrences de `www2` par `web2` (deuxième ligne) :

```
127.0.0.1       localhost
127.0.1.1       www2.localdomain       www2
...
```

3. Fermer le terminal et le rouvrir. Observer le prompt du terminal* pour confirmer que les modifications ont été prises en compte.

Remarque : seul le `root` a le droit d'effectuer ces actions …

```
{
"type" "file",
"source" "/home/etudiant/repo/fw/etc/hosts",
"destination" "/etc/hosts "
}
```

```
- 3 ème étape : reboot
        ```json
        {
            "type": "shell",
            "inline": ["reboot"]
        }
    ```
```

fw	tcpdump	*Analyseur réseau en CLI*
lb	haproxy	*Load balancer*
www1 www2	apache2 php7.0 php7.0-mysql	*Installe le serveur Web Apache et ses extensions pour PHP7*
mysql	mysql-server	*Installe le serveur de base de données MySQL*
log	beep	*Bip bip*

```
{
    "type" : "shell",
    "inline" : ["sudo apt-get update -y && sudo apt-get install -y tcpdump"]
}
```

6.2. *Packer*

Construire une VM :

`Packer` prend en entrée un fichier de configuration au format JSON, qui contient toutes les instructions de construction.

```
# Vérifier le fichier de conf
packer validate exemple/dns.json

# Construire une VM en utilisant les instructions
# du fichier de conf
packer build exemple/dns.json
```

Synthèse 1 :

4.1. *Firewall*

Le routage doit être activé* de manière persistante.

Le script d'activation du firewall (voir Tableau 1) doit être disponible dans le répertoire `/root` et avoir les droits d'exécution.

Vous devez donc modifier le contenu d'un fichier de manière automatisée (pas de manière interactive avec un éditeur de texte). Pour cela, nous vous conseillons d'utiliser une commande `sed` pour décommenter une ligne*, et d'utiliser un *provisionner* pour exécuter cette commande sur la VM.

Activer le routage :

Décommenter la ligne suivante dans `/etc/sysctl.conf` :

```
#net.ipv4.ip_forward=1
```

Le changement prendra effet au prochain démarrage.

```
{
"type" "file",
"source" "/home/etudiant/repo/fw/routing.sh ",
"destination" "/root/routing.sh"
}

{
"type" "shell",
"inline" ["sudo cd /root && sudo chmod +x routing.sh &&
./routing.sh"]
}
```

4.2. *Load balancer*

La passerelle par défaut doit être configurée de manière persistante (il s'agit de `fw` …).

Le fichier de configuration du *load balancer* (voir Tableau 1) doit être placé dans le répertoire `/etc/haproxy`.

File - Provisioners | Packer by HashiCorp

Shell - Provisioners | Packer by HashiCorp