

# Atelier 8 : Image building et cloud-init

## Objectifs de l'atelier :

- **Exercice 1** : Utiliser **cloud-init** pour configurer une machine virtuelle avec Vagrant (créer un utilisateur, installer des paquets).
- **Exercice 2** : Créer un **template Packer** pour générer une image VirtualBox avec Nginx pré-installé.
- **Exercice 3** : Utiliser la commande `packer build` pour construire l'image.

## Pré-requis

Avant de commencer, assurez-vous que les outils suivants sont installés :

1. **Vagrant** : pour créer et gérer des VMs locales.
2. **VirtualBox** : pour gérer la virtualisation des machines.
3. **Packer** : pour créer des images avec un template HCL2.
4. **Git** : pour versionner les fichiers.

## Exercice 1 : Utiliser cloud-init avec Vagrant (45 min)

### Objectif :

Apprendre à utiliser **cloud-init** via `user-data` dans Vagrant pour configurer une machine virtuelle au premier démarrage.

### Étapes :

1. **Créer un répertoire pour l'atelier :**

```
mkdir vagrant-cloud-init
cd vagrant-cloud-init
```

#### Créer le fichier Vagrantfile :

Ce fichier permettra à Vagrant de créer et démarrer une VM Ubuntu avec cloud-init configuré.

```
# Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64" # Image Ubuntu 18.04
  config.vm.network "private_network", type: "dhcp"

  # Configuration cloud-init via user-data
  config.vm.provision "file", source: "user-data", destination:
"/tmp/user-data"
```

```
# Exécution d'un script pour cloud-init
config.vm.provision "shell", inline: "echo 'Running cloud-init...'"

end
```

### Créer un fichier `user-data` :

Ce fichier sera utilisé par cloud-init pour configurer la VM lors du premier démarrage.

```
# user-data
#cloud-config

users:
  - name: devops
    sudo: ALL=(ALL) NOPASSWD:ALL
    shell: /bin/bash

packages:
  - nginx
  - git

runcmd:
  - echo "Cloud-init complete!" > /var/log/cloud-init.log
  - systemctl start nginx
  - systemctl enable nginx

write_files:
  - path: /etc/motd
    content: |
      Welcome to your cloud-init configured VM!
```

### 2. Lancer Vagrant :

Initialisez et démarrez la VM avec Vagrant, puis appliquez le provisioning avec cloud-init.

```
vagrant up
```

### 3. Vérifier l'installation :

Une fois la VM démarrée, connectez-vous via SSH pour vérifier que l'utilisateur `devops` a bien été créé et que Nginx est installé.

```
vagrant ssh
whoami      # Devrait afficher 'devops'
curl http://localhost # Doit afficher la page par défaut de Nginx
```

## Vérification :

- Le fichier `/etc/motd` devrait afficher "Welcome to your cloud-init configured VM!".
- L'utilisateur `devops` doit être créé avec les bonnes permissions.
- Nginx doit être installé et en fonctionnement.

## Exercice 2 : Créer un template Packer pour une image VirtualBox (45 min)

### Objectif :

Créer un template Packer pour générer une image **VirtualBox** avec **Nginx** pré-installé.

### Étapes :

#### 1. Créer un répertoire pour Packer :

```
mkdir packer-image
cd packer-image
```

#### 2. Créer un fichier template Packer (packer.pkr.hcl) :

Ce fichier va définir le processus de création de l'image VirtualBox avec l'installation de Nginx.

```
# packer.pkr.hcl

variable "virtualbox_version" {
  default = "6.1.24"
}

source "virtualbox-iso" "ubuntu" {
  iso_url      = "http://releases.ubuntu.com/20.04/ubuntu-20.04.2.0-
desktop-amd64.iso"
  iso_checksum = "sha256:xxx"
  ssh_username = "vagrant"
  ssh_password = "vagrant"
  ssh_wait_timeout = "30m"
  vm_name      = "ubuntu-nginx-vm"
  disk_size    = 10240
  guest_os_type = "Ubuntu_64"
  http_directory = "./http"

  boot_command = [
    "<wait>", "<enter>", "sudo apt-get update -y", "<enter>"
  ]
}

build {
  sources = ["source.virtualbox-iso.ubuntu"]

  provisioner "shell" {
    inline = [
      "sudo apt-get install -y nginx",
      "echo 'Welcome to Nginx VM!' > /var/www/html/index.html"
    ]
  }
}
```

### Explications :

- **Source** : Crée une VM avec VirtualBox en téléchargeant une image Ubuntu.
- **Provisioner** : Installe Nginx et modifie la page par défaut d'index.
- 2. **Construire l'image avec Packer** :  
Utilisez la commande `packer build` pour créer l'image VirtualBox.
- 3. `packer build packer.pkr.hcl`
- 4. **Vérifier l'image** :  
Une fois la construction terminée, ouvrez VirtualBox pour voir la nouvelle VM. Vous pouvez aussi la tester avec `vagrant` :
- 5. `vagrant up --provider=virtualbox`
- 6. **Vérification** :
  - Connectez-vous à la VM via SSH (`vagrant ssh`) et vérifiez l'installation de Nginx :
  - `curl http://localhost` # Doit afficher la page par défaut de Nginx

## Exercice 3 : Construire l'image avec Packer (30 min)

### Objectif :

Utiliser la commande `packer build` pour construire l'image et vérifier qu'elle fonctionne comme prévu.

### Étapes :

1. **Assurez-vous que le template Packer est prêt** (cf. Exercice 2).
2. **Exécutez la commande suivante pour construire l'image** :

```
packer build packer.pkr.hcl
```

Cela va lancer la création de l'image VirtualBox avec les étapes définies dans le template (téléchargement de l'ISO, création de la VM, installation de Nginx).

3. **Vérification** :  
Une fois le processus terminé, Packer générera une image VirtualBox que vous pourrez télécharger dans VirtualBox.  
Vous pouvez également utiliser `vagrant` pour tester l'image :

```
vagrant init  
vagrant up --provider=virtualbox
```

4. **Vérification finale** :
  - Connectez-vous à la VM avec `vagrant ssh`.
  - Testez que Nginx fonctionne en accédant à `http://localhost`.

# Résumé & Bonnes pratiques

## Conseils :

- **Idempotence** : Utilisez des outils comme **Ansible** pour la configuration au lieu de simples scripts Shell afin d'assurer une configuration prévisible et réutilisable.
- **Versioning** : Utilisez un gestionnaire de version comme Git pour suivre les modifications apportées à vos templates Packer et Vagrantfiles.
- **Testabilité** : Après la création des images, effectuez des tests pour vérifier que l'image répond aux attentes avant de la distribuer (par exemple, vérifier l'état des services comme Nginx).

## Prochaines étapes suggérées :

- **Automatiser la création d'images dans un pipeline CI/CD** avec **Jenkins** ou **GitLab CI**.
- **Créer des templates Packer plus complexes**, intégrant des étapes comme l'installation de bases de données ou d'applications spécifiques.

Cela conclut l'**atelier complet pour "Image building et cloud-init"**. Il comprend les exercices pour apprendre à utiliser **cloud-init** avec Vagrant, créer des images avec **Packer**, et automatiser la construction et la configuration des machines virtuelles. Vous pouvez désormais ajouter des tests dans vos pipelines pour garantir l'intégrité de vos images à chaque modification.