DÉCOUVERTE ET UTILISATION DE L'OUTIL PACKER

Introduction

Dans ce chapitre, nous verrons comment **créer nos propres images machine depuis l'outil Packer**, mais commençons d'abord par **définir ce qu'est une "image machine"**.

Les images machine

Une image machine est une unité statique unique qui contient un système d'exploitation préconfiguré avec des logiciels préinstallés qui sera ensuite utilisée pour **créer rapidement vos nouvelles machines**. Les formats des images machine changent pour chaque plate-forme, certains par exemple incluent des AMIs pour les instances EC2 d'AWS, des fichiers VMDK/VMX pour VMware, OVF pour VirtualBox ...

C'est quoi Packer ?

Packer est un outil open source utilisé qui permet de créer des images machine pour plusieurs plates-formes à partir d'une configuration source unique. Il reste un outil léger qui fonctionne sur tous les principaux systèmes d'exploitation. Il crée donc essentiellement des images prêtes à l'emploi avec le système d'exploitation et les logiciels supplémentaires.

Rappelez-vous, dans notre chapitre sur la création d'une infrastructure AWS hautement disponibles depuis l'outil Terraform , nous avions utilisé le user-data pour installer et configurer notre application php, et je vous avais recommandé

d'utiliser votre propre AMI. Et ça tombe bien ! Car la création de cette image peut très bien être faite à partir de l'outil Packer et ça sera notre objectif à la fin de ce chapitre.

Pourquoi utiliser Packer ?

Voici une **liste des avantages offerts par l'outil Packer** :

- **Rapidité** : une fois vos images créées depuis Packer, vous pouvez les utiliser sur des machines qui seront provisionnées entièrement et configurées en quelques secondes, plutôt que plusieurs minutes ou heures sur les autres méthodes (ex: user-data sur AWS)
- **Portabilité** : packer vous permet de créer plusieurs images machine sur plusieurs environnements, ex : AWS, cloud privé (ex: OpenStack) et des solutions de virtualisation comme VMware ou VirtualBox, etc ...
- **Stabilité** : packer installe et configure tous les logiciels d'une machine au moment de la création de l'image. Donc s'il existe des bugs dans vos scripts, ils seront alors aussi tôt détectés, avant même la création de votre infrastructure.
- **Tests** : une fois qu'une image machine est créée, elle peut être rapidement lancée et testée pour vérifier que vos tâches fonctionnent correctement. Si tel est le cas, vous pouvez être sûr que toutes les autres machines lancées à partir de cette image fonctionneront correctement et rapidement.
- **Facilité** : vous pouvez créer des images depuis Packer à partir d'un seul fichier de configuration JSON très simple à configurer.

Finissons maintenant avec la théorie et passons à la pratique !

Installation de Packer

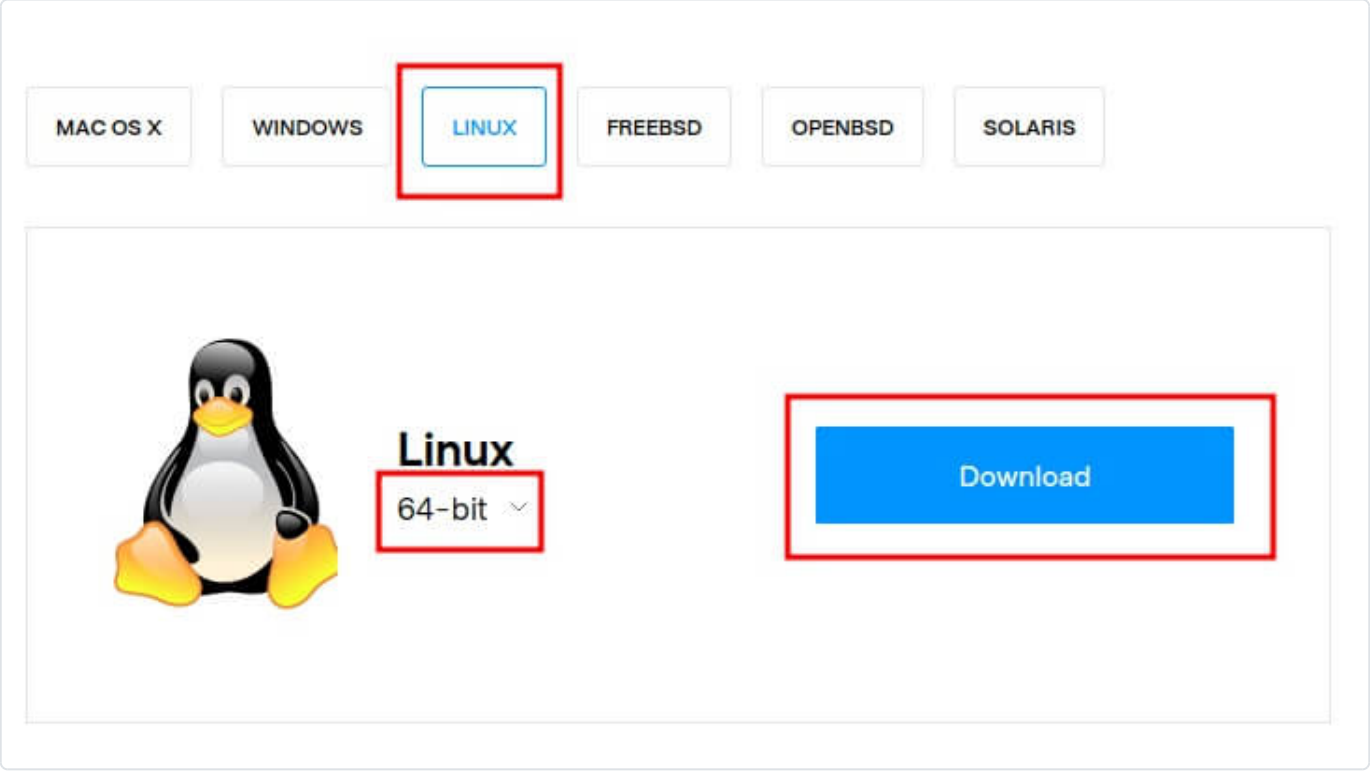
Packer peut être installé des manières suivantes :

- Utilisation de binaires précompilés disponibles pour toutes les plates-formes et architectures prises en charge.
- Installation à partir des sources.
- Installation depuis le gestionnaire de package de votre distribution.

Dans ce chapitre nous **installerons Packer depuis les binaires précompilés**.

Linux

Commençons déjà par télécharger le package approprié pour votre système en cliquant ici. Dans mon cas je suis sur une machine 64 bits Linux sous la distribution Ubuntu. Je vais donc sélectionner le package suivant :



Vous aurez donc un fichier au format `.zip` qui contient votre binaire que vous devez dézipper dans le dossier `/usr/local/bin`, comme suit :

```
sudo unzip  packer*_linux_amd64.zip -d /usr/local/bin/
```

Vous pouvez ensuite vérifier que l'installation a fonctionné en affichant la version de Packer :

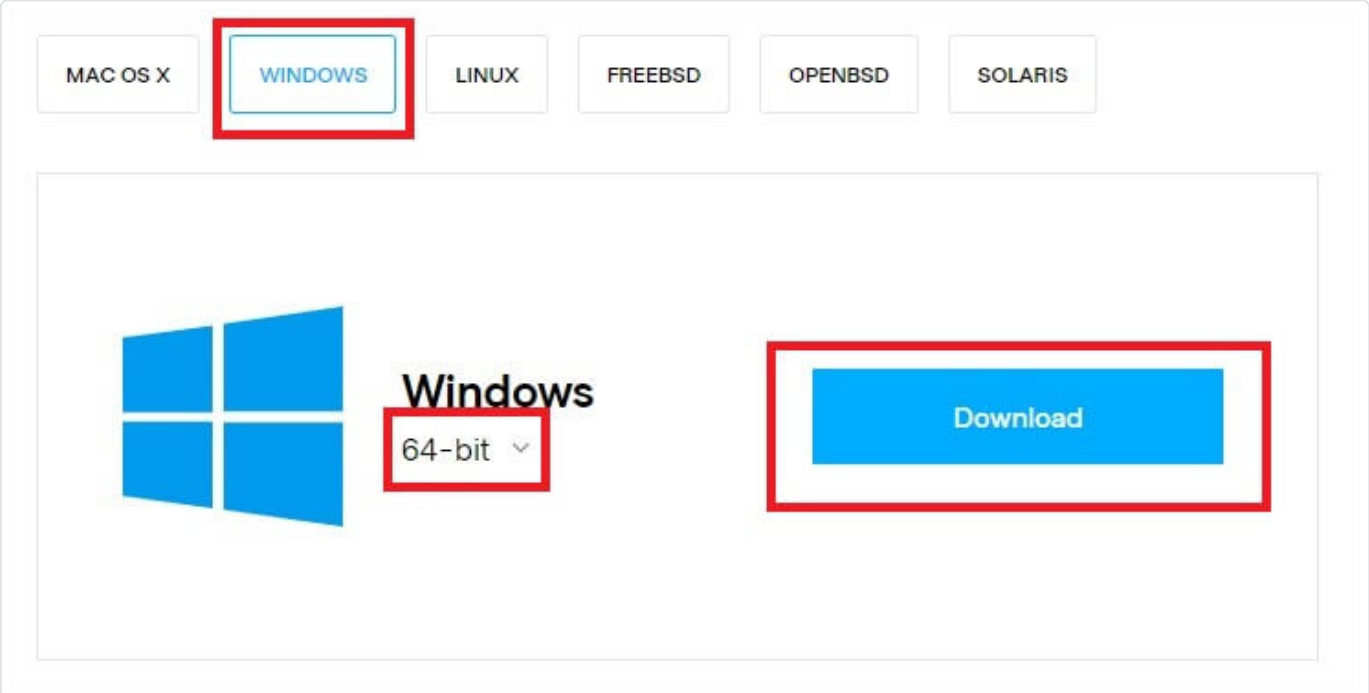
```
packer -v
```

Résultat :

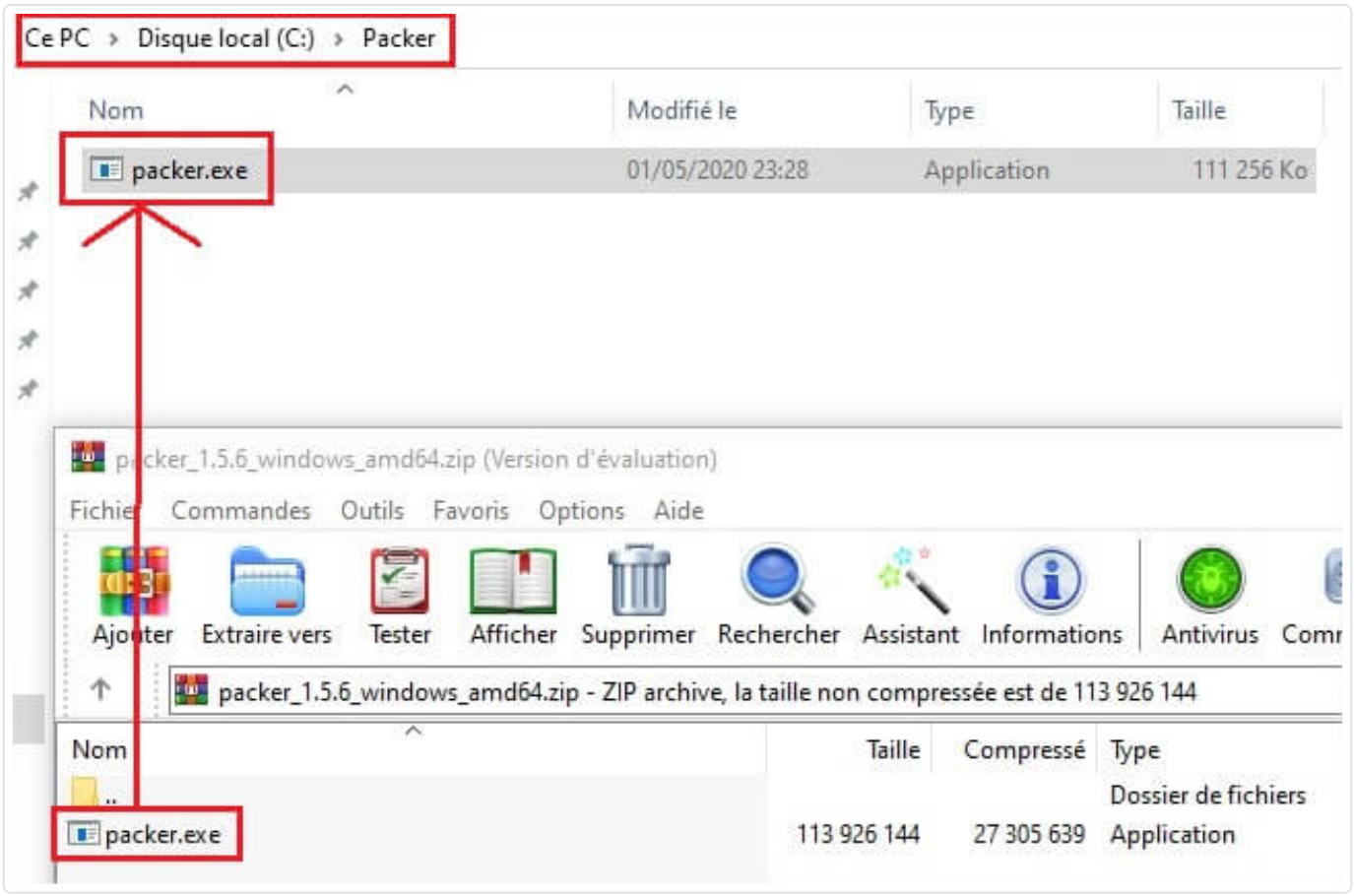
```
1.5.6
```

Windows

Commencez par télécharger le binaire adéquat à votre architecture, dans mon cas je suis sous une machine Windows 64 bits :

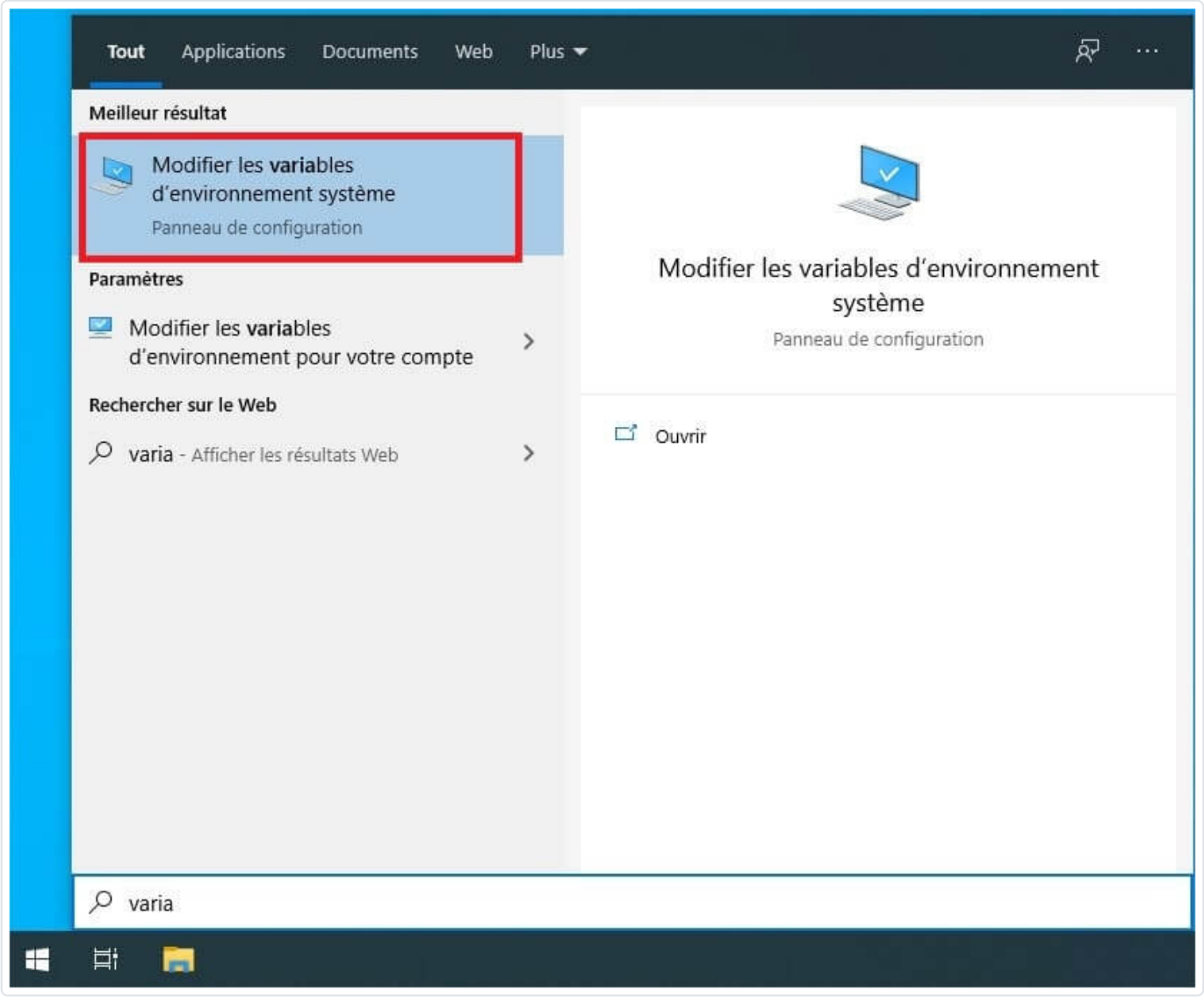


Une fois le téléchargement terminé, créez un dossier sur votre lecteur `C:\` où vous placerez votre exécutable Packer. Allez ensuite trouver le binaire Terraform dans l'explorateur de fichiers et extrayez ce fichier zip dans le dossier que vous avez créé précédemment :

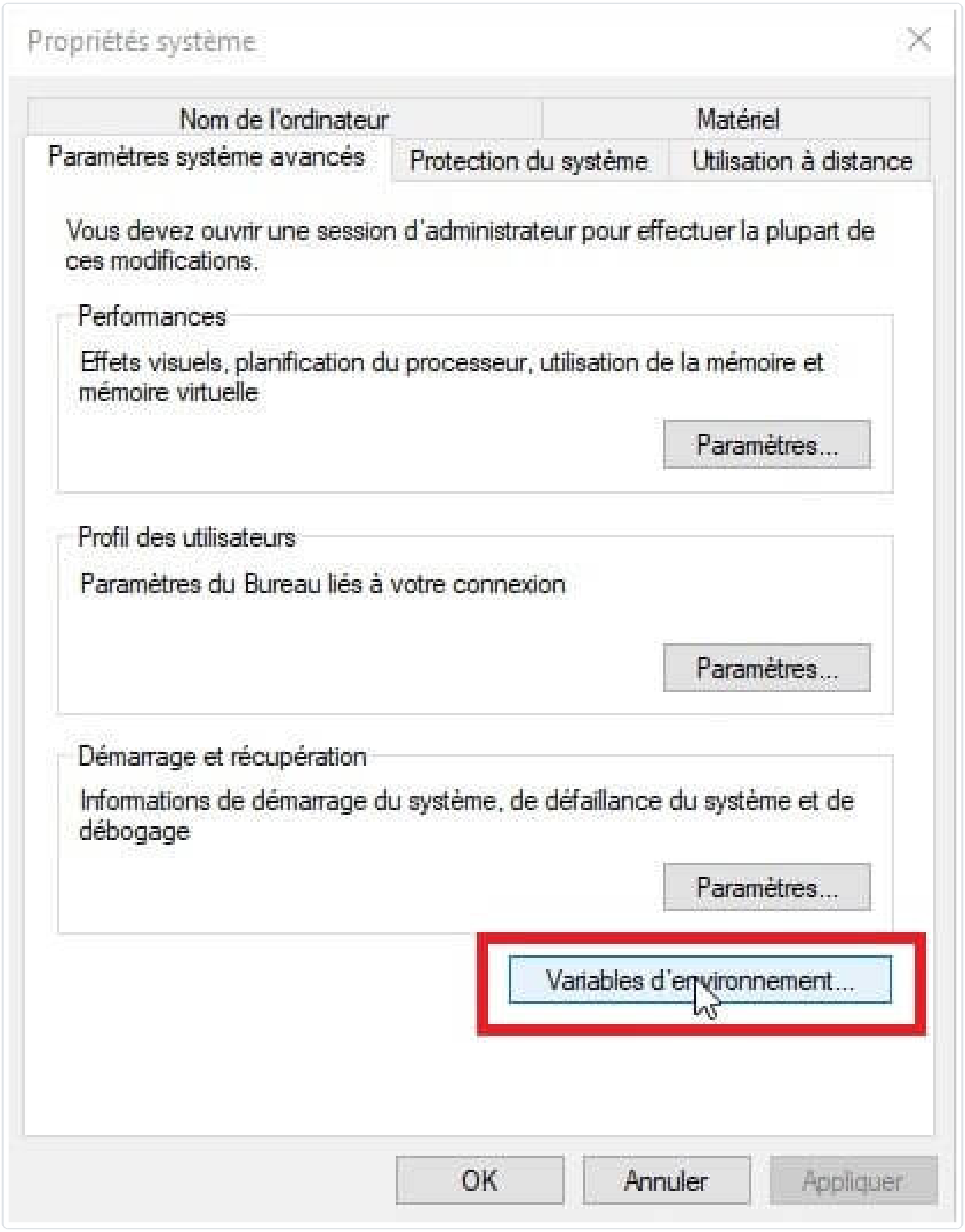


Maintenant il suffit de rajouter votre exécutable Packer à votre variable d'environnement nommée **PATH**, c'est la variable système utilisée par Windows pour localiser les fichiers exécutables, ainsi vous pourrez exécuter le programme Packer depuis n'importe où en ligne de commande. Pour cela, suivez les étapes suivantes :

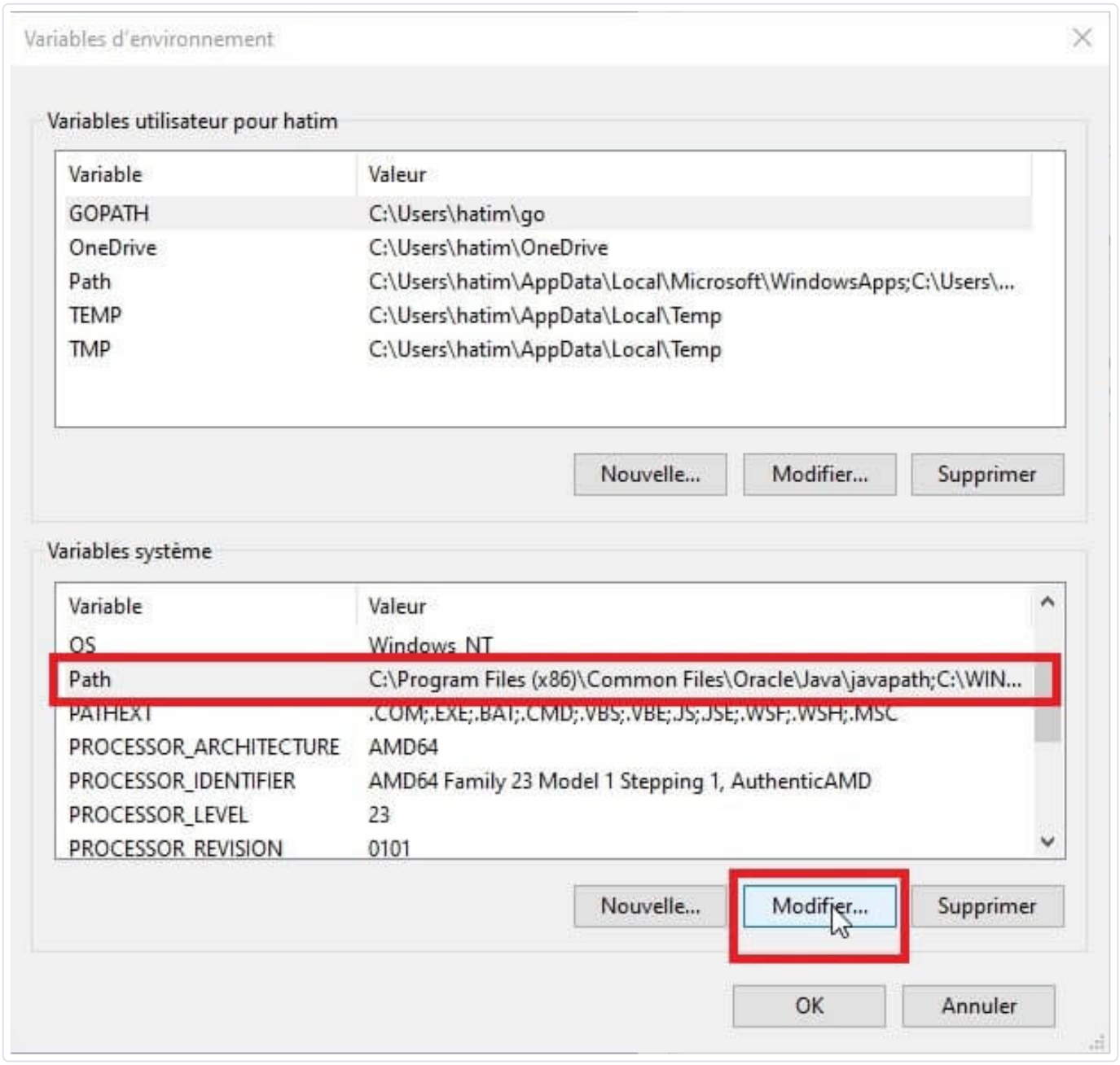
Commencez par ouvrir votre menu Démarrer et tapez "environnement" et la première chose qui apparaît devrait être "Modifier les variables d'environnement système".



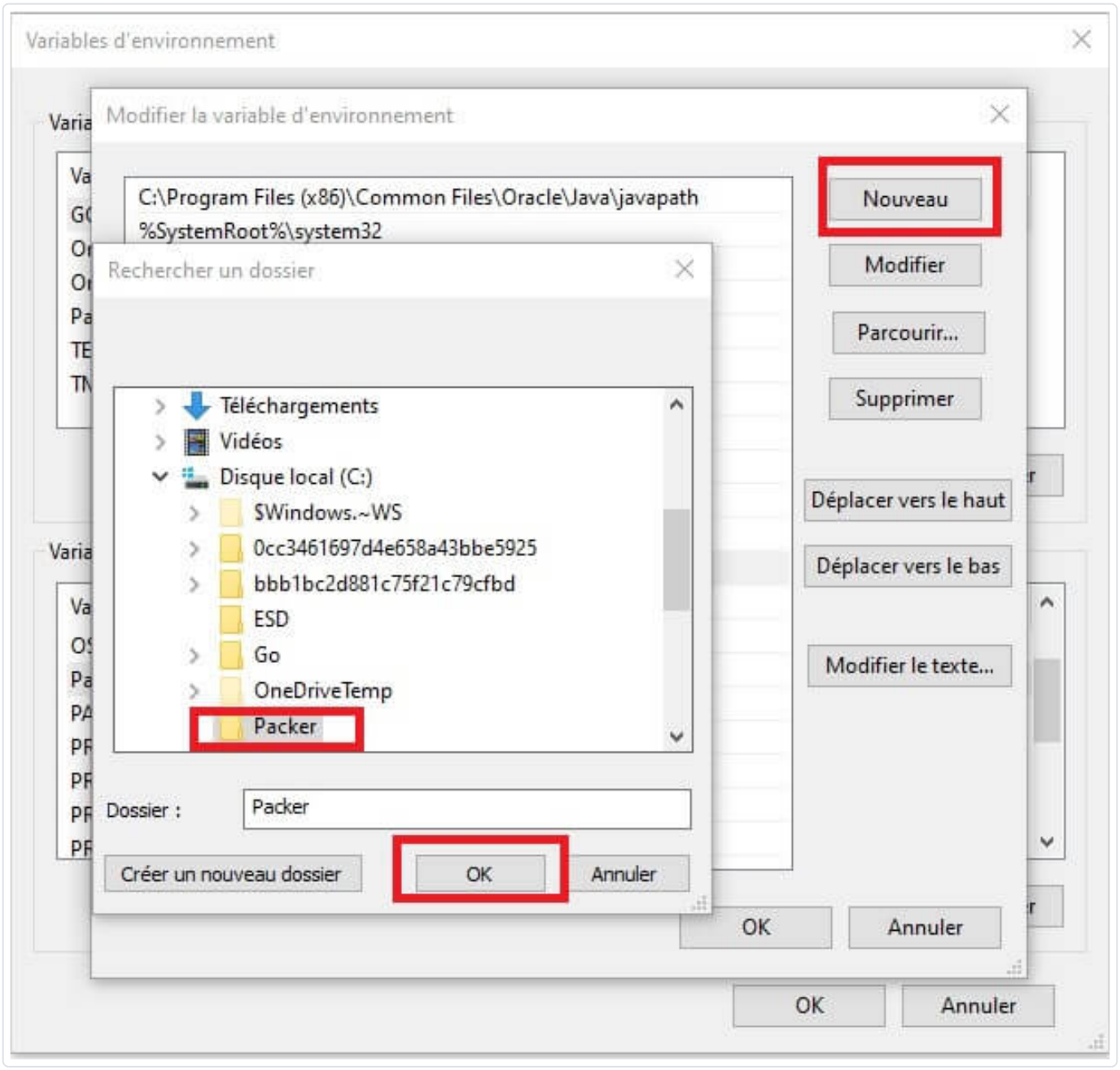
Cliquez dessus et vous devriez voir cette fenêtre, cliquez ensuite sur le bouton "Variables d'environnement" :



Dans la section inférieure où il est indiqué "variables système", recherchez la variable nommée **PATH** et cliquez sur modifier :



Cliquez ensuite sur le bouton "Parcourir" et ajoutez le chemin du dossier où se trouve le binaire **packer.exe** :



Comme pour l'installation sous Linux, il ne reste plus qu'à vérifier si Packer s'est installé avec succès sur votre machine Windows. Ouvrez votre terminal Windows et lancez la commande suivante :

```
packer -v
```

Résultat :

```
1.5.6
```

Manipulation de Packer

Création de notre image

Une fois Packer installé, plongeons-nous dedans et **construisons notre première image machine pour le provider `AWS`**.

> **Information**
>
> Assurez-vous d'utiliser une instance EC2 de type "t2.micro" pour construire votre image, qui alors sera éligible à l'offre gratuite d'AWS.

Dans la terminologie Packer, le fichier de configuration utilisé pour définir une image dans Packer est appelé "template", ce template devra être sous le format JSON.

Je vous dévoile d'abord le code de notre template, puis nous passerons brièvement en revue chaque section. Commencez déjà par créer un fichier json, dans mon cas j'ai choisi de le nommer **ubuntu-ami.json** et remplissez-le avec le contenu suivant :

```
{
    "variables": {
      "aws_access_key": "",
      "aws_secret_key": ""
    },
    "builders": [
      {
        "type": "amazon-ebs",
        "access_key": "{{user `aws_access_key`}}",
        "secret_key": "{{user `aws_secret_key`}}",
        "region": "us-east-1",
```

```
        "source_ami_filter": {
          "filters": {
            "virtualization-type": "hvm",
            "name": "ubuntu/images/*ubuntu-bionic-18.04-amd64-server-*",
            "root-device-type": "ebs"
          },
          "owners": ["099720109477"],
          "most_recent": true
        },
        "instance_type": "t2.micro",
        "ssh_username": "ubuntu",
        "ami_name": "packer-ubuntu-devopssec {{timestamp}}"
      }
    ]
}
```

La section variables

Commençons par l'explication de la section `variables` de votre template. Ici je demande à l'utilisateur de spécifier la clé d'accès et secrète de son utilisateur IAM dans les variables `aws_access_key` et en `aws_secret_key`. Ce type de variable est nommée "**variable utilisateur**", vous pouvez d'ailleurs comme avec la commande `terraform`, utiliser les options `-var` ou `-var-file` :

```
packer build \
-var 'aws_access_key=foo' \
-var 'aws_secret_key=bar' \
template.json
```

Voici la syntaxe à respecter pour **utiliser les variables utilisateurs dans votre code Packer** :

```
{{user `MA_VARIABLE`}}
```

Information

> Pour plus d'informations sur l'utilisation des autres types de variables, rendez-vous à cette page.

Sinon, si vous souhaitez vous passer des variables utilisateurs `aws_access_key` et en `aws_secret_key`, vous pouvez télécharger et configurer votre AWS CLI. Pour rappel, voici la commande à lancer pour configurer votre CLI :

```
aws configure
```

Résultat :

```
AWS Access Key ID [None]: VOTRE_CLE_D_ACCES
AWS Secret Access Key [None]: VOTRE_CLE_SECRETE
Default region name [None]: us-west-1 (REMPLACEZ PAR VOTRE REGION)
Default output format [None]: json
```

Après avoir lancé votre commande, vous retrouverez les informations saisies directement dans le chemin suivant sur Linux :

```
ls -l ~/.aws/
```

Résultat :

```
-rw------- 1 hatim hatim  43 avril 21 10:40 config
-rw------- 1 hatim hatim 116 avril 21 10:40 credentials
```

Sinon sur Windows, ces informations seront stockées ici : `C:\Users\%username%\.aws\`.

La section builders

Étudions désormais la section `builders` où nous trouvons la configuration principale pour la construction de notre image. Dans notre exemple, nous ne configurons qu'un seul builder de type `amazon-ebs`. Il s'agit du constructeur

Amazon EC2 AMI fourni avec Packer. Il se base depuis une AMI source, , en la provisionnant en la reconditionnant en une nouvelle AMI.

Ici, j'utilise le paramètre `source_ami_filter` afin de récupérer automatiquement l'id de mon image source comme nous avions pu le faire dans l'article destiné aux DataSource. Nous récupérerons donc l'id de l'ami Ubuntu sous sa version 18.04 dont le propriétaire est la société canonical avec l'id `099720109477` qui maintient actuellement officiellement cette distribution.

> **Information**
>
> Rappelez-vous que l'id d'une image peut être modifié à tout moment, avec l'action `filters` nous sommes sûrs de récupérer le dernier id de notre AMI Ubuntu.

Ensuite nous gardons l'utilisateur par défaut utilisé par l'ami source qui est dans notre cas `ubuntu`, nous utilisons ensuite le type d'instance `t2.micro` et on nomme notre nouvelle ami avec le préfix `packer-ubuntu-devopssec` et un horodatage.

Approvisionnement

C'est cool, vous venez de créer un template pour votre première image avec Packer. Cependant, cette image n'est en fait qu'un reconditionnement d'une AMI de base existante. La véritable utilité de Packer vient également de la possibilité d'**installer et de configurer des logiciels dans vos images**. En effet, packer prend entièrement en charge l'approvisionnement automatisé afin d'installer des logiciels sur les machines avant de les transformer en images.

Pour cet exemple, commencez d'abord par télécharger les sources de notre application web en cliquant ici et dézipper l'archive dans votre dossier de travail.

Nous utiliserons ensuite le provisionneur de type `shell` intégré et fourni avec Packer pour installer et configurer notre serveur web et le provisionneur de type `file` pour envoyer nos sources vers l'image cible (plus d'informations sur mon article sur les provisionneurs) :

```
{
    "builders": [
    {
        "type": "amazon-ebs",
            "access_key": "{{user `aws_access_key`}}",
            "secret_key": "{{user `aws_secret_key`}}",
            "region": "us-east-1",
            "source_ami_filter": {
                "filters": {
                    "virtualization-type": "hvm",
                    "name": "ubuntu/images/*ubuntu-bionic-18.04-amd64-server-*",
                    "root-device-type": "ebs"
                },
                "owners": ["099720109477"],
                "most_recent": true
            },
            "instance_type": "t2.micro",
            "ssh_username": "ubuntu",
            "ami_name": "packer-ubuntu-devopssec {{timestamp}}"
    }
    ],
    "provisioners": [
    {
        "type": "shell",
        "pause_before": "10s",
        "inline": [
            "sudo apt-get update -y",
            "sudo apt-get install -y apache2",
            "sudo systemctl start apache2",
            "sudo systemctl enable apache2"
        ]
    },
    {
        "type": "file",
        "source": "./src/index.html",
        "destination": "/var/www/html/"
    }]
}
```

Chaque provisionneur dans un template Packer peut prendre la configuration spéciale `pause_before` qui correspond au **temps de pause avant d'exécuter le provisionneur**. Par défaut, il n'y a pas de pause mais il est recommandé d'en implémenter une, afin de laisser suffisamment de temps pour que l'OS s'initialise.

Validation de notre template

Avant d'exécuter notre template et de créer notre image à partir de celui-ci, **validons d'abord la configuration de notre template packer** en exécutant la commande `packer validate`. Cette commande vérifiera la syntaxe ainsi que les valeurs de configuration fournies afin de s'assurer de leurs fiabilités. La sortie doit ressembler à ceci :

```
packer validate ubuntu-ami.json
```

Résultat :

```
Template validated successfully.
```

Construire son image packer

Une fois votre premier template validé, il est temps de **construire votre première image machine**. Cela se fait en appelant la commande `packer build` avec le fichier template. La sortie devrait ressembler à celle ci-dessous (notez que ce processus prend généralement quelques minutes) :

```
packer build ubuntu-ami.json
```

Résultat :

```
amazon-ebs: output will be in this color.

...
...
```

```
==> Builds finished. The artifacts of successful builds a
re:
--> amazon-ebs: AMIs were created:
us-east-1: ami-01ad269378ef4c2af
```

Veuillez noter dans un coin l'ami de votre nouvelle image, car nous l'utiliserons plus tard dans notre configuration Terraform.

Utilisation de notre image

Une fois l'id de notre image récupérée, on peut par la suite l'utiliser dans notre configuration Terraform comme suit :

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_security_group" "my_sg" {
  name        = "allow-ssh"
  description = "security group that allows ssh and all egress traffic"

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "webserver-sg"
  }
}

resource "aws_instance" "my_ec2_instance" {
  ami           = "ami-01ad269378ef4c2af" # notre propre image machine
  instance_type = "t2.micro"
  vpc_security_group_ids = [aws_security_group.my_sg.id]
}

output "instance_ip" {
```

```
  value = aws_instance.my_ec2_instance.public_ip
}
```

Lançons maintenant notre configuration Terraform avec la commande suivante :

```
terraform init && terraform apply
```

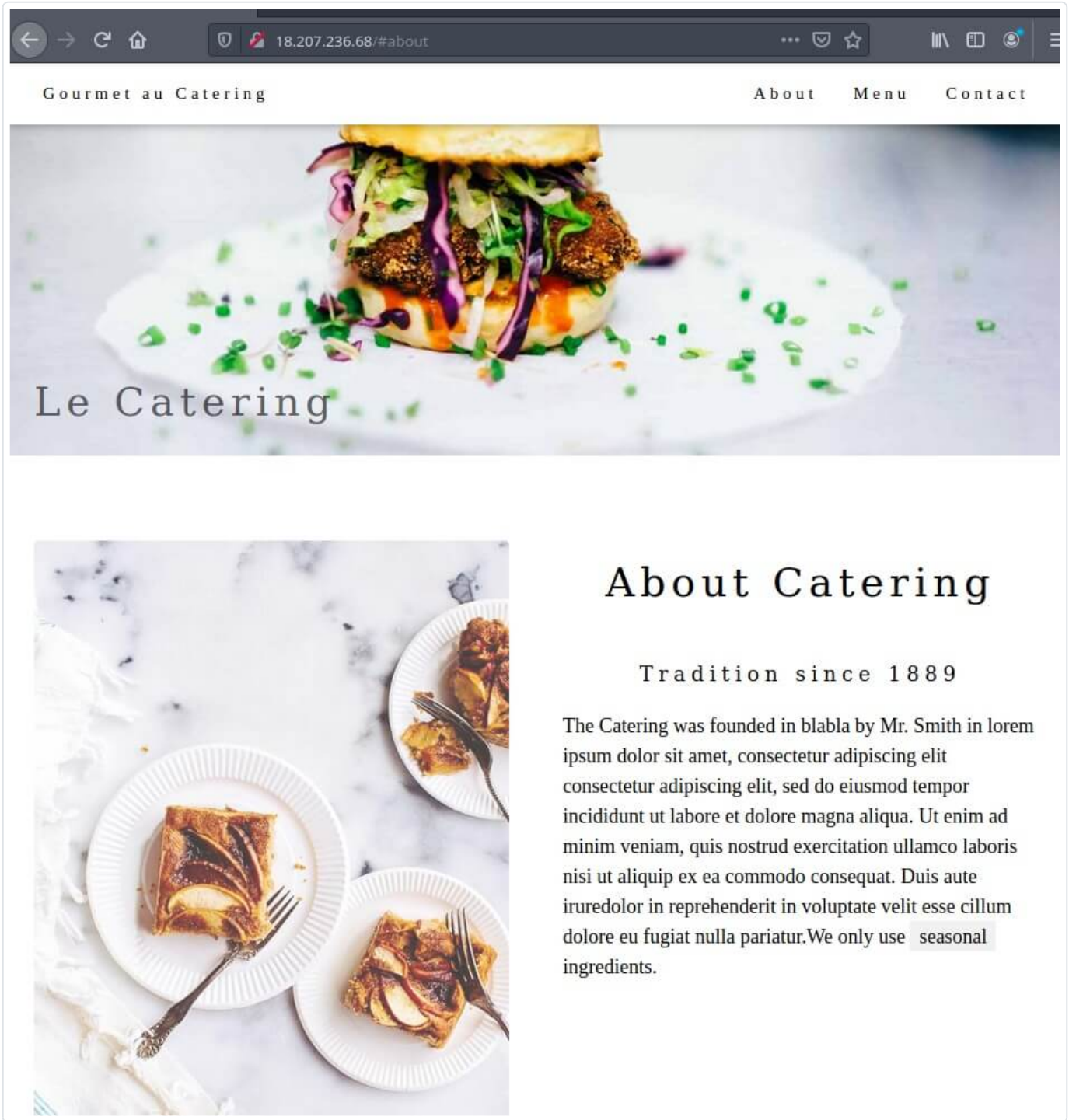
Résultat :

```
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

instance_ip = 18.207.236.68
```

Si vous aviez eu l'habitude d'utiliser les users-data, vous remarquerez que l'utilisation directe d'une image machine est remarquablement plus rapide. Visitons maintenant notre page web depuis notre navigateur :



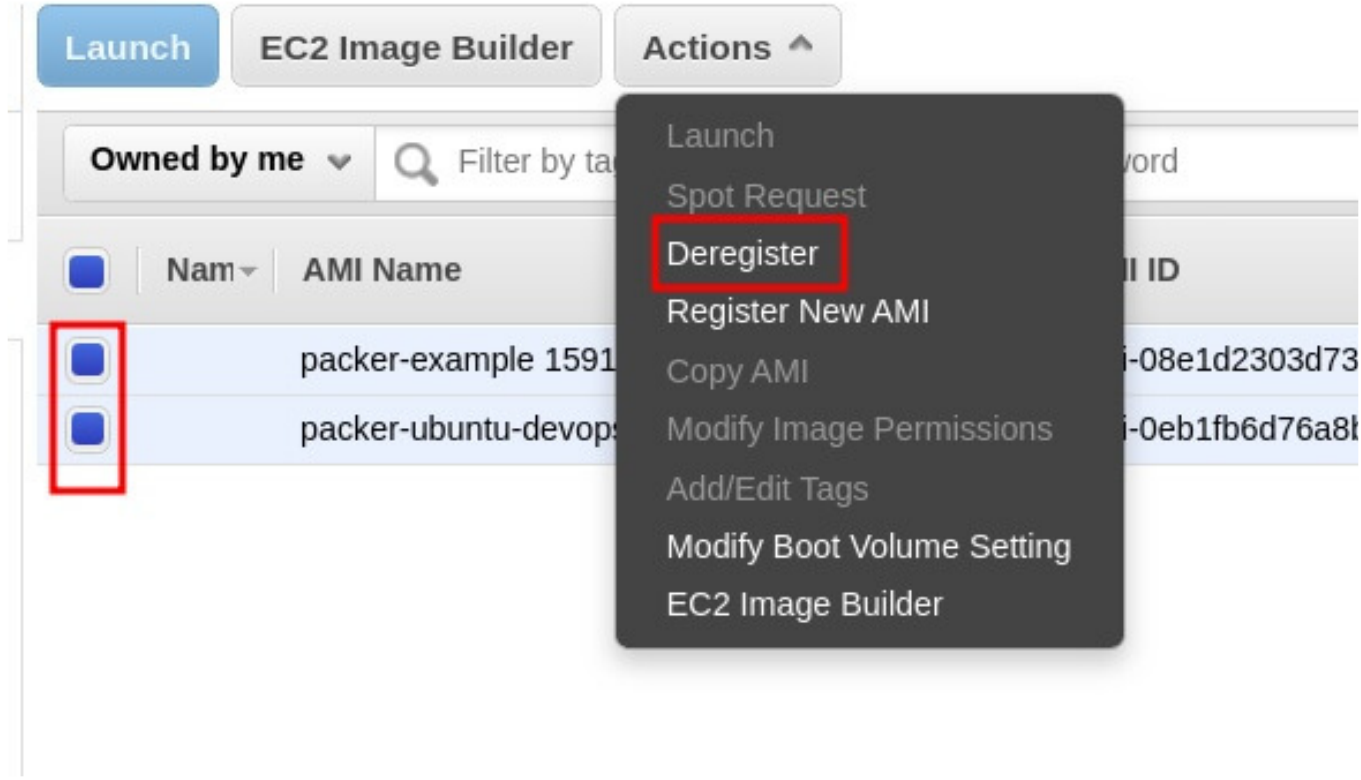
Supprimer votre image

À la fin de l'exécution de votre commande `packer build`, Packer génère les **artefacts** depuis votre template.

Information

Les artefacts sont les résultats d'une génération, comme le cas d'une AMI chez AWS ou un ensemble de fichiers comme pour une machine virtuelle VMware.

Dans le cas d'AWS, les artefacts sont stockés dans un bucket S3 entièrement managé par AWS. Pour supprimer votre AMI, rendez-vous dans la page gestion des AMIs et supprimez l'image ou les images concernée(s), exemple :



Information

Si vous souhaitez garder votre AMI vous serez alors facturé environ $0.01 par mois.

Conclusion

Ceci a été une introduction sur la découverte et l'utilisation de l'outil packer, ça reste un outil très simple à utiliser et nous aurons l'occasion de l'utiliser sur d'autres projets dans de futurs articles.