Projet Cloud : Présentation de terraform

École : Ecole Nationale Supérieure Polytechnique de Yaoundé

Encadreur : Jean-Marc Menaud - Professor - IMT-Atlantique

Participants :

1. **MENRA WEDWANG ROMIAL**
2. **NCHOUWET NGANE LOIC**
3. **KAMGA DJIDJOU WILFRIED JUNIOR**
4. **AMOU'OU AMOU'OU GEORGES JUNIOR**
5. **FOFOU NOBOSSE**

Infrastructure as code (IaC)

Naissance du besoin

Traditionnellement, la gestion du cycle de vie des infrastructures était un processus manuel qui a souvent entraîné des incohérences environnementales. Par exemple, les environnements de test qui ne sont pas entièrement compatibles avec l'environnement de production sont un problème très courant chez les entreprises traditionnelles. Ce type d'incohérences environnementales crée souvent beaucoup de frictions pour la livraison des applications et des solutions, ce qui ralentit ou parfois bloque le rythme de l'innovation dans une organisation numérique.

D'ailleurs, on peut sûrement se le dire, l'automatisation de bout en bout d'un processus de développement d'applications est devenue courante. Bien que certaines parties du ce processus de tests étaient automatisées depuis un certain temps maintenant. Ce n'est pas forcément le cas pour la gestion et le provisionnement d'une infrastructure informatique qui était entre autres le seul domaine qui était un peu ignoré à l'ère de l'automatisation.

Certes, il existe de nombreux outils avec des tableaux de bord sophistiqués qui vous fournissent des mises à jour d'intégrité et effectuent la partie d'approvisionnement de l'infrastructure. Mais ce n'est qu'une partie de l'automatisation de l'infrastructure, ce qui signifie simplement répliquer les étapes plusieurs fois et reproduire plusieurs fois les sorties déjà produites sur plusieurs serveurs auparavant.

En réalité, ce n'est qu'une réplication pour réduire l'intervention humaine et le temps nécessaire pour le faire. Vous le savez sûrement mais le type d'application produit de nos jours et les environnements informatiques de notre nouvelle ère, sont plus que jamais dynamiques qu'antérieurement. De ce fait, vous avez donc besoin de quelque chose qui non seulement évoluera plus rapidement mais qui maintiendra également la cohérence de vos infrastructures, et l'Infrastructure as Code pourrait bien être cette chose.

Qu'est-ce que l'Infrastructure as Code (IaC) ?

Infrastructure as Code (IaC) consiste à remplacer les processus manuels et les procédures d'exploitation standard pour configurer les périphériques matériels et les systèmes d'exploitation par du code qui **gérera et fournira automatiquement votre infrastructure**. Dans l'IaC, vous pouvez configurer et déployer ces composants d'infrastructure plus rapidement avec cohérence en les traitant comme si c'était une application.

Ainsi, chaque fois que vous devez configurer une infrastructure, vous n'avez pas besoin d'aller voir les administrateurs système, de formuler une demande, de créer un ticket et d'attendre qu'ils y participent. Au lieu de cela, vos développeurs et vos équipes d'exploitation peuvent facilement le faire en utilisant votre code.

Pourquoi IaC?

Comme tout processus d'automatisation, les avantages évidents ici sont le **coût**, la **simplicité** et la **rapidité** . Parce qu'avec IaC, vos administrateurs système et vos développeurs peuvent **travailler plus efficacement** sur des tâches plus prioritaires. Vous pouvez répondre à leurs besoins en infrastructure de manière plus dynamique que jamais et vous pouvez littéralement faire tourner une configuration entière en exécutant un seul morceau de code. Augmentant ainsi la vitesse de vos processus globaux, réduisant ainsi les efforts manuels et le temps nécessaire pour répondre aux besoins en infrastructure.

Avec l'IaC, vos développeurs peuvent travailler de manière plus productive en raison de l'efficacité accrue et de la dépendance réduite vis-à-vis d'autres équipes. Ils peuvent lancer leurs propres environnements en fonction de leurs besoins. Avec un seul code, ils peuvent prendre en charge un certain nombre d'étapes qui seraient là sans l'automatisation, ainsi vous contribuez à garder vos environnements propres et organisés.

De plus, il s'intègre parfaitement à la philosophie DevOps, qui je vous rappelle est un changement culturel qui se concentre immédiatement sur la maximisation de la valeur commerciale en optant pour une meilleure communication, collaboration et rétroaction au sein et entre les équipes de développement et d'exploitation. Ainsi l'Infrastructure as Code (IaC) est un élément-clé de la philosophie DevOps avec des avantages pour les équipes de développement et d'exploitation. Dans l'IaC, le cycle de vie complet de l'infrastructure, y compris l'orchestration, l'approvisionnement, la configuration, la surveillance, l'auto-réparation peut être gérée de manière automatisée.

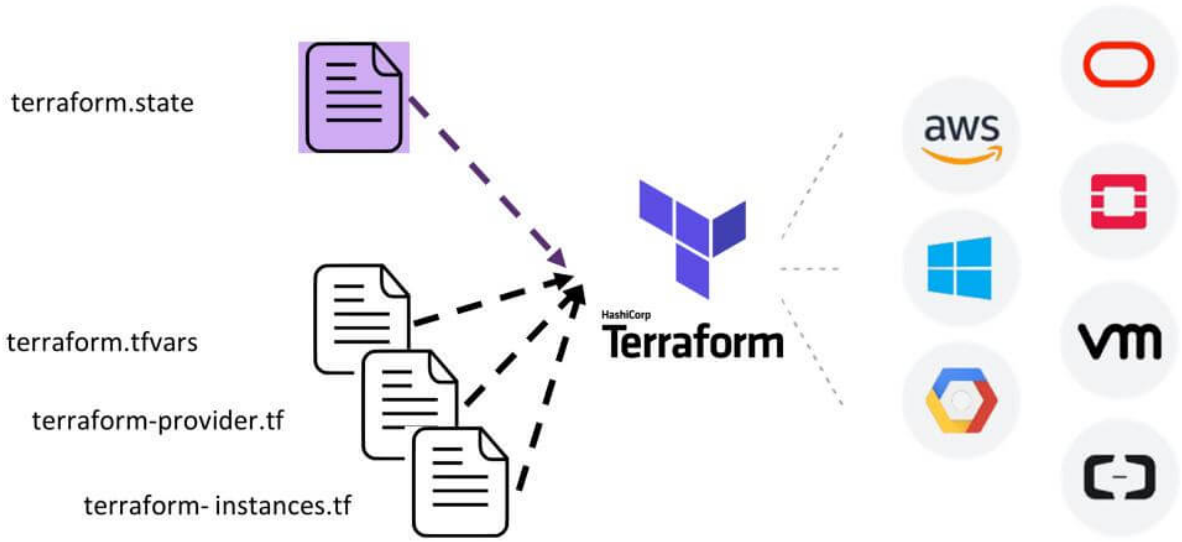
Un autre avantage est sur le plan de la sécurité, avec l'IaC vous pouvez documenter et **suivre les modifications apportées** à votre code avec un système de gestion de code source tel que git, et par la même occasion vous versionnez également les modifications apportées à votre infrastructure globale. Sans oublier que cela vous aidera à **maintenir la cohérence** entre les procédures de déploiement. De plus, avec l'IaC, l'intervention humaine est minimale, ce qui signifie une réduction des erreurs humaines et une uniformité et une standardisation accrues. En conséquence, les problèmes tels que la compatibilité et son fonctionnement sur ma machine sont réduits au minimum.

Terraform

C'est quoi ?

Vous l'aurez compris nous étudierons tout au long de ce cours l'outil Terraform, qui est un outil permettant de créer, modifier et versionner une infrastructure de manière sûre et efficace. Terraform peut **gérer différents fournisseurs d'infrastructure**, allant des Cloud Providers (AWS, Azure, GcP, Alibaba Cloud, etc ...) jusqu'aux solutions internes personnalisées (VmWare, Kubernetes, etc ...).

Comment ça fonctionne ?



Vous codez vos fichiers de configuration qui décrivent à Terraform les composants nécessaires pour exécuter une seule ressource comme par exemple la création d'une machine virtuelle ou l'ensemble de votre datacenter. Ensuite, il générera plan d'exécution décrivant ce qu'il fera pour atteindre l'état que vous souhaitez, puis l'exécute pour construire l'infrastructure décrite. Au fur et à mesure que la configuration change, il sera en mesure de déterminer ce qui a changé et de créer des plans d'exécution incrémentiels qui peuvent être appliqués.

L'infrastructure que Terraform peut gérer comprend des composants de bas niveau tel que les instances de calcul, le stockage et la mise en réseau, ainsi que des composants de haut niveau tels que les entrées DNS, les fonctionnalités SaaS, etc. Nous aurons l'occasion d'étudier tous ces aspects dans les futurs chapitres.

Conclusion

Maintenant que vous connaissez les avantages associés à l'Infrastructure as Code, et que je vous ai introduit et présenté l'outil Terraform, vous voudrez peut-être commencer à l'utiliser dans votre organisation ou pour vos projets personnels. C'est pour cela, que sur le prochain chapitre nous commencerons directement par la configuration notre environnement Terraform sur Linux et Windows.

Installer et configurer votre environnement Terraform

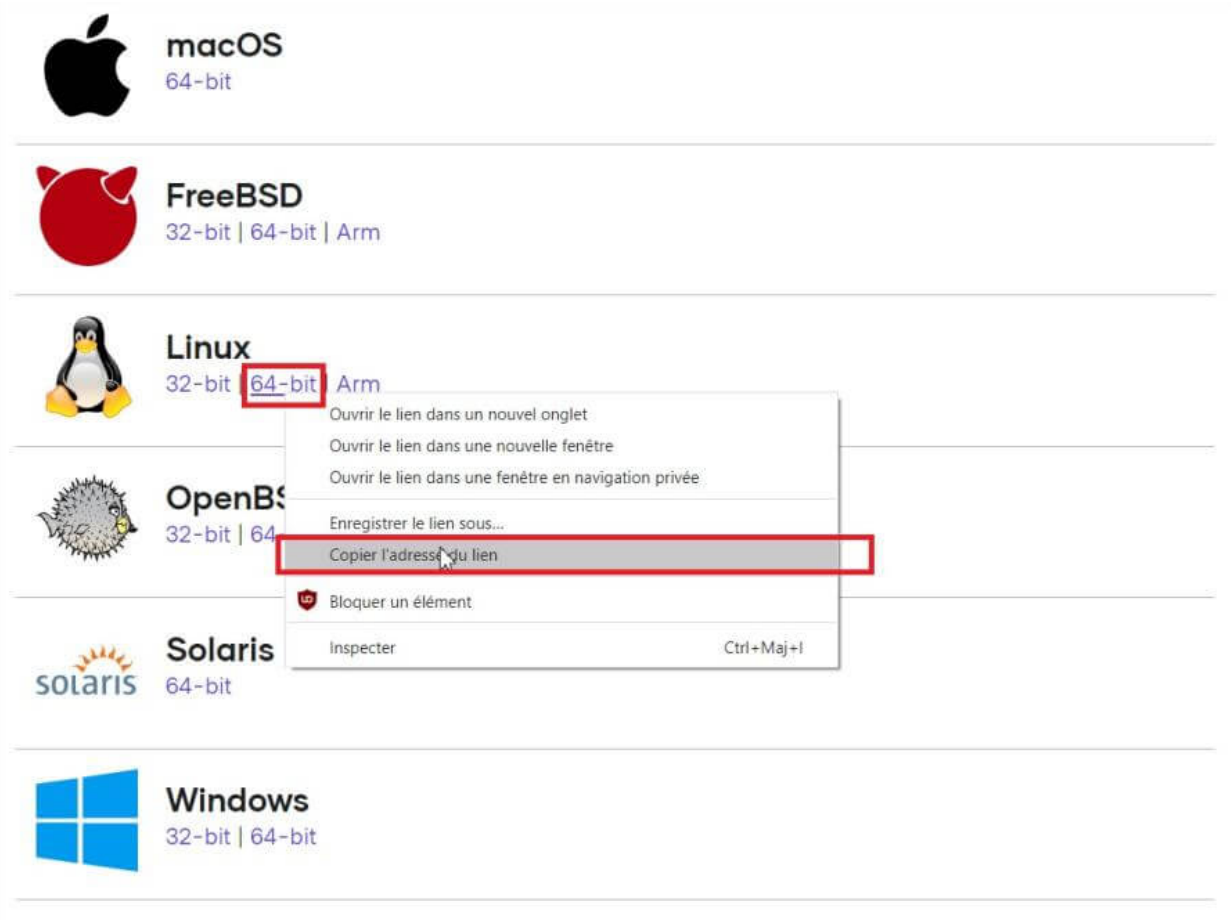
Introduction

Dans ce chapitre, nous découvrirons **comment configurer notre environnement Terraform**. L'avantage, c'est qu'il est distribué sous forme de **package binaire** différentes plates-formes et architectures prises en charge, donc pas besoin de le compiler. Il suffit donc d'installer le binaire Terraform compressé, le décompresser et le déplacer vers un répertoire inclus dans votre système **PATH**, simple non ?

Installation sous Linux

Linux

Dans cette partie, nous verrons **comment procéder à une installation de Terraform sur une machine Linux**. D'abord, rendez-vous sur la page d'installation de Terraform et trouvez le package approprié pour votre système et téléchargez-le, dans mon cas je suis sur la distribution Ubuntu en version 19.10 en architecture 64 bits, donc je dois prendre en compte l'archive zip suivante, en copiant son url :



Ensuite, vous devrez mettre à niveau votre système et vos packages :

```
sudo apt update -y && sudo apt upgrade -yCopier
```

Installez ensuite le package wget et unzip s'ils ne sont pas déjà installés:

```
sudo apt install -y wget unzipCopier
```

Nous sommes maintenant prêts à **télécharger le fichier zip de Terraform pour Linux** depuis le site officiel. Au moment de la rédaction de cet article, la version actuelle de Terraform était la 0.12.24, ce qui nous donne la commande suivante :

```
wget
https://releases.hashicorp.com/terraform/0.12.24/terraform_0.12.24_linux_amd64.zipCopie
r
```

Après avoir téléchargé Terraform, nous allons décompresser l'archive dans le dossier */usr/local/bin/* afin que n'importe quel utilisateur normal puisse exécuter le programme terraform :

```
sudo unzip ./terraform_0.12.24_linux_amd64.zip -d /usr/local/binCopier
```

Enfin, Il ne reste plus qu'à **vérifier si terraform est installé avec succès**, en tapant la commande suivante :

```
terraform -vCopier
```

Résultat :

```
Terraform v0.12.24
```

Cette étape n'est pas vraiment obligatoire, mais vous pouvez déjà commencer à **générer une paire de clés ssh** afin de vous connecter à vos machines virtuelles via le protocole ssh. Pour ce faire, lancez la commande suivante en spécifiant le chemin de votre paire de clés sans indiquer de phrase secrète :

```
Explainssh-keygen -t rsa

Generating public/private rsa key pair.
Enter file in which to save the key (/home/hatim/.ssh/id_rsa): ~/terraform/terraform
Enter passphrase (empty for no passphrase):
Enter same passphrase again: Copier
```

Après le lancement de votre commande vous aurez à votre disposition une clé publique nommée *terraform.pub* et une clé privée nommée *terraform* :

```
ls ~/terraform/Copier
```

Résultat :

```
terraform.pub  terraform
```

Ensuite, il faut penser à sécuriser votre clé privée en mettant à jour ses autorisations avec la commande suivante :

```
chmod 400 terraformCopier
```

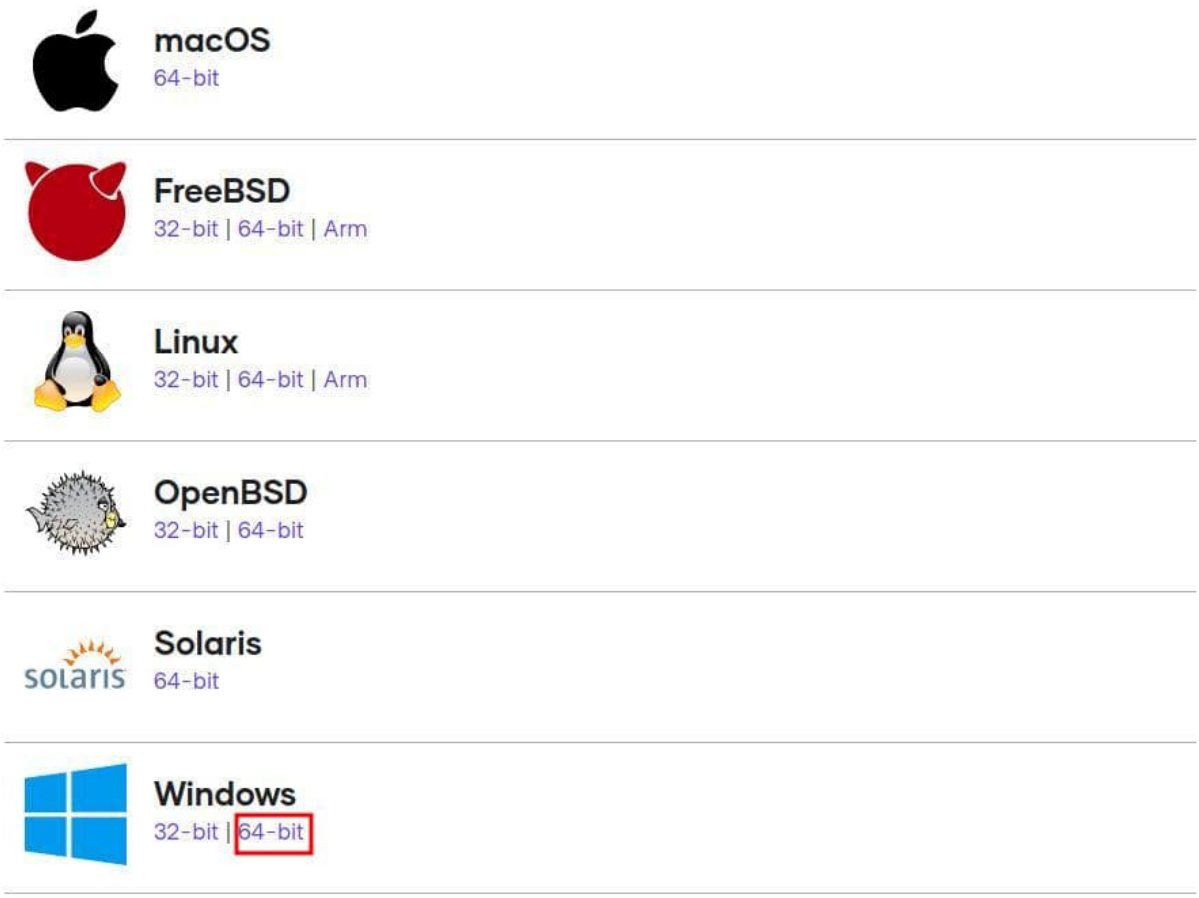
Voilà, l'installation sur Linux est désormais terminée, dans la partie suivante on s'attaquera à l'os Windows.

Installation sous Windows

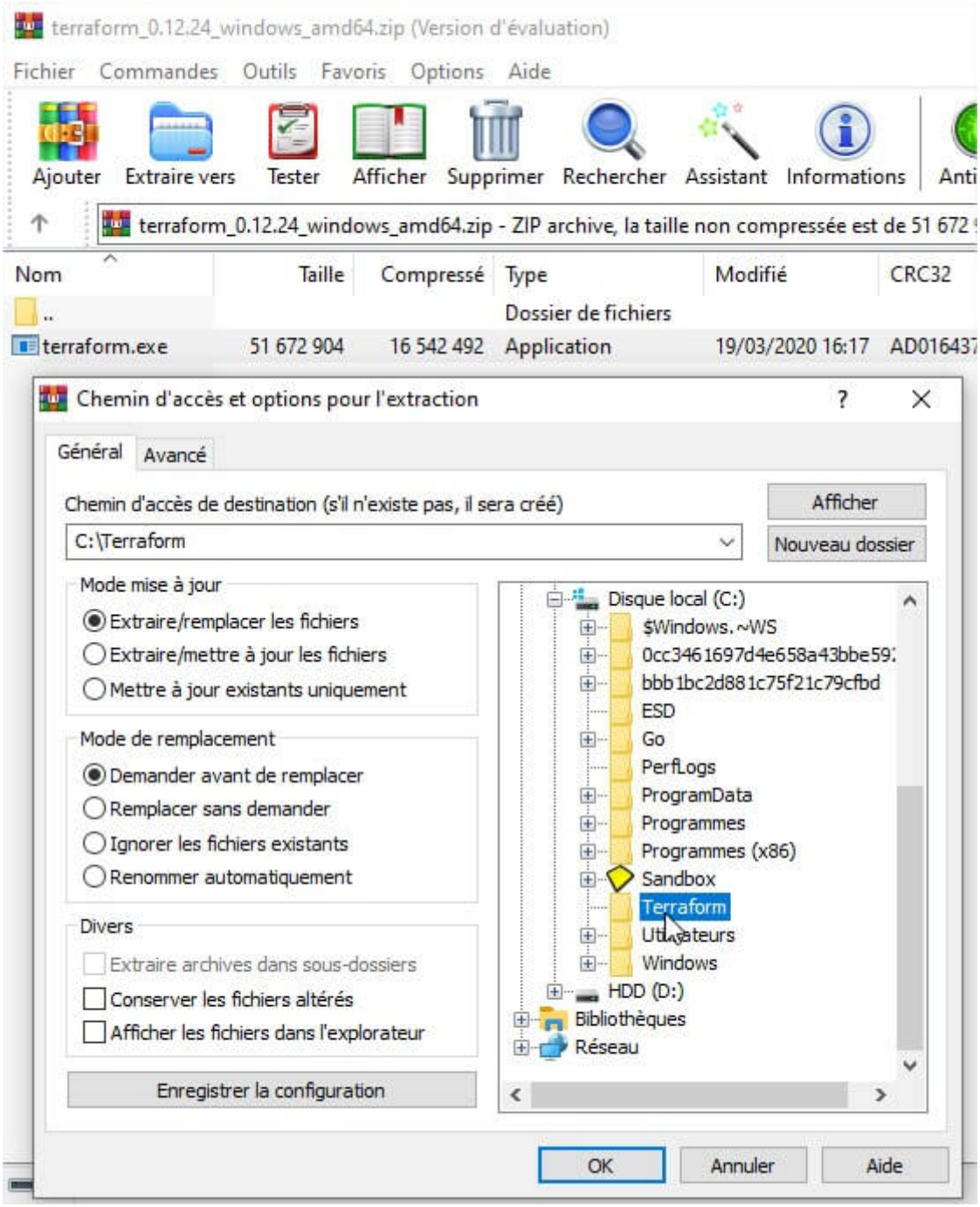
Cette fois si je vais vous montrer comment **installer terraForm sur Windows sous forme de deux méthodes**, une installation manuelle et une installation avec le gestionnaire de paquets windows Chocolatey.

Installation manuelle

Commencez par télécharger la version appropriée de Terraform depuis la page d'installation officielle. Dans mon cas, c'est la version Windows 64 bits :

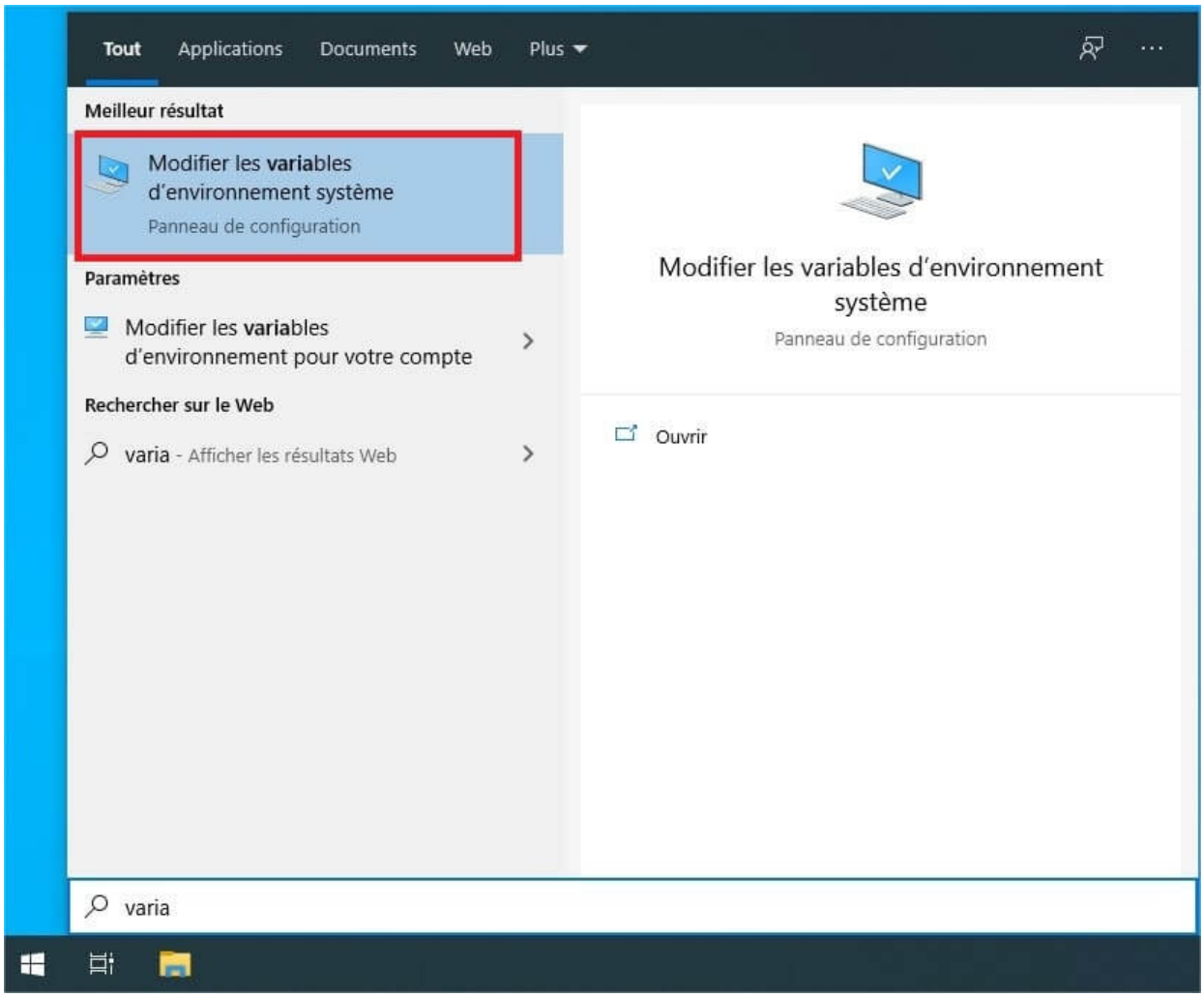


Une fois le téléchargement terminé, créez un dossier sur votre lecteur *C:* où vous pourrez placer l'exécutable Terraform. Allez ensuite trouver le binaire Terraform dans l'explorateur de fichiers et extrayez ce fichier zip dans le dossier que vous avez créé précédemment :

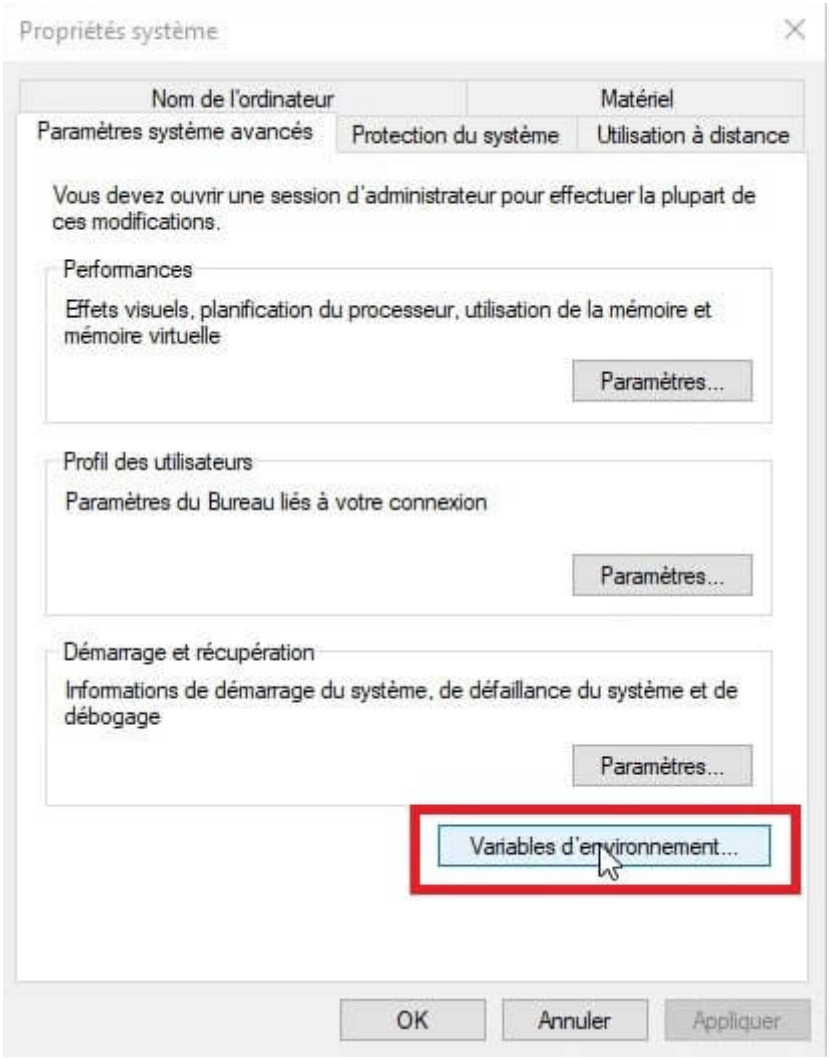


Maintenant il suffit de **rajouter votre exécutable Terraform à votre variable d'environnement nommée *PATH***, c'est la variable système utilisée par Windows pour localiser les fichiers exécutables, ainsi vous pouvez executer le programme Terraform depuis n'importe où en ligne de commande. Pour ce faire,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

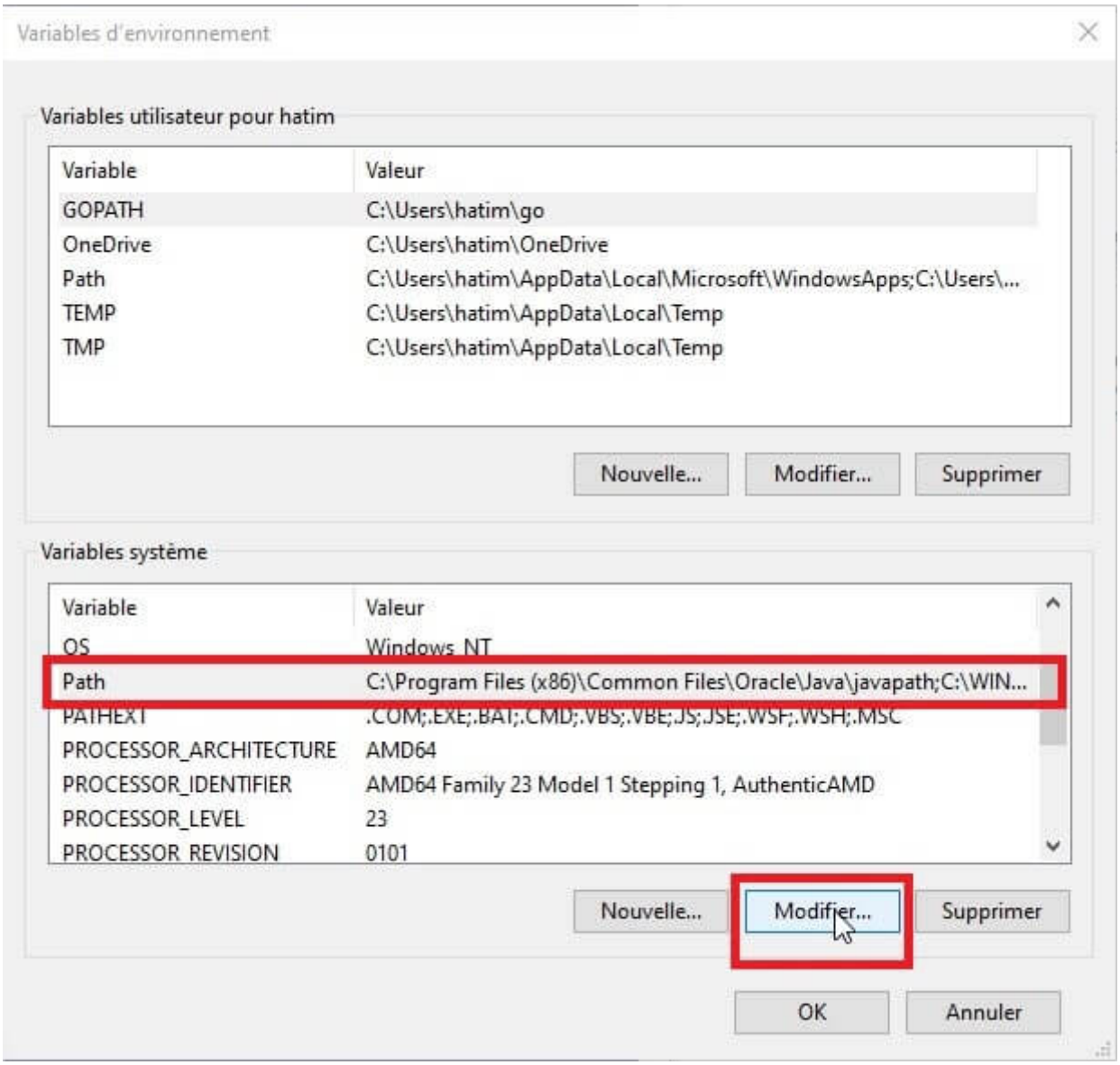
Commencez par ouvrir votre menu Démarrer et tapez "environnement" et la première chose qui apparaît devrait être "Modifier les variables d'environnement système".



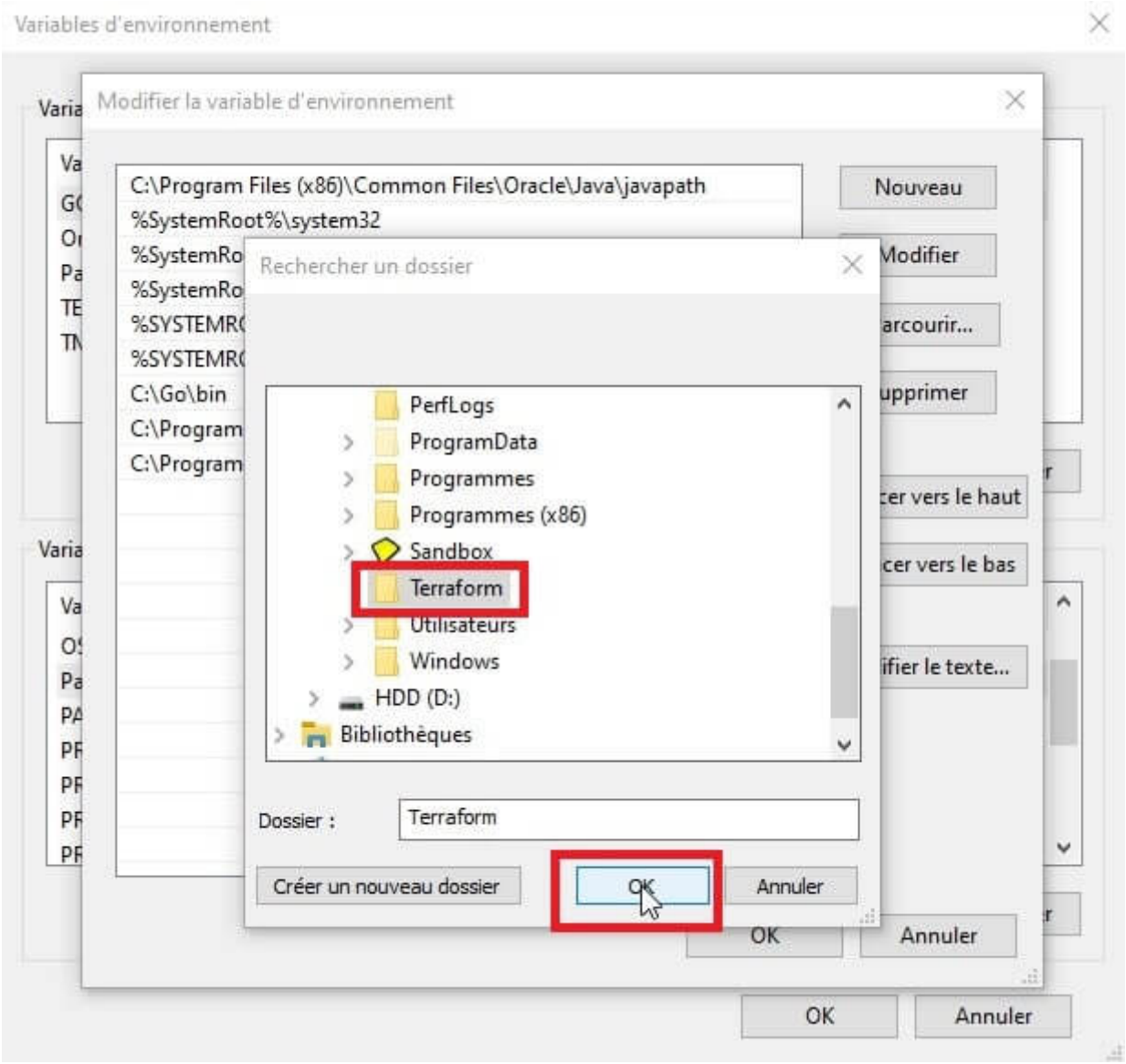
Cliquez dessus et vous devriez voir cette fenêtre, cliquez ensuite sur le bouton "Variables d'environnement" :



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 *terraform.exe* :



Comme pour l'installation sous Linux, il ne reste plus qu'à **vérifier si terraform est installé avec succès sur votre machine Windows**, ouvrez votre powershell et lancez la commande suivante :

```
terraform --versionCopier
```

Résultat :

```
Terraform v0.12.24
```

Installation avec le gestionnaire de paquets Chocolatey

Dans cette méthode nous utiliserons plutôt un gestionnaire de packages pour Windows. Il existe quelques gestionnaires de packages que vous pouvez utiliser pour installer Terraform sur Windows. Pour Windows, mon préféré est Chocolatey. Il rend l'installation, la suppression et la mise à jour (comme les gestionnaires de paquets sous Linux) de logiciels aussi simples qu'une commande sur une seule ligne, et Terraform n'y fait pas exception.

Pour installer Terraform avec Chocolatey, procédez comme suit :

1. Ouvrez une invite CMD / PowerShell en tant qu'administrateur et installez Chocolatey à l'aide de la commande de leur page d'installation.
2. Une fois cette opération terminée, exécutez la commande suivante avec l'option -y afin d'activer l'acceptation automatique :

```
choco install -y terraformCopier
```

Enfin, vérifiez que l'installation a réussi en entrant la commande suivante :

```
terraform --versionCopier
```

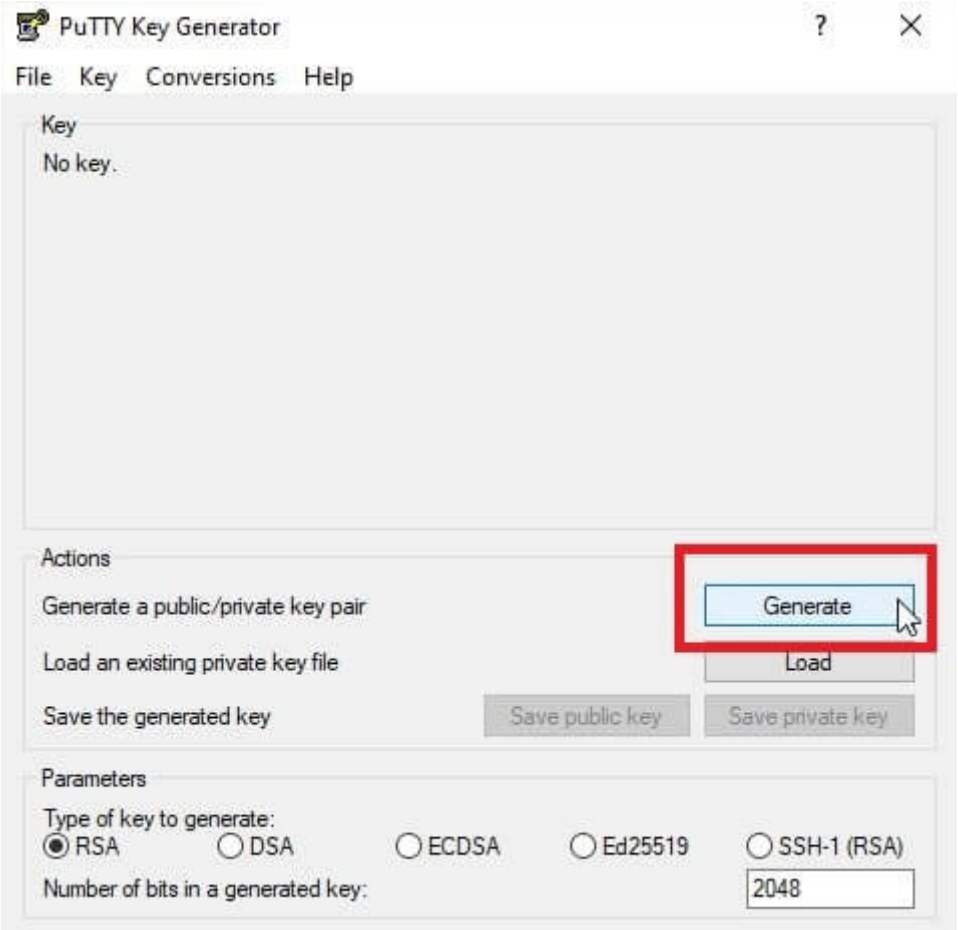
Résultat :

```
Terraform v0.12.24
```

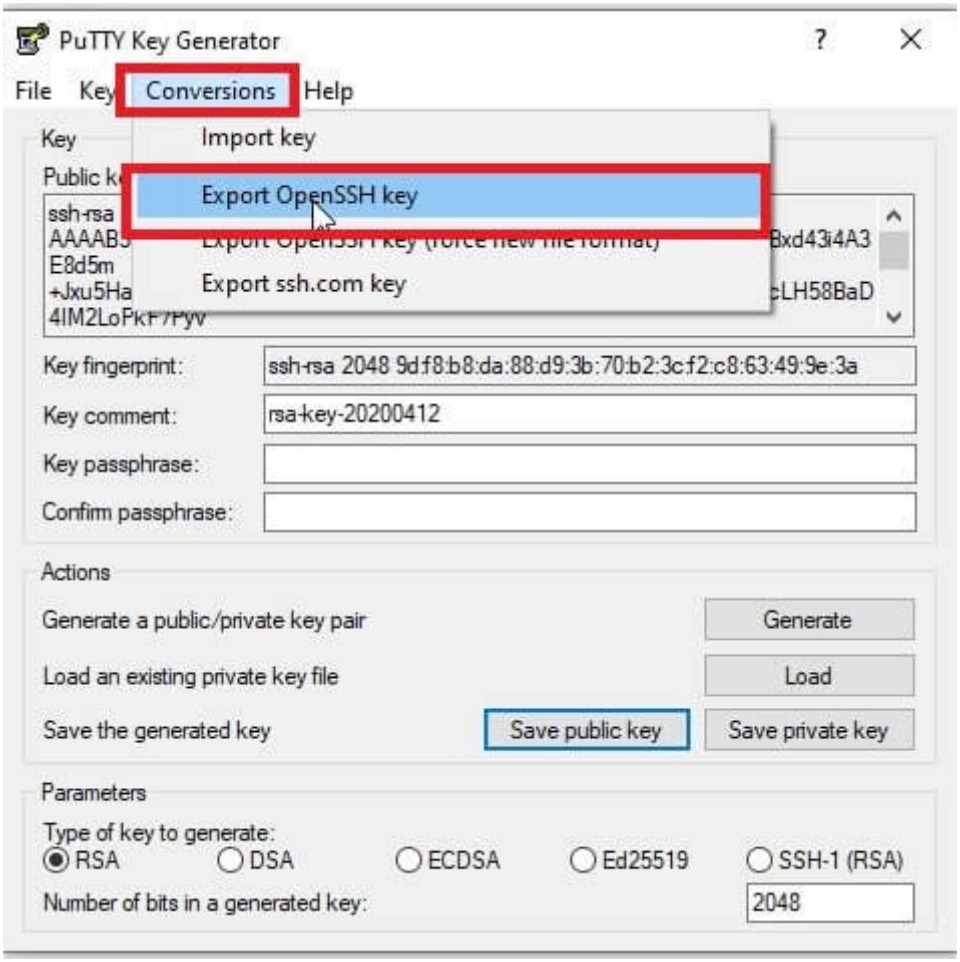
Génération d'une paire de clés SSH sous Windows

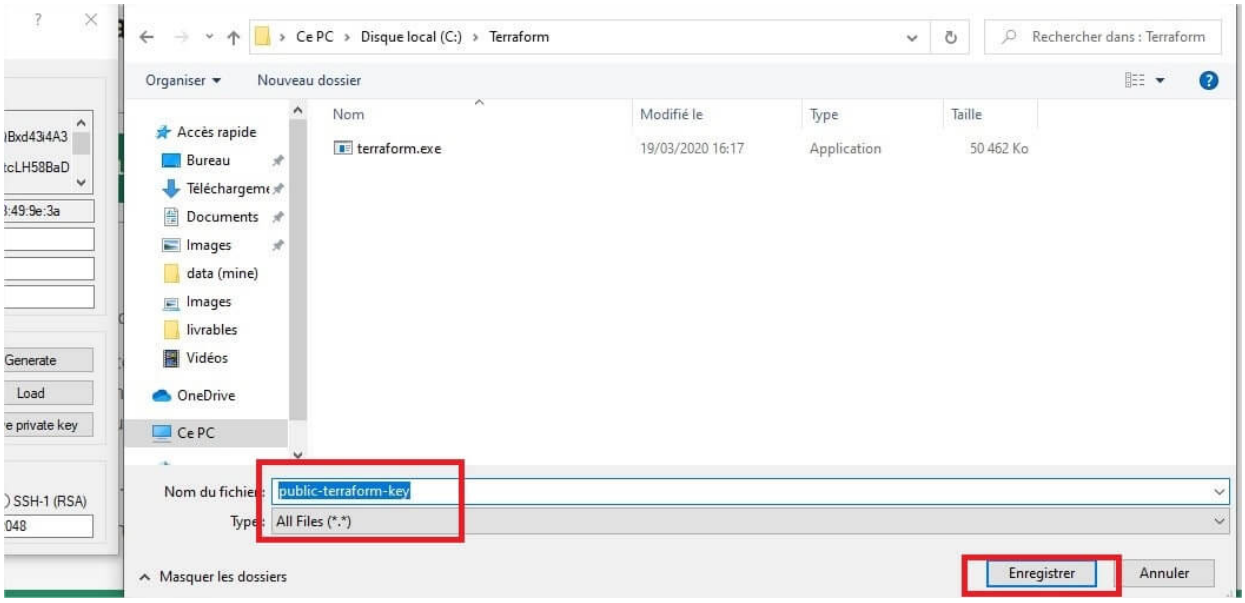
Comme pour Linux, vous pouvez déjà commencer à générer une paire de clés SSH qui sera utilisée pour vous permettre d'accéder en toute sécurité aux instances créées dans des sous-réseaux publics. La génération d'une paire de clés sous Windows est un peu plus fastidieuse que sur Windows. On utilisera l'outil PuTTYgen, vous retrouverez l'exécutable sur la page d'installation de PuTTYgen.

Une fois l'installation terminée, démarrez PuTTYgen et acceptez le type de clé par défaut SSH-2 RSA et définissez le nombre de bits d'une clé générée sur 2048 s'il n'est pas déjà défini. Ensuite, cliquez sur le bouton "Générer", puis déplacez votre souris dans la zone vide pour générer une clé aléatoire :

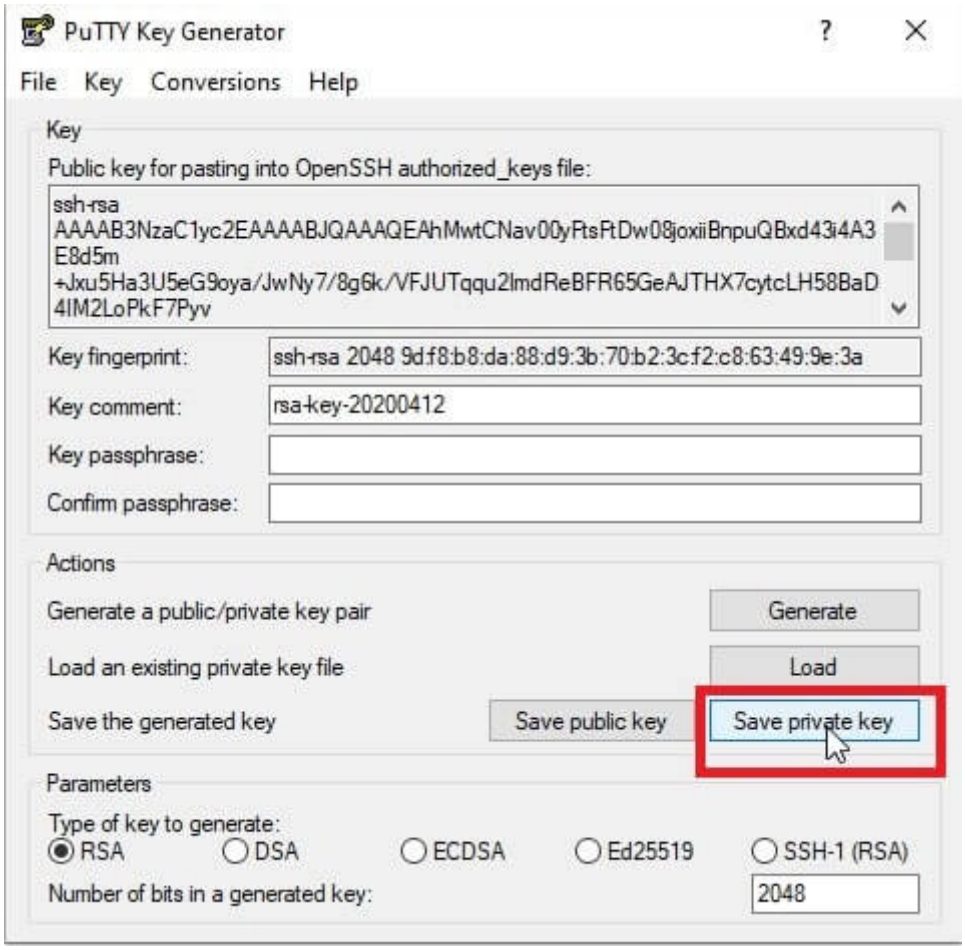


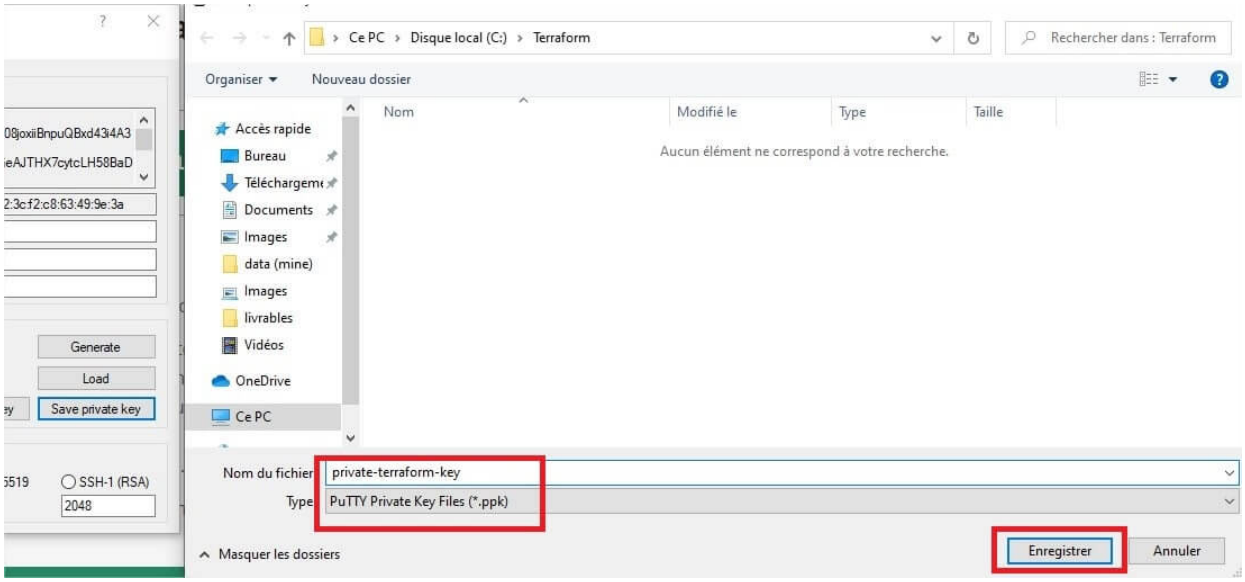
Ensuite, pour enregistrer la clé publique, cliquez sur "Conversions" sur le menu tout haut et ensuite cliquez sur le sous-menu "Export OpenSSH Key". Vous serez invité à confirmer que vous souhaitez enregistrer la clé sans mot de passe, confirmez en cliquant sur "Oui". Vous pouvez stocker cette clé dans n'importe quel emplacement, mais pour plus de simplicité, stockez-la dans le même dossier où se trouve votre exécutable Terraform :





Enfin, Cliquez sur "Save private key", confirmez que vous souhaitez enregistrer la clé sans mot de passe en cliquant sur "Oui". . Enregistrez là dans le même emplacement que votre clé publique sous le format "ppk" (PuTTY Private Key) :


PuTTY Key Generator
File Key Conversions Help
Key
Public key for pasting into OpenSSH authorized_keys file:
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAhMwtCNav00yPtsPtDw08joxiiBnpuQBxd43i4A3
E8d5m
+Jxu5Ha3U5eG9oya/JwNy7/8g6k/VFJUTqqu2lmdReBFR65GeAJTHX7cytcLH58BaD
4IM2LoPkF7Pyv
Key fingerprint:
ssh-rsa 2048 9d:f8:b8:da:88:d9:3b:70:b2:3c:f2:c8:63:49:9e:3a
Key comment:
rsa-key-20200412
Key passphrase:
Confirm passphrase:
Actions
Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key
Parameters
Type of key to generate:
RSA
DSA
ECDSA
Ed25519
SSH-1 (RSA)
Number of bits in a generated key:
2048


Ce PC › Disque local (C:) › Terraform
Rechercher dans : Terraform
Organiser
Nouveau dossier
Accès rapide
Bureau
Documents
Images
data (mine)
Images
livrables
Vidéos
OneDrive
Ce PC
Nom
Modifié le
Type
Taille
Aucun élément ne correspond à votre recherche.
Nom du fichier
private-terraform-key
Type
PuTTY Private Key Files (*.ppk)
Masquer les dossiers
Enregistrer
Annuler
Generate
Load
Save private key
SSH-1 (RSA)
2048

Conclusion

Vous connaissez maintenant les **différentes façons d'installer et d'exécuter Terraform sous Windows et Linux**. Nous sommes donc prêts à utiliser l'outil Terraform pour automatiser nos tâches !