Mamadou SALL

Développeur Fullstack & Ingénieur Devops

Vagrant

I. Définition

Vagrant est un outil permettant de créer des environnements de développement complets. Avec un flux de travail facile à utiliser et une concentration sur l'automatisation, Vagrant réduit le temps de configuration de l'environnement de développement, augmente la parité développement/production et fait de l'excuse « ça marche sur ma machine » une relique du passé.

Vagrant est conçu pour tous comme le moyen le plus simple et le plus rapide de créer un environnement virtualisé

Vagrant est un outil open-source qui facilite la gestion et la création d'environnements de développement virtuels. Il a été développé par HashiCorp. Voici quelques-uns de ses avantages :

1. **Environnements reproductibles :** Vagrant permet de définir l'environnement de développement dans un fichier de configuration, ce qui le rend reproductible. Vous pouvez partager ce fichier avec d'autres membres de l'équipe, garantissant que tout le monde travaille dans le même environnement.
2. **Gestion simplifiée des machines virtuelles :** Vagrant automatise la création et la gestion de machines virtuelles. Vous pouvez utiliser des boîtes (boxes) préconfigurées pour différentes plates-formes, ou créer votre propre configuration.
3. **Isolation :** Les environnements de développement créés par Vagrant sont isolés du système hôte, ce qui évite les conflits entre logiciels, versions de bibliothèques, etc. Cela aide à éviter les problèmes liés à la compatibilité et à garantir une certaine cohérence.
4. **Support multi-fournisseurs :** Vagrant prend en charge différents fournisseurs de machines virtuelles, tels que VirtualBox, VMware, Hyper-V, etc. Cela signifie que vous pouvez utiliser Vagrant avec différents hyperviseurs selon vos besoins.
5. **Facilité de collaboration :** Comme la configuration de l'environnement est définie dans un fichier, il est facile de partager le code et de collaborer sur des projets sans se soucier des différences d'environnement entre les développeurs.
6. **Économie de ressources :** Vagrant permet de provisionner des machines virtuelles uniquement lorsque nécessaire, ce qui contribue à économiser des ressources système en évitant de maintenir des machines virtuelles actives en permanence.
7. **Intégration avec la gestion de configuration :** Vagrant peut être intégré avec des outils de gestion de configuration tels que Ansible, Chef, ou Puppet, facilitant ainsi l'automatisation et la configuration des logiciels sur les machines virtuelles.

En résumé, Vagrant offre un moyen pratique de gérer des environnements de développement, en fournissant une solution reproductible, isolée, et facile à partager pour les équipes de développement.

II. Commandes de base

vagrant init

La commande `vagrant init ubuntu/trusty64` est utilisée pour initialiser un nouveau projet Vagrant en se basant sur une box spécifique. Voici une explication détaillée de cette commande :

1. `vagrant init` **:** Cette partie de la commande signifie que vous voulez initialiser un nouveau projet Vagrant.
2. `ubuntu/trusty64` **:** Cela spécifie la box que vous souhaitez utiliser comme base pour votre environnement Vagrant. En l'occurrence, `ubuntu/trusty64` fait référence à une box Ubuntu basée sur la version 14.04 (Trusty Tahr) de cette distribution Linux. Les "64" indiquent qu'il s'agit d'une version 64 bits.

Lorsque vous exécutez cette commande, Vagrant crée un fichier de configuration appelé `Vagrantfile` dans le répertoire courant. Ce fichier contient la configuration de base de votre environnement Vagrant, y compris les paramètres spécifiés lors de l'initialisation.

Le `Vagrantfile` généré contiendra probablement des lignes telles que celles-ci :

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/trusty64"
end
```

Cela indique que votre machine virtuelle Vagrant sera basée sur la box Ubuntu Trusty 64 bits.

Après avoir initialisé votre projet avec `vagrant init`, vous pouvez personnaliser davantage le `Vagrantfile` en fonction de vos besoins. Par exemple, vous pourriez définir la quantité de mémoire allouée à la machine virtuelle, configurer des ports de redirection, spécifier des scripts de provisionnement, etc. Une fois le `Vagrantfile` configuré, vous pouvez utiliser la commande `vagrant up` pour démarrer la machine virtuelle basée sur la configuration définie dans ce fichier.

vagrant ssh

La commande `vagrant ssh` est utilisée pour établir une connexion SSH avec la machine virtuelle Vagrant que vous avez créée et lancée à l'aide de Vagrant. Voici comment cela fonctionne :

1. **Assurez-vous que la machine virtuelle est en cours d'exécution :** Avant d'utiliser `vagrant ssh`, assurez-vous que votre machine virtuelle est démarrée. Vous pouvez le faire en utilisant la commande `vagrant up`.
2. **Exécutez `vagrant ssh` :** Ouvrez votre terminal, accédez au répertoire où se trouve votre fichier `Vagrantfile` (le fichier de configuration de votre machine virtuelle) et exécutez la commande `vagrant ssh`.

   ```
   vagrant ssh
   ```

3. **Connexion SSH à la machine virtuelle :** Cette commande établit une connexion SSH avec la machine virtuelle. Vous serez connecté en tant qu'utilisateur configuré dans votre `Vagrantfile`. Par défaut, il s'agit souvent de l'utilisateur `vagrant`.

Une fois connecté, vous pouvez utiliser la machine virtuelle via la ligne de commande comme si vous étiez physiquement connecté à la machine. Vous pouvez exécuter des commandes, installer des logiciels, etc.

Lorsque vous avez terminé, vous pouvez quitter la session SSH en utilisant la commande `exit`. Cela vous ramènera à votre terminal local.

```
exit
```

La commande `vagrant ssh` est pratique car elle simplifie l'accès à la machine virtuelle sans avoir à se soucier des détails de l'adresse IP, du port SSH, etc. Elle utilise la configuration spécifiée dans le `Vagrantfile` pour établir la connexion.

vagrant halt

La commande `vagrant halt` est utilisée pour arrêter (mettre en veille) la machine virtuelle Vagrant en cours d'exécution. Elle fonctionne de manière similaire à l'extinction d'un ordinateur.

Pour utiliser la commande `vagrant halt`, suivez ces étapes :

1. Ouvrez votre terminal.

2. Accédez au répertoire où se trouve votre fichier `Vagrantfile`.
3. Exécutez la commande suivante :

```
vagrant halt
```

Cette commande enverra un signal d'arrêt à la machine virtuelle, la mettant en veille. Vous pouvez utiliser `vagrant up` pour redémarrer la machine virtuelle ultérieurement.

Assurez-vous de sauvegarder votre travail avant d'utiliser `vagrant halt`, car cela éteindra la machine virtuelle et tous les travaux non sauvegardés seront perdus.

vagrant destroy

La commande `vagrant destroy` est utilisée pour supprimer complètement une machine virtuelle créée avec Vagrant. Lorsque vous exécutez cette commande, Vagrant arrête la machine virtuelle s'il est en cours d'exécution, puis supprime tous les fichiers associés à la machine virtuelle, y compris le disque virtuel, libérant ainsi l'espace utilisé sur votre système.

Voici comment utiliser `vagrant destroy` :

1. Ouvrez votre terminal.
2. Accédez au répertoire où se trouve votre fichier `Vagrantfile`.
3. Exécutez la commande suivante :

```
vagrant destroy
```

Vagrant vous demandera une confirmation avant de détruire la machine virtuelle. Vous pouvez également ajouter l'option `-f` pour supprimer la machine virtuelle sans confirmation.

```
vagrant destroy -f
```

Assurez-vous d'avoir sauvegardé toutes les données importantes de la machine virtuelle avant d'exécuter `vagrant destroy`, car cette action est irréversible et entraînera la perte de toutes les données non sauvegardées.

Après avoir détruit la machine virtuelle, vous pouvez toujours recréer une nouvelle machine virtuelle en utilisant `vagrant up` lorsque vous en avez besoin.

III. Vagrant boxes

Les boîtes sont le format de package pour les environnements Vagrant. Vous spécifiez un environnement de boîte et des configurations d'exploitation dans votre Vagrantfile . Vous pouvez utiliser une boîte sur n'importe quelle plate-forme prise en charge pour afficher des environnements de travail identiques. Pour permettre aux équipes d'utiliser et de gérer les mêmes boîtiers, les versions sont prises en charge .

Le moyen le plus rapide de commencer consiste à sélectionner un environnement de boîte prédéfini dans le catalogue accessible au public sur Vagrant Cloud . Vous pouvez également ajouter et partager vos propres boîtes personnalisées sur Vagrant Cloud.

La vagrant Boxe c’est comme notre image notre systeme téléchargeable qui contient notre systèmes d’exploitation et le nécessaire pour le déploiement de la machine virtuelle.

1. Les composant d’un vagrant boxes

Un `.box` fichier Vagrant est une archive tar (`tar` , `tar.gz` , `zip`) qui contient toutes les informations permettant à un fournisseur de lancer une machine Vagrant.

Il y a quatre composants différents qui composent une boîte :

- **Artefacts de VM (obligatoire)** : il s'agit de l'image de la VM et des autres artefacts au format accepté par le fournisseur auquel la boîte est destinée. Par exemple, une boîte ciblant le fournisseur VirtualBox peut contenir un `.ofv` fichier et quelques `.vmdk` fichiers.
- **metadata.json (obligatoire)** - Contient une carte avec des informations sur la boîte. Plus important encore, le fournisseur cible.
- **info.json** - Il s'agit d'un document JSON qui peut fournir des informations supplémentaires sur la boîte qui s'affiche lorsqu'un utilisateur exécute `vagrant box list -i`. Plus d'informations sont fournies ici .

- **Vagrantfile** - Le Vagrantfile intégré dans la box Vagrant fournira quelques valeurs par défaut aux utilisateurs de la box. Pour plus d'informations sur la façon dont Vagrant fusionne les fichiers Vagrant, y compris ceux provenant du fichier box, consultez la documentation de Vagrantfile.

2. Installation d’un vagrant boxes

Si vous avez exécuté les commandes du dernier didacticiel, vous n'avez pas besoin d'ajouter de boîte ; Vagrant en a installé un lorsque vous avez initialisé votre projet. Parfois, vous souhaiterez peut-être installer une boîte sans créer un nouveau fichier Vagrant. Pour cela, vous utiliserez la `box add` sous-commande.

Vous pouvez ajouter une boîte à Vagrant avec `vagrant box add`. Cela stocke la boîte sous un nom spécifique afin que plusieurs environnements Vagrant puissent la réutiliser. Si vous n'avez pas encore ajouté de case, faites-le maintenant. Vagrant vous demandera de sélectionner un fournisseur. Tapez `2` et appuyez `Enter` pour sélectionner Virtualbox.

```
$ vagrant box add hashicorp/bionic64
```

Cela téléchargera la boîte nommée `hashicorp/bionic64` dans le catalogue de boîtes Vagrant Cloud de HashiCorp , où vous pourrez trouver et héberger des boîtes.

2. Versionnementd’un vagrant boxes

Depuis Vagrant 1.5, les box prennent en charge le versioning. Cela permet aux personnes qui créent des boîtes de transmettre des mises à jour à la boîte, et les personnes qui utilisent la boîte disposent d'un flux de travail simple pour vérifier les mises à jour, mettre à jour leurs boîtes et voir ce qui a changé.

IV. Vagrantfile

1. Version de configuration

Les versions de configuration sont le mécanisme par lequel Vagrant 1.1+ est capable de rester rétrocompatible avec les fichiers Vagrant 1.0.x de Vagrant, tout en introduisant des fonctionnalités et des options de configuration radicalement nouvelles.

Si vous exécutez `vagrant init` aujourd'hui, le Vagrantfile aura à peu près le format suivant :

```
Vagrant.configure("2") do |config|
  # ...
end
```

Le `"2"` dans la première ligne ci-dessus représente la version de l'objet de configuration `config` qui sera utilisé pour la configuration de ce bloc (la section entre le `do` et le `end`). Cet objet peut être très différent d'une version à l'autre.

Actuellement, il n'existe que deux versions prises en charge : "1" et "2". La version 1 représente la configuration de Vagrant 1.0.x. "2" représente la configuration pour 1.1+ menant à 2.0.x.

2. Variable d’environnement

Dans le fichier Vagrant, nous avons la possibilité de définir des variables, mais Vagrant propose également des variables d'environnement que nous pouvons utiliser pour récupérer des informations sur notre machine virtuelle.

3. Validate Vagrantfile

Vous pouvez vérifier la validité d'un fichier Vagrant en utilisant la commande `vagrant validate`. Cette commande examine le fichier Vagrantfile pour détecter d'éventuelles erreurs de syntaxe ou de configuration. Pour l'utiliser, ouvrez un terminal, accédez au répertoire où se trouve votre fichier Vagrantfile, puis exécutez la commande suivante :

```
vagrant validate
```

Si le fichier est valide, vous verrez un message indiquant "Vagrantfile validated successfully" . En cas d'erreurs, la commande affichera des messages d'erreur spécifiques à corriger.

V. Plugins

Les plugins Vagrant sont des extensions qui ajoutent des fonctionnalités supplémentaires à Vagrant, étendant ainsi ses capacités de base. Les plugins permettent d'intégrer des fonctionnalités spécifiques à un projet ou d'automatiser des tâches liées à la gestion des machines virtuelles. Voici quelques points clés à propos des plugins Vagrant :

1. **Installation des Plugins :** Vous pouvez installer des plugins Vagrant à l'aide de la commande `vagrant plugin install`. Par exemple, si vous voulez installer un plugin nommé "vagrant-example-plugin", vous pouvez exécuter la commande suivante :

   ```
   vagrant plugin install vagrant-example-plugin
   ```

2. **Vagrantfile Configuration :** Certains plugins nécessitent une configuration spécifique dans le fichier Vagrantfile pour être activés. La documentation du plugin spécifique fournira des détails sur la configuration requise.
3. **Types de Plugins :** Il existe plusieurs types de plugins Vagrant, y compris des plugins de provisionnement, des plugins de provisionnement cloud, des plugins de fournisseurs (pour prendre en charge de nouveaux fournisseurs de virtualisation), etc.
4. **Gestion des Plugins :** Vous pouvez lister les plugins installés avec la commande `vagrant plugin list` et les désactiver ou les désinstaller au besoin.
5. **Exemples de Plugins :**
   - **vagrant-vbguest :** Gère les additions invité VirtualBox automatiquement.
   - **vagrant-aws :** Intègre la prise en charge d'Amazon Web Services (AWS) comme fournisseur Vagrant.
   - **vagrant-disksize :** Permet de configurer la taille du disque dur virtuel pour la machine.

6. **Développement de Plugins :** Si vous avez des besoins spécifiques qui ne sont pas couverts par les plugins existants, vous pouvez également développer vos propres plugins Vagrant. La documentation officielle de Vagrant fournit des informations détaillées sur le développement de plugins.

Les plugins Vagrant offrent une flexibilité considérable pour personnaliser et étendre les fonctionnalités de Vagrant en fonction des besoins spécifiques de votre projet ou de votre environnement.

VI. synced folders (dossiers synchronisés)

Les synced folders (dossiers synchronisés) dans Vagrant sont une fonctionnalité qui permet de partager des fichiers entre votre système hôte (l'ordinateur sur lequel vous développez) et la machine virtuelle créée par Vagrant. Cela facilite le partage de données, de scripts, de configurations, et d'autres fichiers entre les deux environnements.

Voici quelques points clés à propos des synced folders dans Vagrant :

1. **Configuration dans le Vagrantfile :** La configuration des synced folders se fait dans le fichier Vagrantfile. Vous pouvez spécifier les dossiers à synchroniser et leur emplacement sur la machine virtuelle en utilisant la commande `config.vm.synced_folder` dans le Vagrantfile.

   Exemple :

   ```
   config.vm.synced_folder "chemin/local", "chemin/remote"
   ```

   Cela synchronisera le dossier local sur votre système hôte avec le dossier distant sur la machine virtuelle.

2. **Automatisation du Partage :** Les fichiers sont synchronisés automatiquement lorsque la machine virtuelle est créée ou démarrée avec la commande `vagrant up`. Les changements dans le dossier local sont également reflétés dans la machine virtuelle en temps réel.

3. **Types de Synced Folders :** Vagrant prend en charge différents types de synced folders en fonction du fournisseur de virtualisation utilisé (comme VirtualBox,

VMware, etc.). Ces types peuvent inclure des méthodes de partage plus efficaces adaptées à chaque environnement.

4. **Options de Configuration :** Vous pouvez spécifier diverses options lors de la configuration des synced folders, telles que la désactivation de la synchronisation automatique, la configuration des permissions, etc. Consultez la documentation de Vagrant pour obtenir une liste complète des options.
5. **Compatibilité Multi-Plateforme :** Les synced folders sont conçus pour être compatibles avec différents systèmes d'exploitation hôtes et invités. Cela signifie que vous pouvez développer sur une plateforme (comme Windows) tout en exécutant votre application dans une machine virtuelle avec un système d'exploitation différent (comme Linux).

VII. Provisioning

Le provisioning dans Vagrant fait référence au processus d'automatisation de la configuration d'une machine virtuelle. Il permet de définir et d'appliquer des configurations spécifiques à une machine virtuelle, telles que l'installation de logiciels, la configuration de services, la création d'utilisateurs, etc. Cela garantit que votre environnement de développement est configuré de manière cohérente et reproductible.

Voici quelques points clés à propos du provisioning dans Vagrant :

1. **Types de Provisioning :** Vagrant prend en charge différents types de provisionnement, notamment :
    - **Shell Provisioning :** L'exécution de scripts shell sur la machine virtuelle.
    - **Chef :** L'utilisation de Chef pour la gestion de configuration.
    - **Puppet :** L'utilisation de Puppet pour la gestion de configuration.
    - **Ansible :** L'utilisation d'Ansible pour le provisionnement.
    - **Salt :** L'utilisation de SaltStack pour le provisionnement.
2. **Configuration dans le Vagrantfile :** Vous spécifiez le provisionnement dans le fichier Vagrantfile en utilisant la commande `config.vm.provision`. Par exemple, pour configurer le provisionnement shell, vous pouvez avoir quelque chose comme :

```
config.vm.provision "shell", path: "script.sh"
```

Cela exécute le script shell `script.sh` sur la machine virtuelle.

3. **Scripts de Provisionnement :** Les scripts de provisionnement (qu'ils soient shell, Chef, Puppet, Ansible, etc.) sont des fichiers qui décrivent les étapes spécifiques à exécuter sur la machine virtuelle. Ces scripts peuvent installer des logiciels, configurer des services, créer des utilisateurs, etc.
4. **Provisionnement Multi-Étapes :** Vous pouvez définir plusieurs étapes de provisionnement dans le fichier Vagrantfile. Par exemple, vous pourriez d'abord exécuter un script shell, puis appliquer une configuration Chef.
5. **Reproducibilité :** Le provisionnement garantit que votre environnement de développement peut être reproduit facilement. Cela est particulièrement utile lorsque vous travaillez en équipe ou que vous déployez votre application sur plusieurs environnements.
6. **Personnalisation de l'Environnement :** Le provisionnement permet également de personnaliser l'environnement de développement en fonction de vos besoins spécifiques.
7. **Intégration avec des Outils de Gestion de Configuration :** Si vous utilisez déjà des outils de gestion de configuration tels que Chef, Puppet, ou Ansible dans votre infrastructure, vous pouvez réutiliser ces configurations pour provisionner vos machines virtuelles Vagrant.