Veille techniques

1. Qu'est-ce qu'une machine virtuelle ?

Une **machine virtuelle** (**VM**) est un environnement entièrement **virtualisé** fonctionnant sur une **machine physique**. Elle dispose de son propre système d'exploitation et utilise une portion des ressources de la machine hôte, telles que le **processeur** (CPU), la **mémoire vive** (RAM), le **stockage** (disque dur) et la **connectivité réseau**, selon la configuration définie.

Comment cela fonctionne ?

Le fonctionnement des **machines virtuelles** repose généralement sur un **hyperviseur**, qui peut être hébergé dans un cloud **privé, public, hybride** ou localement sur une **machine physique**. **L'hyperviseur partitionne** les ressources de l'hôte et en **alloue** une partie à chaque **VM**. Ce processus est géré par des logiciels tels que **VMware** ou **Microsoft Hyper-V**

Avantages :

- Idéal pour tester un nouveau **système d'exploitation** sans risque pour le disque **physique**
- Permet de **développer** des logiciels sur un système d'exploitation différent de celui de la machine hôte
- Offre la possibilité d'exécuter des **logiciels non compatible**s avec le système d'exploitation de la **machine physique**
- **Réduit les coûts** en hébergeant plusieurs **machines virtuelles** sur une seule **machine physique**

Inconvénients :

- **Sécurité :** Une **machine physique** hébergeant plusieurs **machines virtuelles** est plus **vulnérable** aux attaques ; une faille de sécurité dans l'hôte peut affecter toutes les **VM**.
- **Dépendance matérielle :** Les **machines virtuelles** dépendent de la disponibilité et de la puissance de la machine physique. Si celle-ci tombe en panne, toutes les **VM** deviennent inaccessibles, et leurs performances sont limitées par la capacité de l'hôte.

2. Quelle est la différence entre une machine virtuelle et un conteneur ?

Un **conteneur** est une unité d'exécution isolée qui regroupe une **application** et toutes ses **dépendances nécessaires** pour fonctionner de manière cohérente sur **différents environnements**.

Les **conteneurs** encapsulent le **code de l'application**, ses **bibliothèques**, **configurations**, et tout ce qui est nécessaire pour exécuter l'application. Ils permettent ainsi de s'assurer qu'une **application** fonctionnera de manière **identique**, que ce soit en **développement**, en **test** ou en **production**.

La **différence** est que, contrairement à une **machine virtuelle**, un **conteneur** ne nécessite pas de **système d'exploitation** complet, mais partage le **noyau** de l'**OS** de l'**hôte**. Cela rend les **conteneurs** beaucoup plus **légers** et **rapides** à démarrer.

3. Qu'est-ce que le Docker ? Qu'est-ce que la conteneurisation ?

Docker est une plateforme de **conteneurisation** open source qui permet de créer, déployer et exécuter des **applications** dans des **environnements isolés** appelés **conteneurs**.

La **conteneurisation** est un **processus** de **déploiement** logiciel qui **encapsule** le code d'une **application** avec toutes les **bibliothèques** et **fichiers** nécessaires à son exécution, assurant ainsi sa **portabilité** et sa **cohérence** sur n'importe quelle **infrastructure**.

4. Quels sont les avantages de la conteneurisation ?

Les **développeurs** choisissent la **conteneurisation** pour ses nombreux **avantages** dans le déploiement **d'applications modernes**.

- **Portabilité** : Les **conteneurs** permettent de déployer des **applications** sur différents **environnements** sans modification de **code**, assurant une compatibilité sur plusieurs **systèmes d'exploitation**, comme **Linux** et **Windows**.
- **Scalabilité** : **Légers** et **rapides** à exécuter, les conteneurs facilitent la mise à l'échelle en ajoutant plusieurs **instances** sur une **même machine**, sans interférence entre eux.
- **Tolérance aux pannes** : Grâce à leur **isolation**, un **conteneur défectueux** n'affecte pas les autres, ce qui renforce la résilience des applications **conteneurisées**.
- **Agilité** : Les **conteneurs** permettent des **cycles** de mise à jour rapides et des modifications isolées, favorisant des déploiements et des résolutions de problèmes plus **rapides**.
- **Facilité d'intégration continue et déploiement continu (CI/CD)** : La **conteneurisation** simplifie la mise en place de **pipelines CI/CD**, permettant un déploiement rapide et fiable des nouvelles versions de l'**application**.

5. Qu'est-ce qu'une image Docker, quelles différences avec un conteneur ?

Comme le **conteneur**, c'est une **technologie** de **déploiement** d'une **application** mais ,contrairement à un **conteneur**, les **images Docker** sont des **modèles** en **lecture seule** qui contiennent des **instructions** pour créer un **conteneur**. Une **image Docker** est un **instantané** ou un plan des **bibliothèques** et des **dépendances** requises dans un **conteneur** pour qu'une **application** puisse s'exécuter.

6. Qu'est-ce qu'un Dockerfile ?

Dockerfile est un document texte contenant une série de **commandes** nécessaires à la création d'une **image Docker.**

```
# Utiliser une image de base légère avec Python
FROM python:3.9-slim

# Définir le répertoire de travail dans le conteneur
WORKDIR /app

# Copier les fichiers de l'application dans le conteneur
COPY . /app

# Installer les dépendances listées dans requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Exposer le port sur lequel l'application écoute
EXPOSE 5000

# Définir la commande pour lancer l'application
CMD ["python", "app.py"]
```