Questions d'entretien Docker - Guide Professionnel avec Explications

1. Qu'est-ce que Docker ?

Docker est une plateforme open source permettant de développer, expédier et exécuter des applications dans des conteneurs. Un conteneur est une unité standardisée de logiciel contenant tout le nécessaire pour exécuter une application : code, bibliothèques, dépendances et configurations.

2. Quelle est la différence entre les conteneurs et les machines virtuelles ?

Caractéristique	Conteneur Docker	Machine Virtuelle
Système d'exploitation	Partage le noyau de l'hôte	OS complet pour chaque VM
Légèreté	Très léger	Plus lourd
Démarrage	Très rapide (secondes)	Lent (minutes)
Isolation	Moins forte	Forte

3. Quel est le cycle de vie d'un conteneur Docker ?

1. **Create** - Création d'un conteneur à partir d'une image
2. **Start** - Lancement du conteneur
3. **Run** - Exécution du processus principal
4. **Pause/Unpause** - Mise en pause / Reprise
5. **Stop** - Arrêt grâce à un signal
6. **Kill** - Arrêt forcé
7. **Remove** - Suppression du conteneur

4. Quels sont les différents composants de Docker ?

- **Docker Client** : Interface CLI pour interagir avec Docker
- **Docker Daemon** : Service exécutant en arrière-plan
- **Docker Images** : Modèle à partir duquel un conteneur est créé
- **Docker Containers** : Instances des images
- **Docker Registry** : Stocke les images (ex: Docker Hub)
- **Dockerfile** : Script de construction d'une image

5. Quelle est la différence entre `COPY` et `ADD` ?

- `COPY` : Copie uniquement des fichiers/dossiers locaux dans l'image
- `ADD` : Fait tout ce que `COPY` fait + capable de décompresser automatiquement les archives `.tar`, et prendre des fichiers distants via URL

Recommandation : Utilisez `COPY` sauf si vous avez besoin de fonctionnalités spécifiques de `ADD`.

6. Différence entre `CMD` et `ENTRYPOINT`

- `CMD` : Spécifie la commande par défaut, mais peut être écrasée
- `ENTRYPOINT` : Spécifie la commande principale, difficilement écrasable

Exemple :

```
ENTRYPOINT ["python"]
CMD ["app.py"]
```

=> Cela exécutera : `python app.py`

7. Quels sont les types de réseaux Docker et quel est le réseau par défaut ?

- **bridge** (par défaut)
- **host**
- **none**
- **overlay** (Swarm)
- **macvlan**

8. Comment isoler les réseaux entre conteneurs ?

- Créer un réseau personnalisé avec `docker network create`
- Assigner chaque conteneur à un réseau différent

```
docker network create monreseau
```

```
docker run -d --name conteneur1 --network=monreseau nginx
```

9. Qu'est-ce que le build multi-étapes (multi-stage build) ?

C'est une technique permettant de séparer les phases de build et d'exécution pour produire une image finale légère.

Exemple :

```
FROM node:18 AS build
WORKDIR /app
```

```
COPY . .
RUN npm install && npm run build

FROM nginx:alpine
COPY --from=build /app/dist /usr/share/nginx/html
```

10. Qu'est-ce qu'une image "distroless" ?

Ce sont des images dépourvues de système d'exploitation complet (pas de shell, pas de gestionnaire de paquets). Elles n'incluent que les dépendances essentielles pour exécuter l'application, rendant l'image plus sûre et plus légère.

11. Défis rencontrés en production avec Docker

- **Point de défaillance unique** : Docker Daemon est un processus unique, sa panne arrête toutes les applications.
- **Exécution en tant que root** : Problème de sécurité si compromis.
- **Problèmes de ressources** : Si trop de conteneurs tournent sur une petite machine.

12. Quelles mesures prendre pour sécuriser les conteneurs ?

- Utiliser des **images distroless** ou optimisées
- Configurer réseaux personnalisés pour **isoler** les conteneurs
- Scanner les images avec des outils comme `snyk`, `trivy`
- Appliquer le principe du **moindre privilège**
- Signer et vérifier les images (Content Trust)

Conclusion

Docker est un outil puissant pour la conteneurisation, mais sa maîtrise nécessite une compréhension approfondie de ses composants, de sa sécurité et des meilleures pratiques. Bien que flexible et efficace, il présente aussi des défis en production qui doivent être anticipés et gérés correctement.