Questions d'entretien en Shell Scripting pour les DevOps Engineers

1. Citez quelques commandes couramment utilisées en shell

Les commandes les plus utilisées en shell sont :

- `ls` : Liste les fichiers et dossiers du répertoire courant.
- `cp` : Copie des fichiers ou des répertoires.
- `mv` : Déplace ou renomme des fichiers.
- `mkdir` : Crée un nouveau répertoire.
- `touch` : Crée un fichier vide ou met à jour le timestamp d'un fichier.
- `vim` : Éditeur de texte en mode terminal.
- `grep` : Recherche un texte dans un fichier ou une sortie de commande.
- `find` : Recherche des fichiers et des répertoires selon des critères.
- `top` : Affiche en temps réel l'utilisation des ressources système.
- `sar` : Collecte, affiche et enregistre les données de performance du système.
- `df` : Affiche l'espace disque utilisé et disponible sur le système.
- `ps` : Affiche les processus en cours.
- `nano` : Un autre éditeur de texte en mode terminal.

---

2. Écrire un script shell pour afficher la liste des processus en cours

Voici un script pour afficher la liste des processus :

bash

```
#!/bin/bash
echo "Liste des processus en cours d'exécution :"
ps -ef
```

Si on souhaite afficher uniquement les identifiants des processus (`PID`), on peut utiliser `awk` :

bash

```
#!/bin/bash
ps -ef | awk '{print $2}'
```

---

3. Écrire un script pour afficher uniquement les erreurs d'un journal distant

On peut utiliser `curl` pour récupérer le journal et `grep` pour filtrer les erreurs :

bash

```
#!/bin/bash
# Remplacez "URL_DU_LOG" par l'URL réelle du fichier journal
curl -s URL_DU_LOG | grep "ERROR"
```

---

4. Écrire un script shell pour afficher les nombres divisibles par 3 ou 5 mais pas par 15

Voici un script Shell qui réalise cette tâche :

```
bash

#!/bin/bash
echo "Nombres divisibles par 3 ou 5 mais pas par 15 :"

for i in {1..100}
do
    if ([ $(expr $i % 3) -eq 0 ] || [ $(expr $i % 5) -eq 0 ]) && [ $(expr
$i % 15) -ne 0 ]; then
        echo $i
    fi
done
```

---

5. Écrire un script pour compter le nombre de lettres "s" dans "Mississippi"

Le script suivant permet de compter les occurrences de la lettre `s` dans `Mississippi` :

```
bash

#!/bin/bash
x="mississippi"
echo -n "$x" | grep -o "s" | wc -l
```

---

6. Comment déboguer un script shell ?

Pour activer le mode de débogage et voir l'exécution ligne par ligne du script, utilisez :

```
bash

set -x
```

Pour désactiver le mode de débogage :

```
bash

set +x
```

---

7. Qu'est-ce que le crontab sous Linux ? Donnez un exemple d'utilisation

`crontab` est utilisé pour programmer l'exécution automatique de tâches à intervalles réguliers.

Par exemple, pour surveiller l'état du système et envoyer un rapport par e-mail toutes les 6 minutes :

```
bash

*/6 * * * * /chemin/vers/script.sh | mail -s "Rapport du système"
admin@example.com
```

---

8. Comment ouvrir un fichier en lecture seule ?

Utilisez la commande suivante pour ouvrir un fichier en lecture seule avec `vim` :

```
bash

vim -R test.txt
```

---

9. Différence entre un lien symbolique (soft link) et un lien physique (hard link)

Type de lien	Définition
Lien symbolique (Soft link)	Fichier qui pointe vers un autre fichier. Si le fichier original est supprimé, le lien devient invalide.
Lien physique (Hard link)	Copie directe du fichier original. Le fichier reste accessible même si l'original est supprimé.

Exemple :

```
bash

# Créer un lien symbolique
ln -s fichier_original lien_symbolique

# Créer un lien physique
ln fichier_original lien_physique
```

---

10. Différence entre `break` et `continue` dans une boucle

- `break` : Arrête complètement la boucle.
- `continue` : Saute l'itération en cours et passe à la suivante.

Exemple :

```
bash

#!/bin/bash
for i in {1..10}; do
    if [ $i -eq 5 ]; then
        break  # La boucle s'arrête quand i = 5
    fi
    echo $i
done
```

Exemple avec `continue` :

```
bash

#!/bin/bash
for i in {1..10}; do
    if [ $i -eq 5 ]; then
        continue  # Ignore 5 et passe à l'itération suivante
    fi
```

```
    echo $i
done
```

---

11. Inconvénients du Shell Scripting

1. **Erreurs fréquentes et coûteuses** : Une petite erreur peut causer des problèmes importants.
2. **Exécution lente** : Comparé aux langages compilés comme C ou Go.
3. **Utilisation excessive des processus** : Chaque commande lance un processus distinct.
4. **Complexité croissante** : Pas adapté aux tâches trop complexes.

---

12. Types de boucles et quand les utiliser

Type de boucle	Utilisation
`for` loop	Pour parcourir une liste ou une plage de valeurs.
`while` loop	Exécute tant qu'une condition est vraie.
`until` loop	Exécute tant qu'une condition est fausse.

Exemple de boucle `while` :

bash

```
#!/bin/bash
x=1
while [ $x -le 5 ]; do
    echo "Valeur actuelle de x : $x"
    ((x++))
done
```

---

13. Bash est-il dynamique ou statiquement typé ? Pourquoi ?

Bash est **dynamiquement typé** car les variables n'ont pas besoin d'être déclarées avec un type spécifique. Vous pouvez stocker une chaîne de caractères dans une variable et ensuite y affecter un nombre sans erreur.

---

14. Outils de diagnostic réseau

- **`traceroute`** : Affiche les routes empruntées par un paquet.

  bash

  ```
  traceroute google.com
  ```

- **`tracepath`** : Similaire à `traceroute` mais sans besoin de privilèges root.

```
bash

tracepath google.com
```

---

15. Comment trier une liste de noms dans un fichier ?

Utilisez la commande `sort` :

```
bash

sort noms.txt
```

Pour trier en ordre inverse :

```
bash

sort -r noms.txt
```

---

16. Comment gérer un fichier journal volumineux généré quotidiennement ?

On utilise `logrotate` pour la gestion des fichiers logs :

- Compression avec `gzip` ou `zip`.
- Suppression automatique des anciens logs.

Exemple de fichier `/etc/logrotate.d/mon_application` :

```
bash

/var/log/mon_application.log {
    daily
    rotate 7
    compress
    missingok
    notifempty
}
```