Cours sur Linux et le Scripting Shell pour DevOps

Introduction

Qu'est-ce qu'un Système d'Exploitation (OS) ?

Un **système d'exploitation (OS)** est un logiciel qui agit comme un intermédiaire entre le matériel (hardware) et les applications (software). Il gère les ressources de l'ordinateur et permet aux programmes de s'exécuter efficacement.

Pourquoi choisir Linux ?

Linux est l'un des systèmes d'exploitation les plus populaires et utilisés dans le monde du DevOps en raison des avantages suivants :

- **Gratuit et Open Source** : Pas de frais de licence, code source accessible.
- **Sécurisé** : Moins vulnérable aux virus et attaques.
- **Nombreuses distributions** : Ubuntu, Debian, CentOS, Fedora, Arch Linux, etc.
- **Stable et performant** : Excellente gestion des ressources.
- **Personnalisable** : Peut être modifié et adapté aux besoins spécifiques.

Fondamentaux du Scripting Shell

Qu'est-ce qu'un Shell ?

Un **Shell** est une interface qui permet d'interagir avec le système d'exploitation via des commandes. Le scripting Shell permet d'automatiser ces commandes dans des scripts.

Pourquoi utiliser un script Shell ?

- Automatiser des tâches répétitives
- Gagner du temps et éviter les erreurs humaines
- Exécuter plusieurs commandes en une seule fois

Écrire un Script Shell

Un script Shell est un fichier texte contenant une série de commandes. Il doit commencer par un **shebang**, qui définit quel interpréteur utiliser :

```
#!/bin/bash
```

Exemple de script `first-shell-script.sh` :

```
#!/bin/bash

# Affiche la liste des fichiers
```

```
ls -ltr

echo "Script terminé"
```

Pour exécuter le script :

```
chmod +x first-shell-script.sh
./first-shell-script.sh
```

Commandes de base

Commande	Description
`ls`	Liste les fichiers du répertoire
`ls -ltr`	Liste les fichiers avec détails et tri par date
`man`	Affiche le manuel d'une commande
`touch fichier.txt`	Crée un fichier vide
`mkdir mon_dossier`	Crée un dossier
`top`	Affiche les processus en cours
`df -h`	Affiche l'espace disque disponible
`nproc`	Affiche le nombre de cœurs CPU disponibles
`free -g`	Affiche la mémoire disponible en Go

Gestion des Processus

Affichage des processus actifs

```
ps -ef | grep "nom_du_processus"
```

Terminer un processus

```
kill -9 <ID_du_processus>
```

Gestion des signaux avec `trap`

```
trap "echo Ne pas utiliser Ctrl+C" SIGINT
```

Gestion des fichiers et téléchargements

Rechercher un fichier

```
find / -name "nom_du_fichier"
```

Téléchargement de fichiers

- **`curl`** : Récupère des informations depuis internet

  ```
  curl -X GET api.foo.com
  ```

- **`wget`** : Télécharge un fichier depuis internet

  ```
  wget http://example.com/fichier.zip
  ```

Structures de contrôle en Shell

`if-else`

```
a=4
b=10

if [ "$a" -gt "$b" ]
then
    echo "A est plus grand que B"
else
    echo "B est plus grand que A"
fi
```

Boucle `for`

```
for i in {1..100}; do echo $i; done
```

Redirections et pipes

Opérateur	Description
`>`	Redirige la sortie vers un fichier (écrase le contenu existant)
`>>`	Redirige la sortie vers un fichier (ajoute à la fin)
`<`	Redirige un fichier comme entrée
`` ` ``	`` ` `` Envoie la sortie d'une commande à une autre

Exemple :

```
echo "data" | grep "d"
```

Debugging des scripts

Commande	Description
`set -x`	Active le mode debug (affiche chaque commande exécutée)
`set -e`	Arrête le script en cas d'erreur
`set -o pipefail`	Arrête le script si une commande dans un pipe échoue
`set -exo pipefail`	Active toutes ces options

Conclusion

Linux et le scripting Shell sont essentiels pour les DevOps. Maîtriser ces concepts permet d'automatiser les tâches et d'améliorer l'efficacité des opérations. Pratiquez ces commandes et expérimentez avec vos propres scripts pour renforcer vos compétences !