Examen – Automatisation & Scripting

Durée : 1h30

> **Consignes générales :**
>
> - Rendre les scripts avec des noms de fichiers exacts tels qu'indiqués.
> - Vous pouvez utiliser l'aide des commandes (`Get-Help` , `man` , etc.).
> - envoyer un zip nommé VOTRENOM.zip contenant les 3 scripts à ybahi4@myges.fr .

Exercice 1 — PowerShell : Vérification de l'espace disque

Objectif : Créer un script PowerShell simple qui vérifie l'espace disque de `C:` et écrit un log.

Consignes

1. Le script s'appelle **`Check-Disk.ps1`** .
2. Il doit :
   - Récupérer l'espace disque **total** et l'espace **libre** sur le disque `C:` .
   - Calculer le **pourcentage d'espace utilisé**.
3. Il doit écrire une ligne dans un fichier de log :
   `C:\Logs\diskcheck.log`
   sous la forme :

   ```
   2025-11-05 14:10 - C: 40% utilisé (100 Go libres sur 250 Go)
   ```

4. Si le pourcentage d'utilisation est **supérieur à 80%**, le message doit contenir le mot **`ALERTE`** , par exemple :

   ```
   2025-11-05 14:10 - C: 92% utilisé (20 Go libres sur 250 Go) - ALERTE
   ```

5. Le script doit créer le dossier `C:\Logs\` s'il n'existe pas.

Livrables

- Fichier : **`Check-Disk.ps1`**

Exercice 2 — Linux : Audit simplifié du système

Objectif : Écrire un script Bash qui vérifie quelques informations système et génère un rapport.

Consignes

1. Le script s'appelle **`sys_audit.sh`** .
2. Il doit :
   - Afficher l'utilisation du disque racine `/` (commande du type `df -h /`).
   - Afficher la mémoire libre en Mo (commande du type `free -m`).
   - Vérifier si le service **`ssh`** est **actif**.
3. Il doit générer un fichier de rapport :
   `/var/reports/sys_audit-YYYYMMDD.txt`
   (par exemple : `/var/reports/sys_audit-20251105.txt`)
4. Le rapport doit contenir au minimum :

```
=== System Audit Report - 2025-11-05 ===
Disk usage : 40% used (12G/30G)
Free memory: 980 MB
SSH service : active
```

5. À la fin du fichier, ajouter une ligne :

```
Audit completed at HH:MM
```

6. Le script doit créer le dossier `/var/reports/` s'il n'existe pas (avec les droits nécessaires).

Livrables

- Fichier : **`sys_audit.sh`**

Exercice 3 — API Open-Meteo : Collecte météo + JSON & CSV

Objectif : Interroger l'API **Open-Meteo** pour deux villes, puis stocker les résultats en JSON et en CSV.

Villes à utiliser

- **Lille** : latitude `50.6292` , longitude `3.0573`
- **Lyon** : latitude `45.7640` , longitude `4.8357`

API

Pour chaque ville, appeler :

```
https://api.open-meteo.com/v1/forecast?latitude=<LAT>&longitude=<LON>&current_weather=t
```

(en remplaçant `<LAT>` et `<LON>` par les valeurs de la ville)

Consignes

1. Le script s'appelle **`collect_weather`** (langage au choix : Bash + `curl` + `jq` , PowerShell, Python, etc.).
2. Pour chaque ville (Lille et Lyon), extraire au minimum :
    - `current_weather.time`
    - `current_weather.temperature` (°C)
    - `current_weather.windspeed` (km/h)
    - `current_weather.winddirection` (°)
3. Créer/mettre à jour deux fichiers dans le répertoire courant :
    a. **`weather.json`**
    Format **JSON Lines** (une ligne JSON par ville et par exécution).
    Chaque ligne doit contenir au minimum les champs :

    ```
    {
      "city": "Lille",
      "ts": "2025-11-05T13:00",
      "temperature_c": 12.3,
      "windspeed_kmh": 15.0,
      "wind_dir_deg": 220
    }
    ```

    b. **`weather.csv`**
    Si le fichier n'existe pas, écrire l'en-tête :

    ```
    city,ts,temperature_c,windspeed_kmh,wind_dir_deg
    ```

    Puis ajouter une ligne par ville, par exécution, par exemple :

```
Lille,2025-11-05T13:00,12.3,15.0,220
Lyon,2025-11-05T13:00,10.8,10.5,180
```

4. À chaque exécution du script, ajouter une ligne dans **`weather.log`** avec la date/heure et le nombre de villes traitées, par exemple :

```
2025-11-05 13:05 - 2 cities processed (Lille, Lyon)
```

Livrables

- Script `collect_weather` (avec extension selon le langage : `.sh`, `.ps1`, `.py`, …)