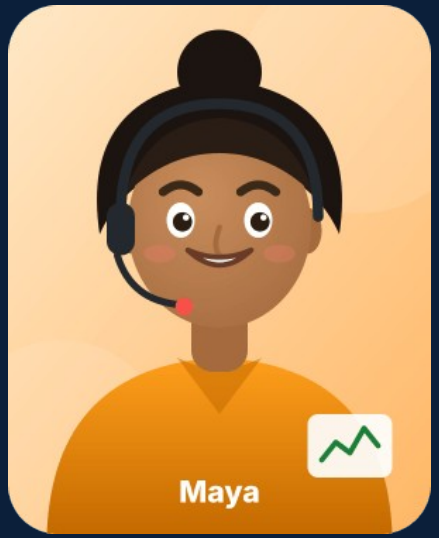
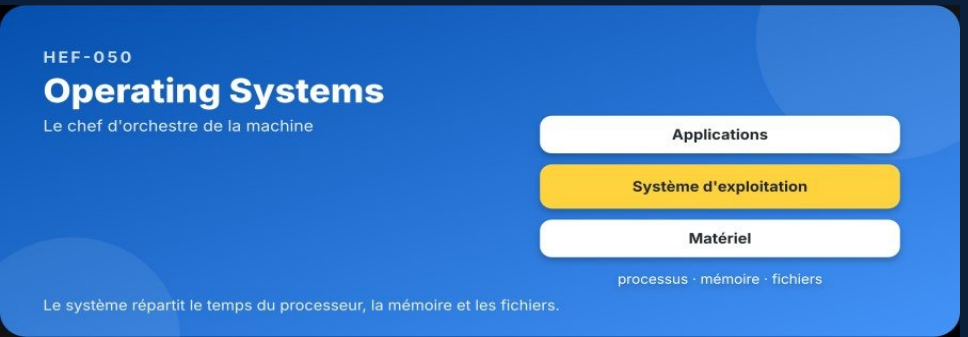
HEF-050 · MISSION 10 / 10

Investigation Linux

Un serveur est lent, un disque est plein, un service plante : par où commencer ?



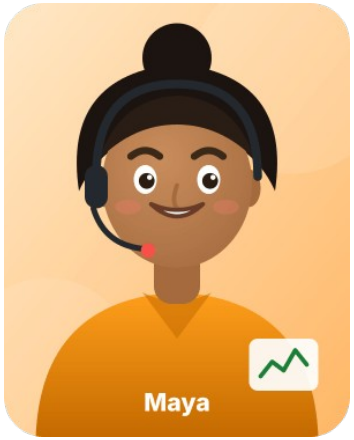


Badge : Enquêteur·rice système

85 min · intermédiaire

L'histoire

Samedi, 22 h. Maya appelle : « Le site est très lent depuis ce soir ! » Sam est en vacances. Kodi ouvre un terminal sur le serveur et se rend compte qu'il ne sait pas par où commencer.



> *« Respire, dit la voix de Sam au téléphone. Tu as une méthode : de la vue d'ensemble vers le détail, une ressource après l'autre. »*

Dans cette dernière mission, tu assembles tout le module en une démarche d'investigation.

Ta mission

À la fin, je serai capable de…

1. appliquer une méthode de diagnostic ressource par ressource (CPU, mémoire, disque, réseau)
2. lire la sortie des commandes de base (`top`, `free`, `df`, `ps`, `ss`)
3. relier un symptôme à une cause grâce aux notions du module
4. écrire un script de diagnostic en Python
5. rédiger un rapport d'incident clair

Défi d'échauffement

Une voiture fait un bruit anormal. Tu n'es pas mécanicien. En quelle **ordre** vérifies-tu : le niveau d'huile, les pneus, les freins, le moteur ? Pourquoi commence-t-on souvent par les vérifications faciles et générales avant de démonter quoi que ce soit ?

Le concept

La méthode USE

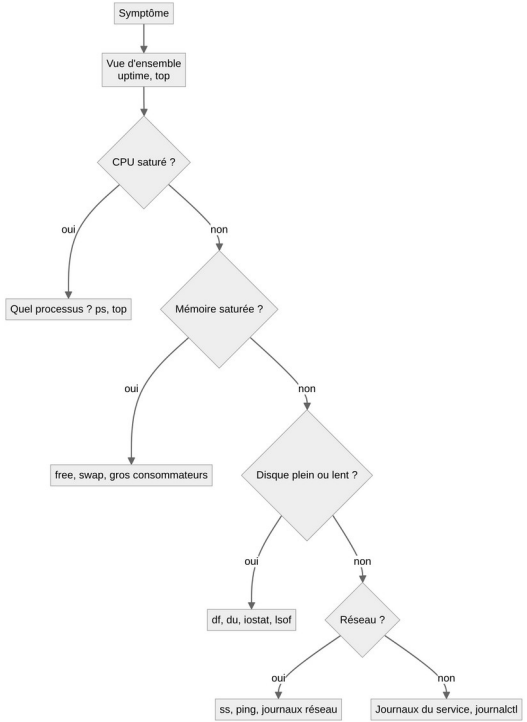
Pour chaque **ressource** (CPU, mémoire, disque, réseau), on regarde trois choses (méthode **USE** de Brendan Gregg) :

Lettre	Question	Exemple
U — Utilisation	à quel point est-elle occupée ?	CPU à 98 %
S — Saturation	y a-t-il une file d'attente ?	15 processus en attente de CPU
E — Erreurs	y a-t-il des erreurs ?	erreurs de disque, paquets perdus

Le parcours d'enquête

Le concept

Le parcours d'enquête (suite)



Le concept

Le parcours d'enquête (suite)

Description du schéma : à partir du symptôme, on regarde d'abord une vue d'ensemble, puis on teste successivement CPU, mémoire, disque, réseau, et enfin les journaux du service. À chaque étape, si la ressource est saturée, on détaille les causes ; sinon on passe à la suivante.

Les commandes de base

Besoin	Commande	À regarder
Charge globale	`uptime`	trois moyennes de charge (1, 5, 15 min)
Processus actifs	`top`, `ps aux --sort=-%cpu \`	`head`
Mémoire	`free -h`	`available`, `swap used`
Disque (espace)	`df -h`	`Use%` proche de 100 %
Disque (qui occupe)	`du -sh /var/* \`	`sort -h`

Le concept

Les commandes de base (suite)

Besoin	Commande	À regarder
Fichiers ouverts	`lsof | grep deleted`	
Réseau	`ss -tulpn`, `ping`	ports en écoute, latence
Journaux	`journalctl -u service -n 100`	erreurs récentes
Appels système	`strace -p PID`	ce que fait un processus

Lire la charge moyenne

La **charge** (*load average*) indique le nombre moyen de processus prêts ou bloqués sur les E/S. Sur une machine à **4 cœurs**, une charge de 4 signifie « pleinement occupée » ; 12 indique une saturation (file d'attente).

Le concept

Exemples de causes (liées au module)

Symptôme	Causes possibles	Mission du module
Machine très lente + disque actif	swap / thrashing	5
« No space left on device »	disque plein, ou log supprimé mais ouvert	6
« Permission denied »	droits ou utilisateur du service	7
Service qui ne s'arrête pas	SIGTERM ignoré, tâche bloquée	9
Processus bloqué en `D`	attente d'E/S	8
Un cœur à 100 %, les autres libres	programme mono-thread	3

Le concept

Sortie fictive d'un diagnostic

```
$ uptime
 22:14:03 up 41 days,  load average: 11.80, 9.40, 5.10     (4 cœurs)
$ free -h
              total   used   free  available   swap used
Mem:           7.8G   7.6G   120M      180M        3.9G
$ df -h /
Filesystem  Size  Used Avail Use%
/dev/sda1    50G   32G   16G  67%
```

Lecture : charge 11,8 sur 4 cœurs, mémoire presque épuisée, **swap utilisé à 3,9 Go** → thrashing probable (mission 5). Le disque n'est pas la cause.

Exemple pas à pas

Problème : écrire un script qui lit des « sorties de commandes » (fournies en texte) et pose un diagnostic automatique.

```
import re

SORTIES = {
    "uptime": "22:14:03 up 41 days, load average: 11.80, 9.40, 5.10",
    "cpus": 4,
    "free": "Mem: 7800 7600 120 180 swap_used 3900",
    "df": "/dev/sda1 50G 32G 16G 67%",
}

def diagnostiquer(s):
    constats = []
    charge1 = float(re.search(r"load average: ([\d.]+)", s["uptime"]).group(1))
    if charge1 > 2 * s["cpus"]:
        constats.append(f"CPU saturé : charge {charge1} pour {s['cpus']} cœurs")
    mots = s["free"].split()
```

Exemple pas à pas (suite)

```
    total, utilise, libre, disponible = (int(mots[i]) for i in range(1, 5))
    swap = int(mots[-1])
    if disponible < 0.1 * total:
        constats.append(f"mémoire presque épuisée : {disponible} Mo disponibles su
   r {total}")
    if swap > 500:
        constats.append(f"swap utilisé : {swap} Mo → thrashing probable")
    pct = int(s["df"].split()[-1].rstrip("%"))
    if pct > 90:
        constats.append(f"disque presque plein ({pct} %)")
    return constats or ["rien d'anormal détecté"]

for c in diagnostiquer(SORTIES):
    print("-", c)
```

Résultat : trois constats (CPU saturé, mémoire presque épuisée, swap utilisé), pas de problème de disque. L'enquête continue sur **quel processus** consomme la mémoire.

Exemple pas à pas (suite)

Rapport d'incident type :

Section	Contenu
Symptôme	site très lent depuis 21 h 30
Constats	charge 11,8/4 cœurs, mémoire disponible 180 Mo, swap 3,9 Go
Cause probable	un processus consomme trop de mémoire → thrashing
Action	identifier le processus, le redémarrer proprement (SIGTERM), vérifier la fuite
Prévention	alerte sur la mémoire disponible, limite mémoire du service

À toi, avec aide

Voici trois diagnostics. Pour chacun, écris la cause probable et la prochaine commande à lancer.

1. `df -h` : `/` à 100 %, `du` montre `/var/log` à 38 Go.
2. `top` : un seul processus à 100 % CPU, charge 1,0 sur 8 cœurs, le site est lent.
3. `df -h` : `/` à 62 %, mais l'application dit « No space left on device ».

À toi, avec aide · réponse

Indices progressifs

Indice 1

Pour 1, pense aux journaux sans rotation.

Indice 2

Pour 2, la charge est faible : le problème n'est pas la saturation globale ; pense au GIL / mono-thread.

Indice 3

Pour 3, deux explications possibles : les inodes sont épuisés, ou un fichier supprimé est encore ouvert.

À toi, avec aide · réponse

Réponse

1. **Journaux sans rotation** qui remplissent le disque. Prochaine commande : `ls -lhS /var/log | head` puis configurer la rotation (`logrotate`) ; attention à ne pas supprimer un fichier encore ouvert (le tronquer avec `> fichier`).
2. **Programme mono-thread** : un cœur sature pendant que 7 sont inactifs. Prochaine commande : `top -H -p PID` (threads), `py-spy` ou profil ; paralléliser (processus) ou optimiser le code.
3. **Inodes épuisés** (`df -i`) ou **fichier supprimé encore ouvert** (`lsof | grep deleted`). Prochaine commande : `df -i` puis `lsof +L1`.

À toi, en solo

Ton propre diagnostic. Écris `diagnostic_local()` qui, sur la machine où il tourne (Linux, avec repli propre sinon), collecte : la charge moyenne (`os.getloadavg`), le nombre de cœurs, la mémoire totale et disponible (`/proc/meminfo`), l'espace disque libre, les 3 plus gros processus en mémoire (parcours de `/proc/*/status`), et affiche un court rapport avec des alertes selon des seuils que tu choisis.

À toi, en solo · réponse

Pistes de correction

```
import os, shutil

def meminfo():
    infos = {}
    with open("/proc/meminfo") as f:
        for ligne in f:
            cle, _, v = ligne.partition(":")
            infos[cle] = int(v.split()[0])        # en Ko
    return infos

def gros_processus(n=3):
    res = []
    for entree in os.listdir("/proc"):
        if entree.isdigit():
            try:
                nom, rss = None, 0
                with open(f"/proc/{entree}/status") as f:
                    for ligne in f:
                        if ligne.startswith("Name:"):
```

À toi, en solo · réponse

Pistes de correction (suite)

```
                            nom = ligne.split()[1]
                        elif ligne.startswith("VmRSS:"):
                            rss = int(ligne.split()[1])
                if nom:
                    res.append((rss, nom, int(entree)))
            except OSError:
                pass
    return sorted(res, reverse=True)[:n]

def diagnostic_local():
    rapport, alertes = [], []
    cpus = os.cpu_count() or 1
    try:
        c1, c5, c15 = os.getloadavg()
        rapport.append(f"charge {c1:.2f} {c5:.2f} {c15:.2f} pour {cpus} cœurs")
        if c1 > 2 * cpus:
            alertes.append("CPU saturé")
    except OSError:
        rapport.append("charge non disponible")
```

À toi, en solo · réponse

Pistes de correction (suite)

```
    try:
        m = meminfo()
        dispo_pct = 100 * m["MemAvailable"] / m["MemTotal"]
        rapport.append(f"mémoire disponible {m['MemAvailable'] // 1024} Mo ({dispo
   _pct:.0f} %)")
        if dispo_pct < 10:
            alertes.append("mémoire presque épuisée")
        for rss, nom, pid in gros_processus():
            rapport.append(f"  {nom} (pid {pid}) : {rss // 1024} Mo")
    except OSError:
        rapport.append("/proc non disponible")
    total, _, libre = shutil.disk_usage("/")
    pct = 100 * (1 - libre / total)
    rapport.append(f"disque / : {pct:.0f} % utilisé")
    if pct > 90:
        alertes.append("disque presque plein")
    return rapport, alertes

rapport, alertes = diagnostic_local()
```

À toi, en solo · réponse

Pistes de correction (suite)

```
print("\n".join(rapport))
print("ALERTES :", alertes or "aucune")
```

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Redémarrer avant de comprendre	On perd les preuves, le problème revient	Collecter les informations d'abord
Regarder une seule ressource	On passe à côté de la vraie cause	Parcourir CPU, mémoire, disque, réseau
Ignorer les journaux	La réponse y était	Lire les derniers messages d'erreur
Corriger sans noter	Personne ne sait ce qui a été fait	Journal d'enquête et rapport

Mini-boss

Mini-boss

Dossier d'incident final du module HEF-050. On te donne cette situation fictive (à compléter par toi avec des sorties de commandes crédibles) :

> *Lundi, 9 h 15. Le service de paiement de HashBox répond avec 12 secondes de retard. L'équipe a vu un pic de trafic à 9 h 00 et une mise à jour du service de reporting à 8 h 45.*

1. Écris des « sorties de commandes » réalistes (`uptime`, `free`, `df -h`, `ps`, `ss`, extrait de journal) pour **deux** causes différentes plausibles (par exemple : fuite mémoire + swap ; ou un job de reporting qui sature le disque).
2. Écris un programme qui prend ces sorties et pose un diagnostic.
3. Rédige un rapport d'incident complet : chronologie, symptôme, constats, cause racine, correction, vérification, prévention (alertes, limites de ressources, ordonnancement du job).
4. Relie chaque constat à une mission du module (processus, mémoire virtuelle, fichiers, permissions, signaux…).

Mini-boss · réponse

Pistes de correction

Un bon dossier contient : une chronologie à la minute ; des constats chiffrés (charge/cœurs, mémoire disponible, utilisation de swap, occupation disque, processus en tête) ; une hypothèse testée et une hypothèse écartée avec la preuve ; une correction minimale (limiter la mémoire du job, déplacer le job hors pic, ajouter de la rotation de logs, redémarrer proprement avec SIGTERM) ; une vérification mesurable (latence redescendue sous X ms, mémoire stabilisée) ; une prévention (alerte sur mémoire disponible, quota disque, ordonnancement du reporting la nuit) et un schéma de décision tel que celui du module.

Quiz éclair

Quiz éclair

1. La méthode USE regarde pour chaque ressource :

A. utilisation, saturation, erreurs
B. uptime, swap, espace
C. usure, signal, état

2. Une charge de 12 sur une machine à 4 cœurs indique :

A. une machine inactive
B. une saturation (file d'attente)
C. une panne réseau

3. Un disque affiche 62 % d'utilisation mais l'application dit « No space left on device ». Piste probable :

Quiz éclair

Quiz éclair (suite)

A. carte réseau défectueuse

B. pas assez de cœurs

C. inodes épuisés ou fichier supprimé encore ouvert

4. 🔁 *Révision mission 09* : Pour demander poliment à un processus de se terminer, on envoie :

A. SIGTERM

B. SIGKILL

C. SIGSTOP

5. 🔁 *Révision mission 09* : Un journal avec niveaux permet de :

A. compresser les fichiers

B. filtrer par gravité

Quiz éclair

Quiz éclair (suite)

C. accélérer le programme

Quiz éclair · réponse

Réponses

1. **A**.
2. **B**.
3. **C**.
4. **A**.
5. **B**.

Je réfléchis

1. Quelle étape de la démarche (vue d'ensemble, hypothèse, test, rapport) as-tu envie de renforcer ?
2. Qu'est-ce qui t'aide à rester calme quand quelque chose ne fonctionne pas ?

À retenir

1. Du général au particulier : charge, CPU, mémoire, disque, réseau, journaux.
2. USE : utilisation, saturation, erreurs.
3. Relier les symptômes aux notions : swap, inodes, droits, signaux.
4. Collecter avant de redémarrer.
5. Un rapport clair vaut une correction réussie.

Félicitations

Tu as terminé **HEF-050 — Systèmes d'exploitation**. Tu sais comment le noyau gère processus, threads, mémoire virtuelle, fichiers, permissions et flux, comment arrêter proprement un service et comment mener une enquête sur un serveur en difficulté.
→ Prochain module : **HEF-060 — Réseaux**

MISSION ACCOMPLIE

+120 XP

Badge débloqué : Enquêteur·rice système

Fin du module HEF-050. Bravo !

