RAPPORT

BTS SIO - E6

Situation professionnelle n°2

Déploiement d'une solution de supervision de disponibilité avec Uptime Kuma

18/06/2026

NOM	DATE	TAMPON
CHERIF SOULEYMANE	18/06/2026	

SOMMAIRE :

1. LE CAHIER DES CHARGES :

1. Contexte et Problématique

Avec la sécurisation des accès distants (Situation professionnelle n°1), l'infrastructure numérique de la PME GSB continue de se développer. Cependant, aucun outil ne permet aujourd'hui de savoir si un service est disponible ou non : une panne n'est détectée que lorsqu'un utilisateur la signale, ce qui retarde la prise en charge et nuit à la qualité de service. La PME souhaite donc disposer d'un tableau de bord de supervision simple, affichant en temps réel l'état de ses services numériques.

1.1. Objectif principal

Mettre en place une solution de supervision permettant de visualiser en temps réel la disponibilité des services numériques de la PME GSB, afin de détecter une panne dès qu'elle survient plutôt que d'attendre qu'un utilisateur la signale.

1.2. Choix techniques

Environnement technologique :

- **Système :** Debian 12 (ARM64), virtualisée sous UTM.
- **Conteneurisation :** Docker, pour déployer rapidement les services sans les installer directement sur le système hôte.
- **Outil de supervision :** Uptime Kuma (open source, tableau de bord web).
- **Service surveillé de démonstration :** Un serveur web Nginx (conteneur Docker), simulant un site vitrine de la PME, dont la panne peut être simulée à volonté.

1.3. Prérequis

- Un hyperviseur installé sur la machine hôte (UTM).
- Une VM Debian 12 opérationnelle, avec accès réseau depuis la machine hôte.
- Des droits d'administration (sudo) sur la VM.

1.4. Compétences

Compétences du référentiel mises en œuvre :

- Déployer et exploiter une solution de supervision réseau.
- Mettre en œuvre une architecture conteneurisée avec Docker.
- Garantir la détection rapide d'un incident de disponibilité.

1.5. Déroulement

Phase 1 : Préparation

- Installation de Docker sur la VM Debian 12.
- Vérification de la connectivité réseau entre la VM et la machine hôte.

Phase 2 : Déploiement

- Lancement du conteneur Nginx (service surveillé) sur le port 8080.
- Lancement du conteneur Uptime Kuma sur le port 3001.

Phase 3 : Configuration

- Création du compte administrateur dans Uptime Kuma.
- Ajout du moniteur HTTP(S) pointant vers le service Nginx.
- Correction de l'URL (problème de résolution réseau Docker identifié et résolu).

Phase 4 : Test et validation

- Simulation d'une panne par arrêt du conteneur Nginx.
- Vérification de la détection automatique en moins de 60 secondes.
- Rétablissement du service et confirmation du retour en ligne.

1.6. Contraintes

- **Contrainte technique :** Solution légère, fonctionnant nativement sur architecture ARM, sans émulation lourde.
- **Contrainte de simplicité :** Outils open source et gratuits, sans licence.
- **Contrainte d'autonomie :** Aucune dépendance à une infrastructure existante (pas de VM ni service tiers à réactiver).
- **Contrainte de réactivité :** Une panne doit être détectée en moins d'une minute (intervalle de vérification court).

2. MISE EN ŒUVRE :

2.1. Architecture et infrastructure

L'infrastructure mise en place repose sur une machine hôte (MacBook Air M2) faisant tourner une machine virtuelle Debian 12 ARM64 via l'hyperviseur UTM. À l'intérieur de cette VM, Docker héberge deux conteneurs indépendants communiquant via le réseau bridge interne de Docker.

Composant	Technologie	Adresse / Port	Rôle
Machine hôte	macOS / UTM	—	Hôte de la virtualisation
VM Debian 12	ARM64 / UTM	192.168.64.2	Système invité (Docker host)
Conteneur Nginx	Docker	172.17.0.1:8080	Service surveillé (cible)
Conteneur Uptime Kuma	Docker	192.168.64.2:3001	Outil de supervision

2.2. Installation de Docker sur Debian 12

Connexion à la VM Debian via UTM, puis installation du moteur Docker depuis les dépôts officiels Debian :

```
debian@debian:~$ sudo apt update

debian@debian:~$ sudo apt install -y docker.io

debian@debian:~$ sudo systemctl enable docker --now
```

> *Docker.io est le paquet Docker disponible nativement dans les dépôts Debian 12. Il prend en charge nativement l'architecture ARM64, ce qui évite toute couche d'émulation coûteuse en performance.*

2.3. Déploiement des conteneurs

2.3.1. Lancement du conteneur Nginx

Nginx est déployé comme service de démonstration. Il simule un service web interne de la PME dont on souhaite surveiller la disponibilité.

```
sudo docker run -d --name nginx -p 8080:80 nginx:latest
```

Vérification immédiate de la réponse HTTP depuis la VM :

```
curl -v http://localhost:8080

< HTTP/1.1 200 OK

< Server: nginx/1.31.2

...

<h1>Welcome to nginx!</h1>
```

2.3.2. Lancement du conteneur Uptime Kuma

Uptime Kuma est déployé avec un volume persistant pour conserver la configuration entre les redémarrages :

```
sudo docker run -d --name uptime-kuma \
  -p 3001:3001 \
  -v uptime-kuma:/app/data \
  louislam/uptime-kuma:latest
```

Vérification que les deux conteneurs sont actifs :

```
sudo docker ps
CONTAINER ID   IMAGE                      STATUS
a1b2c3d4e5f6   louislam/uptime-kuma:1     Up 2 minutes (healthy)
f6e5d4c3b2a1   nginx:latest               Up 2 minutes
```

2.4. Configuration d'Uptime Kuma

2.4.1. Accès au tableau de bord

Le tableau de bord est accessible depuis la machine hôte (Mac) via le navigateur. L'accès se fait en HTTP car aucun certificat TLS n'est configuré dans ce contexte de démonstration :

```
http://192.168.64.2:3001
```

2.4.2. Création du compte administrateur

Lors du premier accès, Uptime Kuma invite à créer un compte administrateur. Ce compte protège l'accès au tableau de bord et à la configuration des moniteurs.

Identifiant	Valeur
Nom d'utilisateur	`admin`
Mot de passe	`Uptime123`

2.4.3. Création du moniteur Nginx

Dans le tableau de bord, on crée un nouveau moniteur de type HTTP(S) visant le service Nginx. Les paramètres retenus sont les suivants :

Paramètre	Valeur configurée
Nom du moniteur	`Nginx Test`
Type	`HTTP(S)`
URL	`http://172.17.0.1:8080`
Intervalle de vérification	`60 secondes`
Code de statut attendu	`200 OK`

2.5. Problème rencontré et solution apportée

Problème : Lors de la première configuration, le moniteur affichait « Hors ligne » de manière persistante, alors même que le conteneur Nginx fonctionnait correctement.

> *Erreur observée : [Nginx Test] [DOWN] connect ECONNREFUSED 192.168.64.2:8080*

Analyse : Uptime Kuma s'exécute lui-même à l'intérieur d'un conteneur Docker. Depuis cet espace isolé, l'IP de la VM (192.168.64.2) attribuée par UTM n'est pas accessible directement. En revanche, l'interface bridge Docker (docker0) écoute sur 172.17.0.1 et est atteignable par tous les conteneurs.

Vérification :

```
# Test depuis l'intérieur du conteneur Uptime Kuma

sudo docker exec -it uptime-kuma curl http://172.17.0.1:8080

# Résultat : <h1>Welcome to nginx!</h1>  -> Accessible !
```

Solution : Modifier l'URL du moniteur pour utiliser l'IP du bridge Docker (172.17.0.1) plutôt que l'IP UTM de la VM.

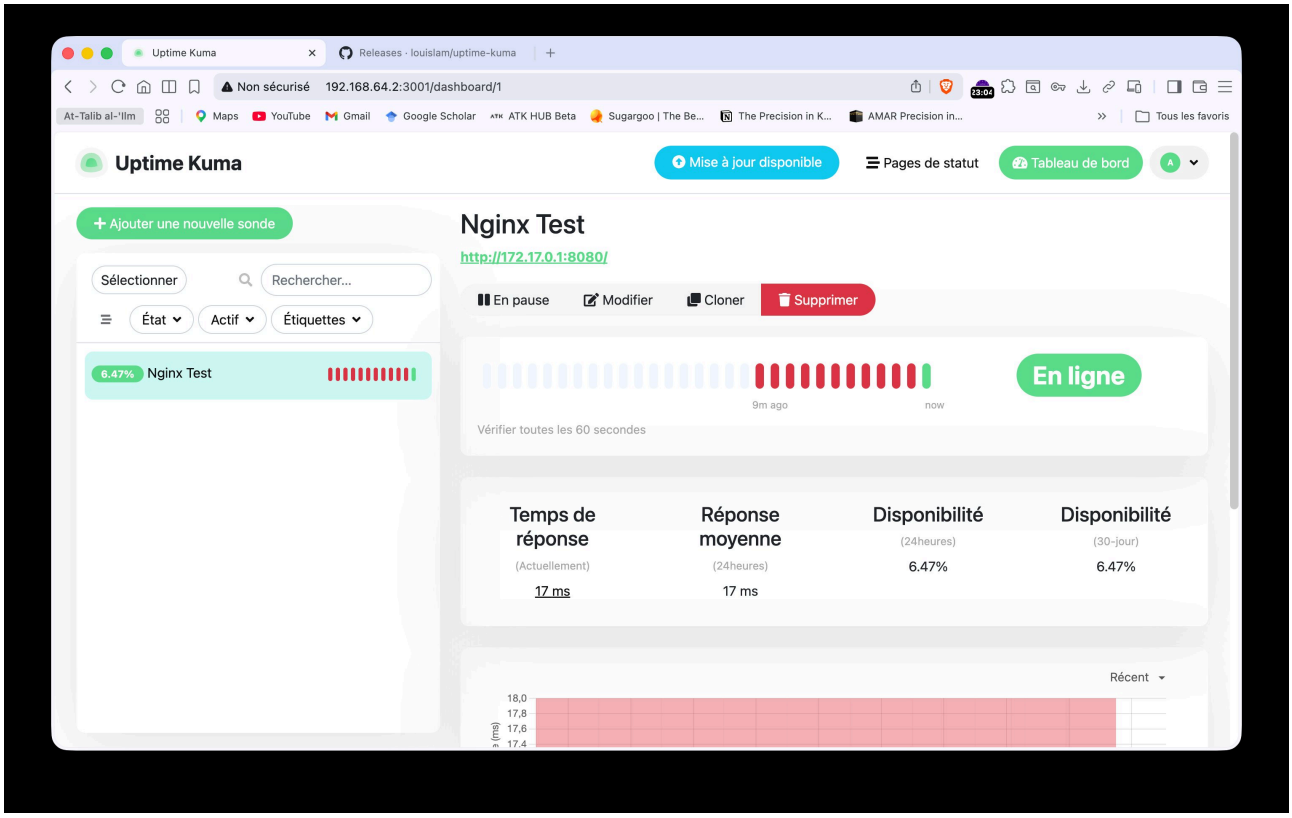
	URL	Résultat
Avant	`http://192.168.64.2:8080`	ECONNREFUSED
Après	`http://172.17.0.1:8080`	200 OK - En ligne ✓

3. TESTS ET RECETTES :

3.1. Tests et Vérifications

3.1.1. Test de disponibilité — Service EN LIGNE

Après correction de l'URL du moniteur, Uptime Kuma détecte correctement que le service Nginx répond. Le tableau de bord affiche l'indicateur « En ligne » en vert, avec un temps de réponse de 17 ms.



Dashboard Uptime Kuma — Moniteur Nginx Test en ligne (200 OK, 17 ms)

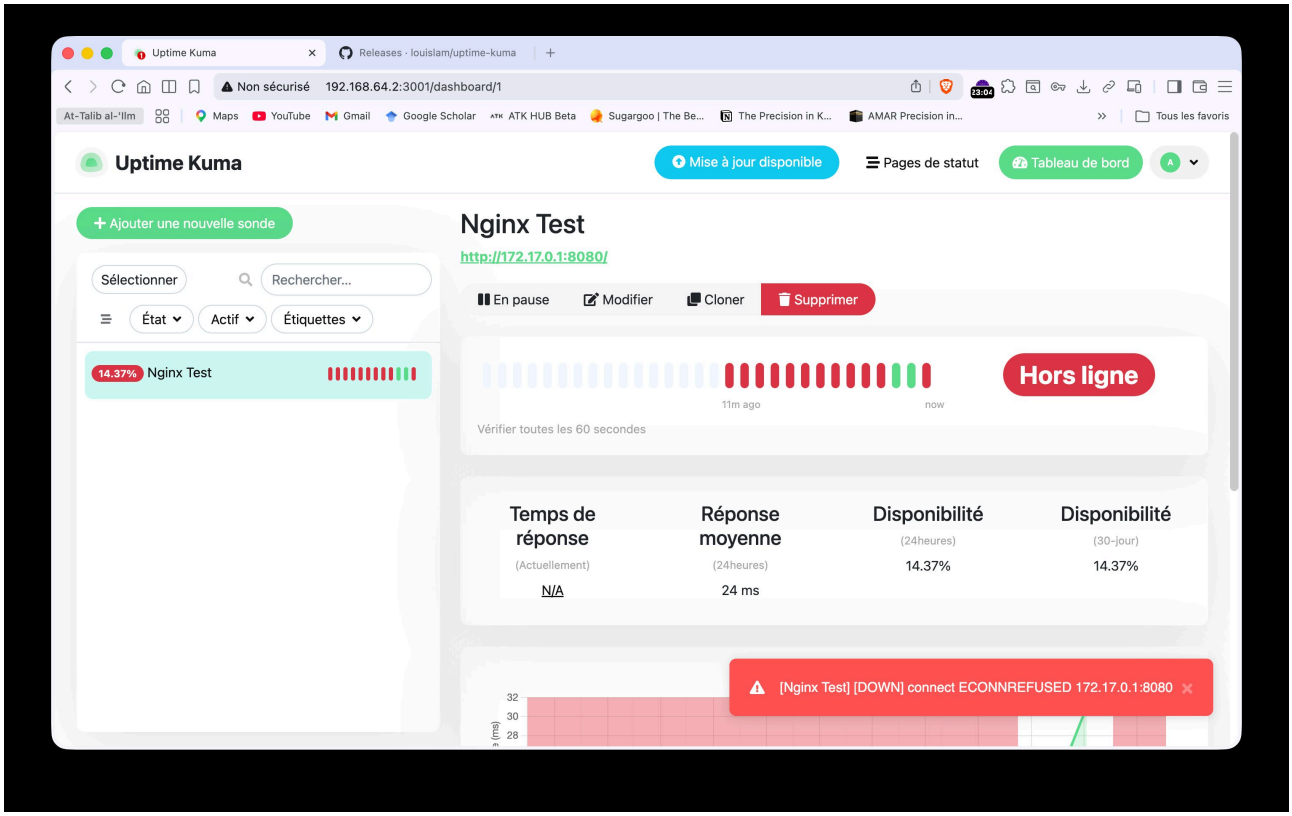
3.1.2. Simulation de panne — Arrêt du service

La panne est simulée en arrêtant manuellement le conteneur Nginx depuis le terminal de la VM Debian. Cette opération reproduit fidèlement le comportement d'un service qui tombe en production.

```
debian@debian:~$ sudo docker stop nginx
nginx
debian@debian:~$
```

Commande sudo docker stop nginx — arrêt du service surveillé

Dans les 60 secondes suivant l'arrêt, Uptime Kuma détecte automatiquement l'indisponibilité du service. L'indicateur bascule en « Hors ligne » rouge, et une notification apparaît en bas de l'écran : [DOWN] connect ECONNREFUSED 172.17.0.1:8080.



Dashboard Uptime Kuma — Détection de la panne (ECONNREFUSED, Hors ligne)

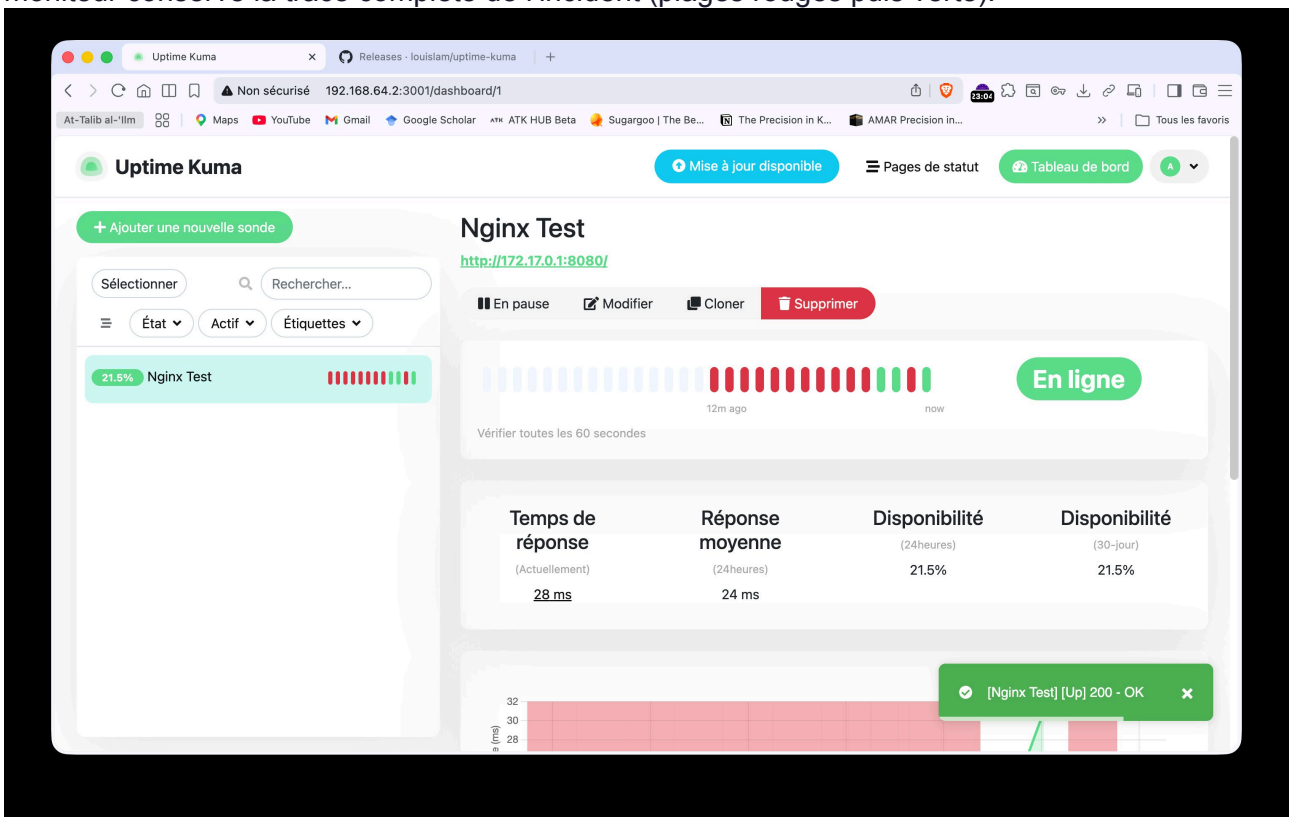
3.1.3. Rétablissement du service

Le service est rétabli par redémarrage du conteneur Nginx. Cette étape valide que la supervision détecte non seulement la panne, mais aussi le retour à la normale.

```
debian@debian:~$ sudo docker start nginx
nginx
debian@debian:~$
```

Commande sudo docker start nginx — rétablissement du service

Dans les 60 secondes suivant le redémarrage, le moniteur repasse « En ligne » vert. La notification [Up] 200 - OK confirme que le service répond à nouveau correctement. L'historique du moniteur conserve la trace complète de l'incident (plages rouges puis verte).



Dashboard Uptime Kuma — Retour en ligne confirmé ([Up] 200 - OK)

> *Bilan du test : la chaîne complète UP → DOWN → UP est validée. Délai de détection < 60 secondes dans les deux sens. L'historique Uptime Kuma conserve la trace de l'incident.*

4. POINTS FUTURS À DÉVELOPPER :

La solution déployée couvre le besoin de base de supervision. Plusieurs axes d'amélioration sont identifiés pour renforcer la solution en conditions réelles :

- **Notifications actives :** Configurer les alertes par e-mail ou via Slack/Teams afin qu'une notification soit envoyée automatiquement à l'administrateur système dès qu'un service tombe.
- **Surveillance multi-cibles :** Ajouter des moniteurs pour le VPN OpenVPN (E6 n°1), le serveur de fichiers Windows Server 2022, et d'autres services critiques de la PME.
- **Accès sécurisé HTTPS :** Configurer HTTPS sur l'interface Uptime Kuma via un reverse proxy Nginx avec un certificat auto-signé ou Let's Encrypt pour éviter l'avertissement « Non sécurisé ».
- **Réseau Docker dédié :** Utiliser un réseau Docker dédié (docker network create) pour permettre aux conteneurs de communiquer par nom plutôt que par IP. Cela évite le problème rencontré avec l'IP bridge et facilite la maintenance.
- **Sauvegarde des données :** S'assurer que le volume uptime-kuma est sauvegardé régulièrement pour ne pas perdre la configuration des moniteurs en cas de réinstallation.
- **Haute disponibilité :** Déployer Uptime Kuma sur une machine distincte de celles qu'il surveille, afin qu'une panne de l'hôte ne rende pas la supervision elle-même indisponible.

5. BILAN :

Cette situation professionnelle a permis de mettre en place une solution de supervision simple, opérationnelle et adaptée aux contraintes de la PME GSB. L'ensemble du déploiement (VM, Docker, conteneurs, configuration) a été réalisé de manière autonome sur architecture ARM sans émulation.
Le principal problème rencontré — la communication réseau inter-conteneurs Docker — a été diagnostiqué et résolu de manière méthodique, en comprenant le fonctionnement de l'interface bridge docker0 (172.17.0.1). Cette résolution illustre l'importance de maîtriser l'architecture réseau des environnements conteneurisés.
Les tests menés ont validé l'ensemble du cycle de supervision : détection automatique d'une panne en moins de 60 secondes, historique de disponibilité consultable, et détection du retour à la normale. L'outil Uptime Kuma s'est avéré particulièrement adapté : léger, open source, multiplateforme et sans licence.

Objectif	Statut
Déployer Docker sur Debian 12 ARM64	✅ **Atteint**
Lancer Uptime Kuma et Nginx en conteneurs	✅ **Atteint**
Configurer un moniteur HTTP(S)	✅ **Atteint**
Détecter une panne en moins de 60 secondes	✅ **Atteint**
Détecter le retour en ligne automatiquement	✅ **Atteint**
Comprendre le réseau bridge Docker	✅ **Atteint (problème résolu)**

Cette situation professionnelle s'inscrit dans la continuité de la n°1 (VPN pfSense/OpenVPN) et contribue à bâtir une infrastructure PME progressivement plus robuste, sécurisée et supervisée. Les points futurs identifiés (notifications, multi-cibles, HTTPS) constituent la feuille de route naturelle pour une mise en production.