PortailSIO

NOUVELLE FORMULE

TUTORIEL

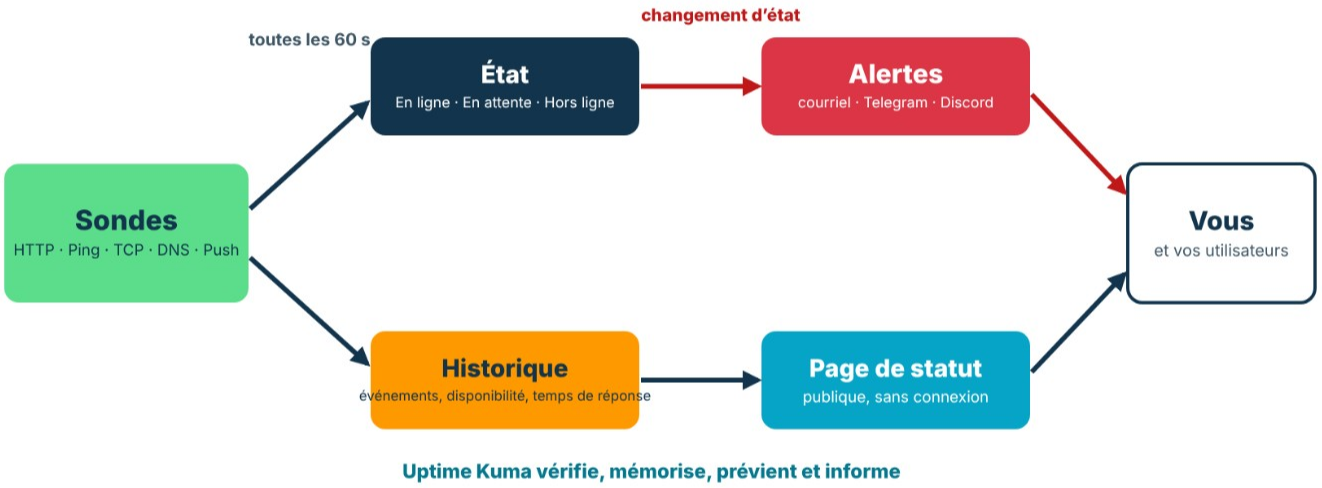
DOCKER COMPOSE · UPTIME KUMA 2

# Installer avec Docker Compose

# KUMA

NIVEAU
Intermédiaire

## Des sondes, des alertes et une page de statut publique





1 Préparer le serveur Debian 13

2 Installer Docker Engine et Docker Compose

3 Écrire le projet Compose et démarrer Uptime Kuma

4 Première connexion et premières sondes

5 Sondes avancées et alertes

6 Page de statut, maintenance et exploitation



CC BY-NC

Utilisation commerciale interdite

© PortailSIO – 11/2026


**Ce tutoriel en ligne**

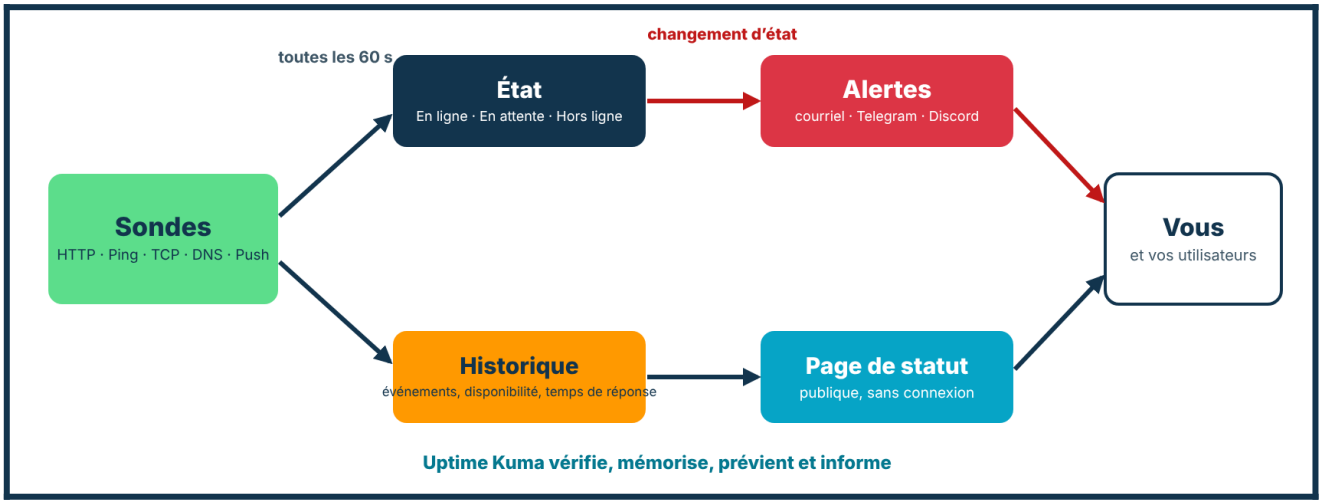
Scannez le QR code. Tous les tutoriels :

portail.sio-ndlp.fr/tutoriels.php

Ce tutoriel a été réalisé avec l'hyperviseur © Proxmox VE. Il peut être réalisé dans un environnement de virtualisation personnel à l'aide des logiciels © VirtualBox ou © VMware Player. Il **suppose que vous disposez d'une machine virtuelle Debian 13** (installation minimale avec serveur SSH) et, pour la partie « sondes », de quelques machines à surveiller (ici celles des tutoriels Zabbix et Active Directory).

**Uptime Kuma** est un outil libre (licence MIT) de **supervision de disponibilité** que l'on héberge soi-même. Il vérifie à intervalle régulier que vos services **répondent** (site web, serveur, port, nom DNS, certificat…), vous **prévient** dès qu'un service tombe (courriel, Telegram, Discord, Teams, webhook…) et peut publier une **page de statut** lisible par vos utilisateurs. La version **2** (2.5 au moment de la rédaction) est utilisée ici, avec l'interface en français.

Son principe tient en une phrase : **une sonde interroge une cible toutes les 60 secondes ; si elle ne répond pas comme prévu, l'état passe à « Hors ligne » et une alerte part**. Les sondes alimentent un historique (disponibilité, temps de réponse) et, si on le souhaite, une page de statut publique :

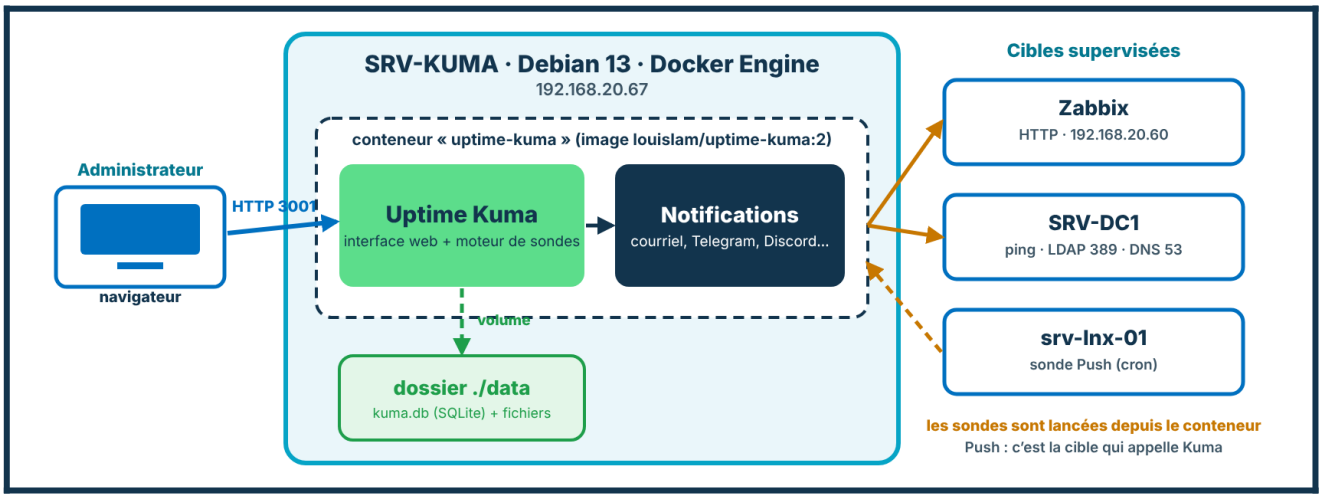



## Uptime Kuma ou Zabbix ?

| | Uptime Kuma | Zabbix |
|---|---|---|
| **Question posée** | « Est-ce que ça répond ? » (disponibilité) | « Dans quel état est la machine ? » (processeur, mémoire, disques, services…) |
| **Installation** | Un conteneur, 10 minutes | Serveur, base de données, interface et agents à déployer |
| **Agent sur les machines** | Aucun (sauf sonde Push facultative) | Agent installé sur chaque hôte |
| **Page de statut publique** | Oui, intégrée | Non |
| **Quand l'utiliser** | Surveiller des sites, des ports, des DNS, des sauvegardes ; informer les utilisateurs | Superviser finement un parc de serveurs et prévoir les pannes |

Les deux outils sont **complémentaires** : Zabbix mesure, Uptime Kuma vérifie de l'extérieur que le service est réellement rendu.

Dans ce tutoriel, nous installons Uptime Kuma **avec Docker Compose** : l'application tourne dans **un seul conteneur** et toute l'installation se décrit dans **un seul fichier** (`compose.yaml`). Les données (base, réglages) sont conservées dans le dossier `data` du serveur, ce qui rend la sauvegarde et la mise à jour très simples :




## Ce que l'on met en place

- **Une installation** Docker Compose d'Uptime Kuma 2 sur Debian 13, avec son dossier de données
- **Des sondes** de plusieurs types : HTTP, ping, port TCP, DNS, certificat, et sonde Push pour une tâche planifiée
- **Des alertes** par courriel (et le principe pour Telegram ou Discord), testées par une vraie panne
- **Une page de statut** publique et une **maintenance planifiée** qui évite les fausses alertes
- **L'exploitation** : sauvegarde du dossier `data`, mise à jour de l'image, publication en HTTPS derrière Traefik

Dans le cadre de ce tutoriel, nous utilisons les machines suivantes (VLAN SERVEURS, réseau 192.168.20.0 /24) :

| Machine | Rôle | Système | Adresse IP |
|---|---|---|---|
| **SRV-KUMA** | Serveur Docker exécutant Uptime Kuma (nom `srv-kuma`) | Debian 13 | 192.168.20.67 |
| **SRV-ZBX** | Cible : interface web de Zabbix (tutoriel Zabbix) | Debian 13 | 192.168.20.60 |
| **srv-lnx-01** | Cible : exécute une tâche de sauvegarde (sonde Push) | Debian 13 | 192.168.20.61 |
| **SRV-DC1** | Cible : contrôleur de domaine (ping, LDAP, DNS) | Windows Server 2025 | 192.168.20.250 |

Pour **SRV-KUMA**, prévoyez au minimum **1 vCPU, 2 Go de mémoire et 10 Go de disque** : Uptime Kuma est très léger. Ces valeurs, comme les adresses, les noms et les mots de passe d'exemple, sont à adapter à votre environnement. Les commandes sont exécutées avec un compte autorisé à utiliser `sudo` (ou directement en `root`, sans le préfixe `sudo`).

# 01

ÉTAPE 1 SUR 6

# Préparer le serveur Debian 13

Docker n'a besoin que d'un système à jour, d'un nom, d'une adresse fixe et de l'heure exacte. L'heure compte ici plus qu'ailleurs : les alertes, les historiques et les dates d'expiration des certificats s'appuient dessus. Ouvrez une session (console Proxmox ou SSH) sur **SRV-KUMA** puis effectuez les manipulations suivantes :

1. Mettez le système à jour : `sudo apt update && sudo apt full-upgrade -y`
2. Donnez son nom au serveur et déclarez-le dans le fichier `hosts` (nom court et nom complet) :

```
TERMINAL
sudo hostnamectl set-hostname srv-kuma
echo "192.168.20.67  srv-kuma.sio.local  srv-kuma" | sudo tee -a /etc/hosts
```

3. Réglez le fuseau horaire : `sudo timedatectl set-timezone Europe/Paris`
4. Donnez au serveur une **adresse IP fixe** (adaptez le nom de la carte, ici `ens18`, avec `ip -br address`) : éditez `/etc/network/interfaces` puis redémarrez le réseau avec `sudo systemctl restart networking` :

```
/etc/network/interfaces
auto ens18
iface ens18 inet static
    address 192.168.20.67/24
    gateway 192.168.20.254
```

5. Indiquez les **DNS du domaine** dans `/etc/resolv.conf` : les sondes DNS et HTTPS de ce tutoriel en ont besoin pour résoudre les noms du domaine (par exemple `srv-dc1.sio.local`) :

```
/etc/resolv.conf
search sio.local
nameserver 192.168.20.250
nameserver 192.168.20.251
```

6. Vérifiez le résultat : nom, fuseau, heure synchronisée, adresse IP et résolution d'un nom du domaine :

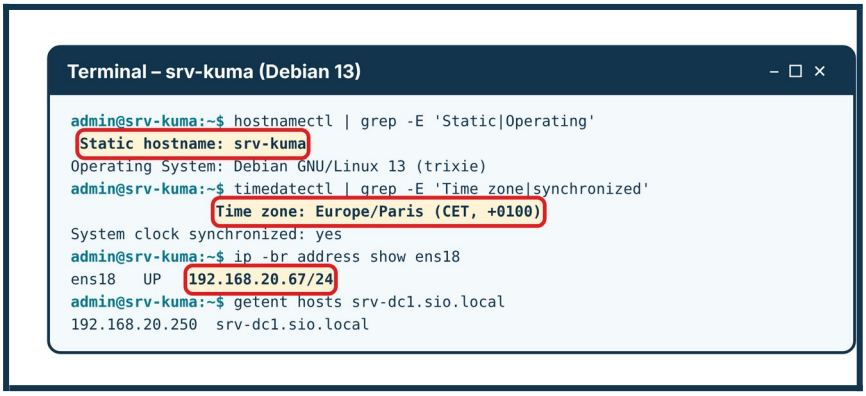



*Figure 1 – Contrôle de la préparation du serveur*

**À SAVOIR**

Si « System clock synchronized » indique « no », installez un service de temps (`sudo apt install -y chrony`). Une horloge fausse décale l'historique et fausse le calcul des échéances de certificats.

**✔ POINT DE CONTRÔLE**

- `hostnamectl` affiche le nom **srv-kuma** et Debian GNU/Linux **13**
- L'adresse IP **192.168.20.67** est fixe
- Le fuseau horaire est **Europe/Paris** et l'horloge est **synchronisée**
- `getent hosts srv-dc1.sio.local` renvoie l'adresse du contrôleur

# 02

ÉTAPE 2 SUR 6

# Installer Docker Engine et Docker Compose

**Docker Engine** est le moteur qui exécute les conteneurs ; **Docker Compose** est son extension qui décrit toute une application (plusieurs conteneurs, un réseau, des volumes) dans **un seul fichier** et la pilote avec une seule commande. Docker s'installe depuis le **dépôt officiel de Docker** : les paquets de Debian sont plus anciens et n'incluent pas Compose. Sur le serveur :

## A – Ajouter le dépôt officiel de Docker

1 Retirez d'éventuels paquets concurrents (sur une installation neuve, il n'y a rien à retirer) puis installez les outils de téléchargement :

TERMINAL

```
sudo apt remove $(dpkg --get-selections docker.io docker-compose docker-doc \
    podman-docker containerd runc 2>/dev/null | cut -f1)
sudo apt update && sudo apt install -y ca-certificates curl
```

2 Enregistrez la clé de signature du dépôt :

TERMINAL

```
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg \
    -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

3 Déclarez le dépôt (format « deb822 »). Le nom de la version de Debian, `trixie`, est lu automatiquement dans le système :

TERMINAL

```
sudo tee /etc/apt/sources.list.d/docker.sources > /dev/null <<EOF
Types: deb
URIs: https://download.docker.com/linux/debian
Suites: $(. /etc/os-release && echo "$VERSION_CODENAME")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
sudo apt update
```

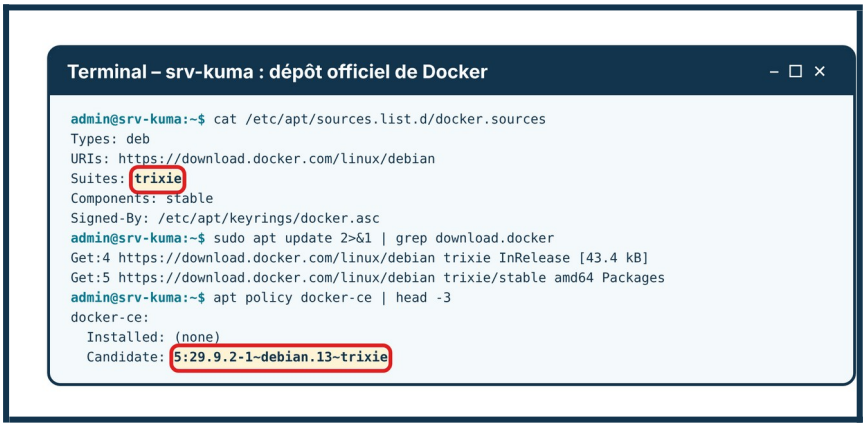
## B – Installer Docker Engine et le plugin Compose

1 Installez les cinq paquets de Docker :

TERMINAL

```
sudo apt install -y docker-ce docker-ce-cli containerd.io \
    docker-buildx-plugin docker-compose-plugin
```

| Paquet | Rôle |
|---|---|
| **docker-ce** | Le moteur (service `docker`) qui crée et exécute les conteneurs |
| **docker-ce-cli** | La commande `docker` |
| **containerd.io** | Le composant de bas niveau qui exécute réellement les conteneurs |
| **docker-buildx-plugin** | L'outil de construction d'images (`docker build`) |
| **docker-compose-plugin** | **Docker Compose** : la commande `docker compose` |




**À SAVOIR**

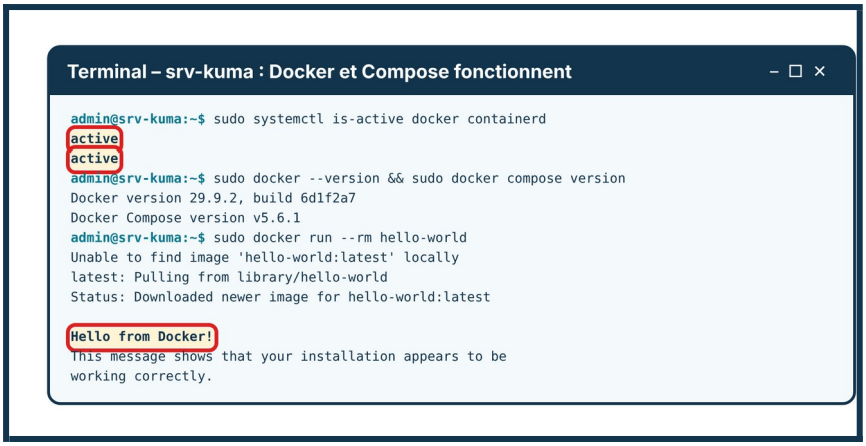
Si la ligne « Candidate » indique une version venant de Debian (`docker.io`), le dépôt n'est pas pris en compte : relisez le fichier `docker.sources` (indentation, nom de la version).

*Figure 2 – Dépôt officiel de Docker et version candidate*

## C – Vérifier que Docker fonctionne

1 Le service démarre tout seul à l'installation. Contrôlez-le, affichez les versions, puis lancez le conteneur de test officiel :

```
TERMINAL
sudo systemctl is-active docker containerd
sudo docker --version && sudo docker compose version
sudo docker run --rm hello-world
```




**À SAVOIR**

Le message « Hello from Docker! » prouve que le moteur sait télécharger une image sur Internet et exécuter un conteneur. L'option `--rm` supprime ensuite ce conteneur de test.

*Figure 3 – Docker et Compose en état de marche*

**À SAVOIR**

Les commandes de ce tutoriel utilisent `sudo docker`. Ajouter votre compte au groupe `docker` permettrait de s'en passer, mais ce groupe équivaut à des droits **root** sur le serveur : on l'évite en production. Autre point d'attention : les ports **publiés** par Docker (`ports:`) ne sont pas filtrés par un pare-feu configuré sur l'hôte (ufw, firewalld) ; limitez au besoin leur accès par une règle réseau en amont.

✔ **POINT DE CONTRÔLE**

- Le dépôt `download.docker.com` apparaît dans `apt update` et `apt policy docker-ce` propose une version **5:29** (ou supérieure)
- `systemctl is-active docker` répond **active**
- `docker compose version` affiche une version de **Docker Compose**
- `docker run --rm hello-world` affiche « **Hello from Docker!** »

ÉTAPE 3 SUR 6

# 03 Écrire le projet Compose et démarrer Uptime Kuma

Un **projet Compose** est un dossier qui contient le fichier `compose.yaml` : il décrit le conteneur, ses réglages, ses ports et l'endroit où il garde ses données. Ici, le projet ne contient qu'**un seul service**.

## A – Créer le dossier et le fichier compose.yaml

1 Créez le dossier du projet, puis le fichier avec `sudo nano /opt/uptime-kuma/compose.yaml` :

TERMINAL

```
sudo mkdir -p /opt/uptime-kuma
cd /opt/uptime-kuma
```

/opt/uptime-kuma/compose.yaml

```
name: uptime-kuma

services:
  uptime-kuma:
    image: louislam/uptime-kuma:2
    container_name: uptime-kuma
    restart: unless-stopped
    volumes:
      - ./data:/app/data
    ports:
      - "3001:3001"
```

Le fichier est écrit en **YAML** : l'**indentation** (deux espaces, jamais de tabulation) donne la structure. Rôle de chaque ligne :

| Ligne | Rôle |
|---|---|
| `image: louislam/uptime-kuma:2` | L'image officielle, **branche 2**. La balise `2-slim` est une variante allégée sans base MariaDB intégrée ; évitez `latest`, déclarée obsolète et restée sur l'ancienne version 1 |
| `container_name` | Donne au conteneur un nom fixe, plus simple à citer dans les commandes (`docker logs uptime-kuma`) |
| `restart: unless-stopped` | Le conteneur redémarre tout seul après un redémarrage du serveur, sauf si vous l'avez arrêté vous-même |
| `./data:/app/data` | Le dossier `data` du projet contient **toutes les données** (base, réglages). Il survit à la recréation du conteneur. Évitez un dossier partagé en **NFS**, non pris en charge |
| `3001:3001` | Publie l'interface sur le **port 3001** du serveur. Pour la réserver à un proxy installé sur le même serveur, écrivez `127.0.0.1:3001:3001` |

**REMARQUE IMPORTANTE**

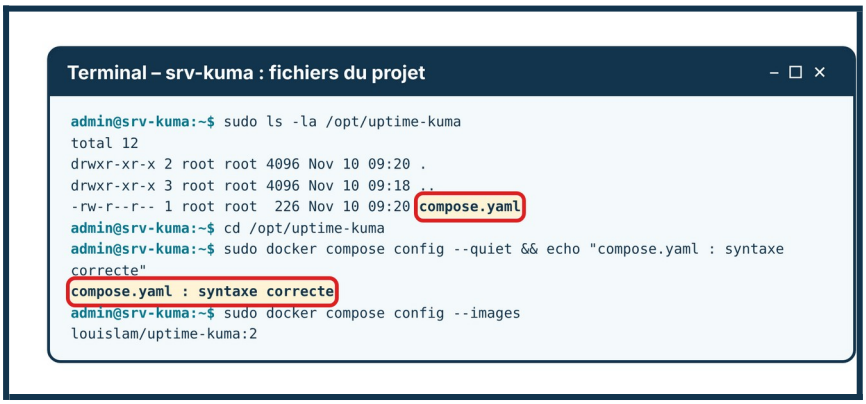
Dans un fichier YAML, une **erreur d'indentation** change le sens du fichier ou l'invalide. Évitez de copier-coller depuis un document qui remplace les espaces par des caractères spéciaux : la commande de validation ci-dessous repère la plupart des erreurs.

## B – Valider le projet et télécharger l'image

1 Validez la syntaxe, affichez l'image qui sera utilisée, puis téléchargez-la :

TERMINAL

```
sudo docker compose config --quiet && echo "compose.yaml : syntaxe correcte"
sudo docker compose config --images
```




**À SAVOIR**

`docker compose config` relit le fichier, le vérifie et l'affiche tel que Docker le comprendra. Sans message d'erreur, la syntaxe est valide.

*Figure 4 – Fichiers du projet et validation de la syntaxe*

TERMINAL

```
sudo docker compose pull
```

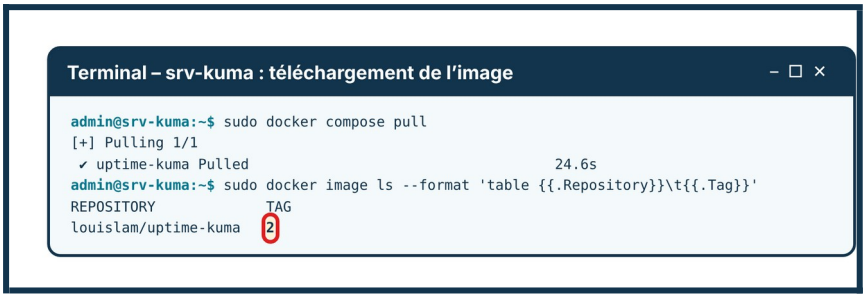



*Figure 5 – Téléchargement de l'image*

**À SAVOIR**

Une **image** est le modèle en lecture seule de l'application ; le **conteneur** en est l'instance en cours d'exécution. Le téléchargement n'est nécessaire qu'une fois par version.

## C – Démarrer le conteneur

**1** Lancez le projet en arrière-plan, puis contrôlez l'état du conteneur et le port d'écoute :

**TERMINAL**

```
sudo docker compose up -d
sudo docker compose ps
ss -tlnp | grep 3001
```

Le premier démarrage prend **quelques dizaines de secondes** : l'état passe de « starting » à « **healthy** » quand l'application répond. Docker a créé tout seul le dossier `data` à côté du fichier `compose.yaml`.

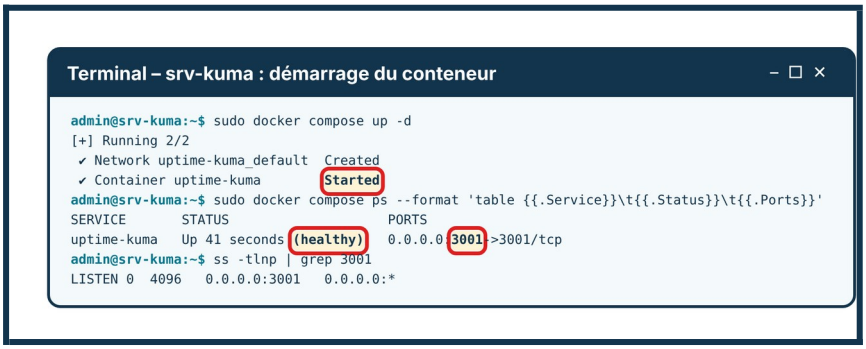



*Figure 6 – Démarrage du conteneur et état de santé*

**À SAVOIR**

« (healthy) » provient du contrôle de santé intégré à l'image : Docker interroge régulièrement l'application. Si l'état reste « unhealthy », consultez le journal (commande suivante).

Si le conteneur ne passe pas à l'état « healthy », lisez son journal :

**TERMINAL**

```
sudo docker compose logs --tail 20 uptime-kuma
```

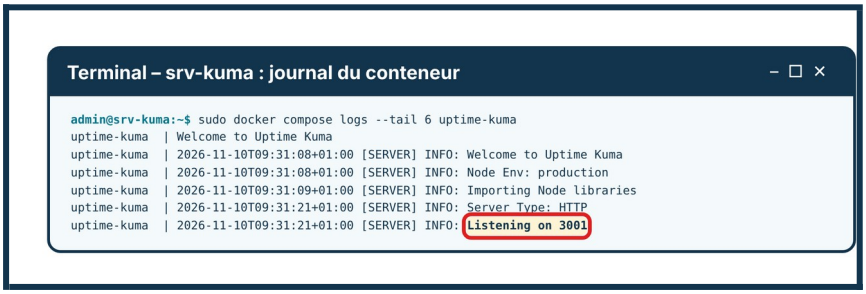



*Figure 7 – Journal du conteneur uptime-kuma*

**À SAVOIR**

La ligne « Listening on 3001 » indique que le serveur web est prêt. Le texte exact des messages varie d'une version à l'autre.

**À SAVOIR**

Docker **publie les ports par ses propres règles de pare-feu**, qui passent avant celles d'`ufw` : un port publié avec `ports:` reste joignable même si `ufw` le bloque. Ici, le port 3001 doit être joignable du réseau d'administration ; pour le restreindre, publiez-le sur `127.0.0.1` (voir plus loin la publication derrière un proxy) ou filtrez en amont.

✔ POINT DE CONTRÔLE

- `docker compose config --quiet` ne renvoie **aucune erreur**
- `docker compose ps` indique **uptime-kuma** « Up … (healthy) » avec le port **3001** publié
- `ss -tlnp | grep 3001` montre le port à l'écoute
- Le dossier `/opt/uptime-kuma/data` existe

# 04 ÉTAPE 4 SUR 6 Première connexion et premières sondes

## A – Choisir la base de données et créer le compte administrateur

**1** Depuis un poste du réseau, ouvrez un navigateur sur `http://192.168.20.67:3001`. L'**assistant de premier démarrage** vous demande d'abord : « **Quelle base de données souhaitez-vous utiliser ?** ». Choisissez **SQLite** puis **Continuer** :




*Figure 8 – Assistant : choix de la base de données*

À SAVOIR

**SQLite** : un simple fichier dans `data`, parfait pour un laboratoire ou quelques dizaines de sondes, et très simple à sauvegarder. **Embedded MariaDB** (image complète) convient mieux à une installation plus chargée ; **MariaDB/MySQL** se connecte à un serveur existant. Le choix ne se change pas simplement ensuite : réfléchissez avant de valider.

**2** Créez le **compte administrateur** : nom d'utilisateur, mot de passe, répétition du mot de passe, puis **Créer** :

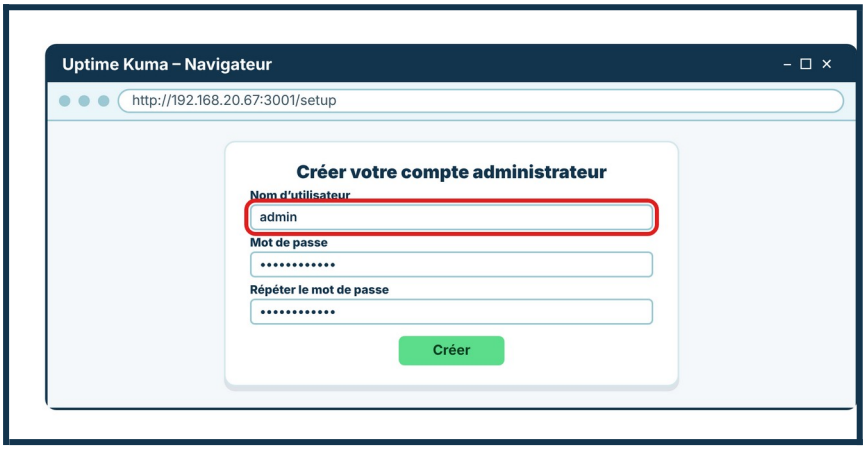



*Figure 9 – Création du compte administrateur*

À SAVOIR

Choisissez un **mot de passe solide** : ce compte contrôle toutes les sondes et les notifications. Il n'y a pas de mot de passe par défaut, c'est à ce moment que vous le définissez.

## B – Découvrir le tableau de bord

Après connexion, le **Tableau de bord** s'affiche. À gauche, la liste des **sondes** (vide pour l'instant) ; à droite, le résumé (nombre de sondes en ligne, hors ligne, en maintenance, inconnues, en pause) et les **événements importants**. Le bouton **Ajouter une nouvelle sonde** est en haut à droite :

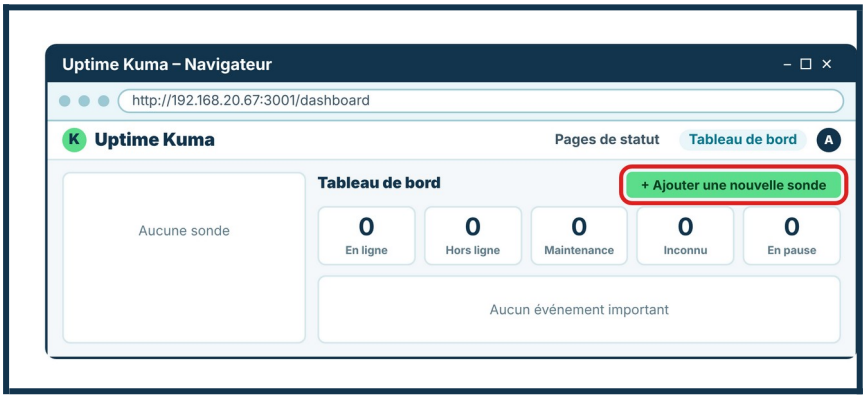



*Figure 10 – Tableau de bord vide*

**À SAVOIR**

Les deux entrées du menu du haut : **Tableau de bord** (vos sondes) et **Pages de statut** (publication). Les paramètres, les notifications et les maintenances s'ouvrent depuis le menu du compte, en haut à droite.

## C – Première sonde : une page web (HTTP)

La sonde HTTP(s) charge une adresse web et vérifie le **code de réponse**. Prenons l'interface web de Zabbix (tutoriel précédent) :

1. Cliquez sur **Ajouter une nouvelle sonde**, puis renseignez le formulaire (les autres champs gardent leur valeur par défaut) :

| Champ | Valeur | Rôle |
|---|---|---|
| **Type de sonde** | `HTTP(s)` | Charge l'URL et contrôle le code de réponse |
| **Nom d'affichage** | `Zabbix – interface web` | Le nom qui apparaîtra dans les listes, alertes et pages de statut |
| **URL** | `http://192.168.20.60` | L'adresse à tester |
| **Intervalle de vérification** | `60` | Une vérification par minute (en secondes) |
| **Tentatives** | `1` | Nouvelles tentatives avant de déclarer « Hors ligne » : évite les fausses alertes sur un raté isolé |
| **Intervalle entre chaque nouvelle tentative** | `60` | Délai entre deux tentatives |
| **Codes HTTP acceptés** | `200-299` | Tout autre code (404, 500…) est une panne |

2 Cliquez sur **Sauvegarder** :

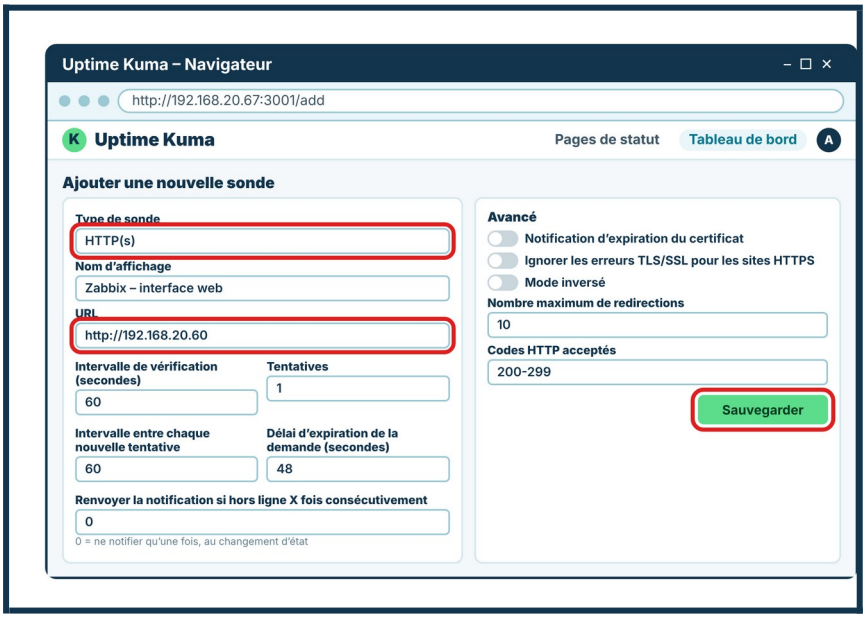



*Figure 11 – Formulaire d'ajout d'une sonde HTTP*

**À SAVOIR**

Avec **Tentatives = 1**, une première réponse manquée place la sonde « **En attente** » (orange) ; la seconde la passe « **Hors ligne** » (rouge) et déclenche l'alerte. Plus vous augmentez les tentatives, moins vous avez de fausses alertes, mais plus la détection est lente.

La sonde se lance immédiatement. En quelques secondes, la fiche affiche l'état **En ligne**, le **temps de réponse**, la **disponibilité** (24 heures, 30 jours, 1 an), les dernières vérifications (barre de petits traits verts) et l'historique des événements :

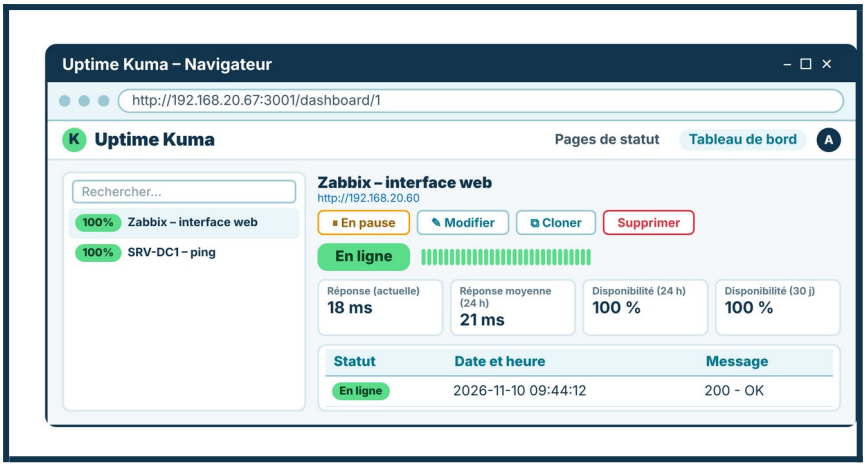



*Figure 12 – Détail d'une sonde en ligne*

**À SAVOIR**

Chaque trait de la barre est une vérification : vert = en ligne, rouge = hors ligne, orange = en attente, bleu = maintenance. Les boutons **En pause**, **Modifier**, **Cloner** et **Supprimer** pilotent la sonde ; **Cloner** accélère la création de sondes semblables.

## D – Deuxième sonde : un ping

1 Cliquez de nouveau sur **Ajouter une nouvelle sonde**. Type : **Ping**. Nom d'affichage : `SRV-DC1 — ping`. Nom d'hôte / adresse IP : `192.168.20.250`. Laissez le reste par défaut et **Sauvegardez**

2 Contrôlez que les deux sondes sont **En ligne** dans la liste de gauche

## Les types de sonde à connaître

| Type | Ce qu'il vérifie | Exemple |
|---|---|---|
| **HTTP(s)** | Une URL répond avec un code accepté | Interface web de Zabbix, GLPI, Nextcloud |
| **HTTP(s) – Mot-clé** | La page répond **et contient** un mot attendu | Le texte « Connexion » sur une page de login |
| **Ping** | L'hôte répond au ping (ICMP) | Un serveur, un routeur |
| **TCP Port** | Un port accepte une connexion | LDAP 389, RDP 3389, SMTP 25 |
| **DNS** | Un nom est résolu avec la valeur attendue | `srv-dc1.sio.local` → `192.168.20.250` |
| **Push** | C'est la cible qui **appelle Kuma** régulièrement | Une sauvegarde nocturne |

✔ POINT DE CONTRÔLE

- Vous avez créé le **compte administrateur** et vous vous connectez à l'interface
- La sonde **Zabbix – interface web** est **En ligne**, avec un temps de réponse affiché
- La sonde **SRV-DC1 – ping** est **En ligne**
- Vous savez expliquer les champs **Intervalle de vérification** et **Tentatives**

# 05

ÉTAPE 5 SUR 6

# Sondes avancées et alertes

## A – Un port TCP : l'annuaire LDAP

Une machine peut répondre au ping tout en ayant un service arrêté. La sonde **TCP Port** teste le service lui-même :

1. Ajoutez une sonde. Type : **TCP Port**. Nom d'affichage : `SRV-DC1 – LDAP 389`. Nom d'hôte : `192.168.20.250`. Port : `389`. **Sauvegardez**

## B – Une sonde DNS

Un DNS en panne est invisible pour un ping mais paralyse tout le monde. La sonde **DNS** interroge un serveur précis et vérifie la **réponse** :

1. Ajoutez une sonde. Type : **DNS**. Nom d'affichage : `DNS — srv-dc1.sio.local`. Nom d'hôte : `srv-dc1.sio.local`. **Serveur DNS utilisé** : `192.168.20.250`, port `53`. **Type d'enregistrement DNS recherché** : `A`. **Valeur attendue** : `192.168.20.250` :

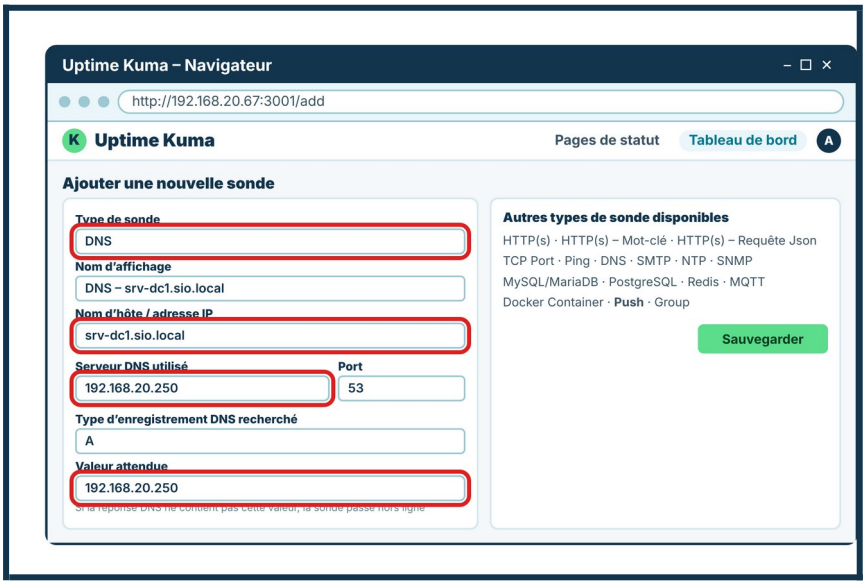



*Figure 13 – Formulaire d'une sonde DNS*

**À SAVOIR**

Si le serveur ne répond pas **ou** répond une autre adresse que la valeur attendue, la sonde passe « Hors ligne » : elle détecte aussi un enregistrement DNS modifié par erreur.

## C – Un certificat HTTPS qui expire

Un certificat expiré rend un site inaccessible. Une sonde HTTP(s) sur une adresse en **https** peut vous **prévenir avant l'échéance** :

1. Créez une sonde **HTTP(s)** sur l'adresse HTTPS d'un de vos services (par exemple `https://glpi.sio.local`) et activez **Notification d'expiration du certificat**. Le détail de la sonde affiche alors la date d'expiration (« Expiration Cert SSL. »). Les délais d'alerte (7, 14 et 21 jours par défaut) se règlent dans **Paramètres** › **Notifications**
2. Si le certificat est délivré par une **autorité interne** (SIO-CA), le conteneur ne la connaît pas et la sonde se met en erreur. Donnez-lui le certificat de l'autorité : copiez le fichier `sio-ca.pem` dans le dossier `data`, puis ajoutez la variable `NODE_EXTRA_CA_CERTS` au service :

TERMINAL
```
sudo cp sio-ca.pem /opt/uptime-kuma/data/sio-ca.pem
```

/opt/uptime-kuma/compose.yaml (à ajouter dans le service)
```
    environment:
      - NODE_EXTRA_CA_CERTS=/app/data/sio-ca.pem
```

TERMINAL
```
sudo docker compose up -d
```

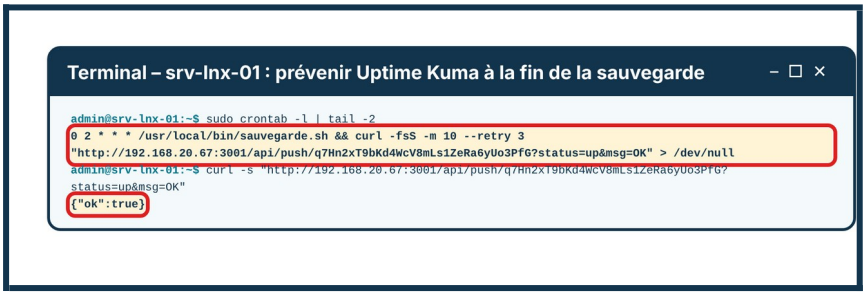
**REMARQUE IMPORTANTE**

L'option **Ignorer les erreurs liées au certificat SSL/TLS** fait taire l'erreur, mais **masque aussi les vrais problèmes** (certificat expiré, mauvais nom). Préférez toujours de fournir l'autorité de certification à Uptime Kuma.

## D – Une sonde Push : surveiller une tâche planifiée

Certaines tâches (sauvegarde nocturne, script) ne peuvent pas être interrogées : c'est **à elles de dire qu'elles ont réussi**. La sonde **Push** génère une adresse que la tâche appelle ; **si l'appel n'arrive pas dans le délai prévu, la sonde passe « Hors ligne »**.

1. Ajoutez une sonde. Type : **Push**. Nom d'affichage : `Sauvegarde nocturne`. Réglez l'**intervalle de vérification** sur la période attendue **plus une marge** : pour une tâche quotidienne, `90000` secondes (25 heures). **Sauvegardez**, puis copiez la **Push URL** affichée
2. Sur la machine concernée (ici **srv-lnx-01**), appelez cette adresse **à la fin du script, seulement s'il a réussi**. Le `&&` de la ligne cron garantit cela : on teste d'abord l'adresse à la main :




**À SAVOIR**

La réponse `{"ok":true}` confirme que Kuma a reçu le signal. Le paramètre `msg` est affiché dans l'historique. Si le script échoue, le `curl` n'est jamais exécuté : au bout de 25 heures, Kuma passe la sonde « Hors ligne » et alerte.

*Figure 14 – Appel de l'adresse Push depuis une tâche planifiée*

## E – Créer une notification

Pour l'instant, une panne ne se voit que si l'on regarde le tableau de bord. Créons un **canal d'alerte** : le **courriel**. Les réglages se trouvent dans **Paramètres › Notifications**, bouton **Créer une notification** :

1. **Type de notification** : `Courriel (SMTP)`. Donnez-lui un nom d'affichage
2. Renseignez le serveur : **nom d'hôte**, **port** (587 en général, avec « Sécurité » adaptée), **nom d'utilisateur** et **mot de passe** si le serveur l'exige, **Depuis l'adresse** (expéditeur) et **Vers l'adresse** (destinataire)
3. Activez **Activé par défaut** (les futures sondes utiliseront cette notification) et **Appliquer sur toutes les sondes existantes**, puis cliquez sur **Tester** : vous devez recevoir un courriel. **Sauvegardez**

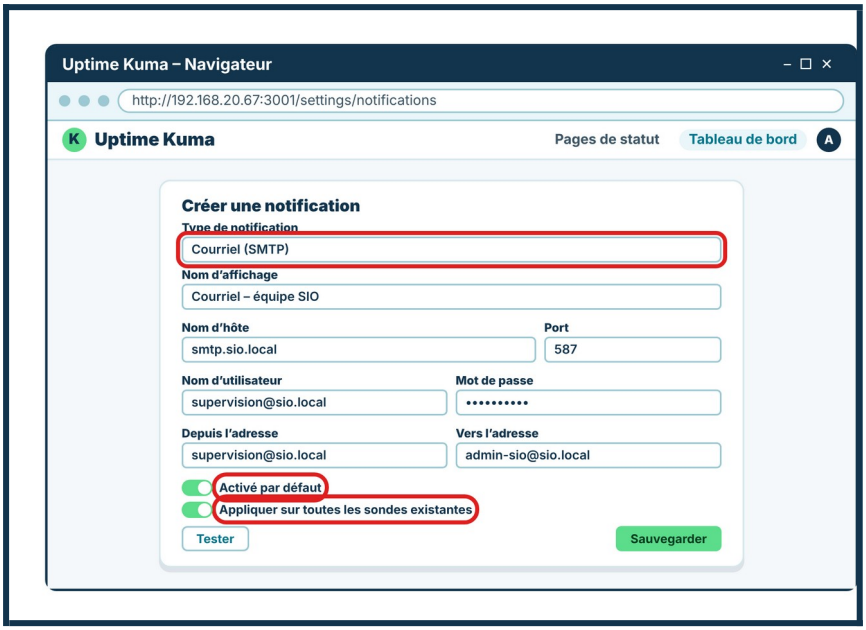



*Figure 15 – Création d'une notification par courriel*

**À SAVOIR**

**Tester** envoie un message d'essai : c'est la seule façon de savoir si le serveur de messagerie accepte vos réglages. Sans « Appliquer sur toutes les sondes existantes », les sondes déjà créées n'enverraient rien.

| Canal | Ce qu'il faut fournir |
|---|---|
| **Courriel (SMTP)** | Serveur, port, sécurité, identifiants, adresses expéditeur et destinataire |
| **Telegram** | Le **jeton** d'un robot créé avec @BotFather et l'**identifiant** de la conversation ou du groupe |
| **Discord** | L'**URL du webhook** d'un salon (Paramètres du salon › Intégrations › Webhooks) |
| **Webhook** | Une **URL** appelée en POST avec les détails de l'alerte, pour brancher un autre outil |

## F – Provoquer une panne pour tester

1. Arrêtez volontairement l'interface web de Zabbix : sur **SRV-ZBX**, `cd /opt/zabbix && sudo docker compose stop zabbix-web`
2. Observez le tableau de bord de Kuma : la sonde passe **En attente** (orange) puis **Hors ligne** (rouge), avec le message d'erreur, et un **courriel d'alerte** arrive :




*Figure 16 – Tableau de bord pendant la panne*

**À SAVOIR**

Les **événements importants** conservent les changements d'état avec l'heure et le message de la sonde (ici, connexion refusée : le port est fermé, la machine répond).

3. Redémarrez le service : `sudo docker compose start zabbix-web`. La sonde repasse **En ligne** et Kuma envoie un second courriel : **retour à la normale**

✔ POINT DE CONTRÔLE

- Les sondes **TCP**, **DNS** et **Push** sont créées et **En ligne**
- L'appel `curl` de la sonde Push renvoie `{"ok":true}`
- Le bouton **Tester** de la notification envoie bien un courriel
- Une panne provoquée entraîne **un courriel de panne**, puis **un courriel de retour à la normale**

# 06

ÉTAPE 6 SUR 6

# Page de statut, maintenance et exploitation

## A – Publier une page de statut

Une **page de statut** montre à vos utilisateurs l'état des services, **sans connexion** et sans exposer le reste de l'interface. Elle réduit le nombre d'appels du type « est-ce que ça marche ? ».

1. Ouvrez **Pages de statut**, puis **Ajouter une page de statut** (ou « Nouvelle page de statut »). Indiquez un **nom** (`État des services SIO`) et un **chemin** (`sio`), puis **Continuer** : la page sera publiée sous `/status/sio`
2. Dans l'éditeur, renseignez le titre et la description, cliquez sur **Ajouter un groupe** (`Infrastructure`), puis **Ajouter une sonde** pour y glisser vos sondes. Cliquez sur **Sauvegarder** :

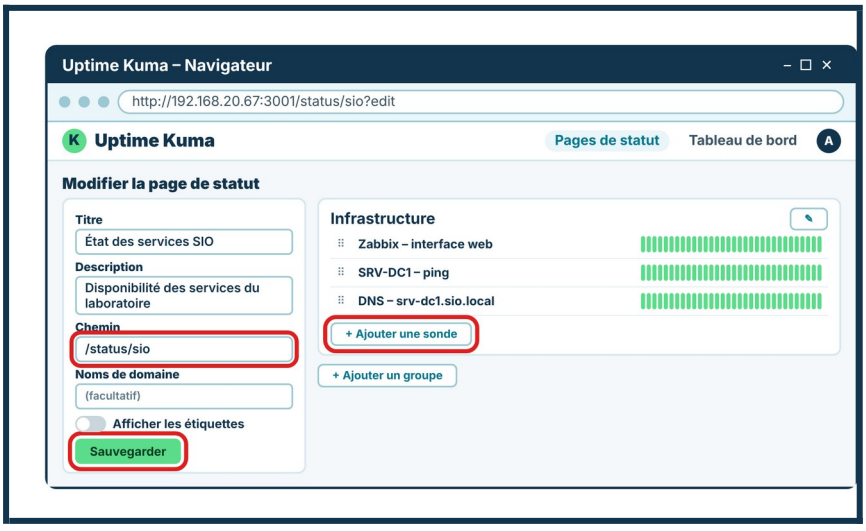



*Figure 17 – Modification de la page de statut*

À SAVOIR

On peut créer plusieurs pages (une par public ou par service), personnaliser le pied de page, ajouter un **nom de domaine** dédié et masquer ou afficher les étiquettes des sondes.

La page publique s'ouvre à l'adresse `http://192.168.20.67:3001/status/sio`. Le bandeau du haut résume : « Tous les systèmes sont opérationnels », ou signale les services en panne. Les **incidents** se publient avec **Créer un incident** (message épinglé en haut de la page) :

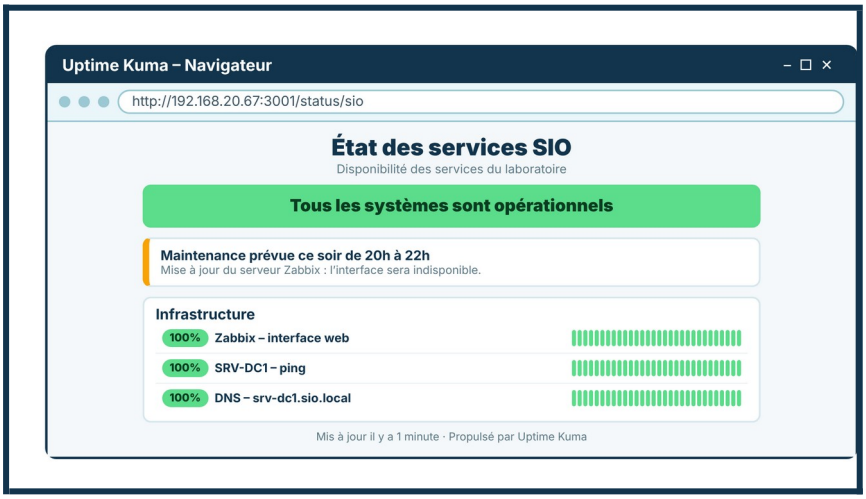



*Figure 18 – Page de statut publique*

**À SAVOIR**

Cette page est visible **sans compte** par quiconque a accès au serveur : ne publiez que ce qui peut être connu (évitez les adresses IP internes dans les noms de sondes d'une page vue de l'extérieur).

## B – Planifier une maintenance

Quand vous intervenez volontairement sur un service (mise à jour, redémarrage), les alertes sont inutiles et inquiètent. Une **maintenance** met les sondes concernées **en pause pour les alertes** pendant le créneau et prévient les utilisateurs sur la page de statut :

1. Ouvrez le menu du compte (en haut à droite) › **Maintenance**, puis **Planifier la maintenance** (« Créer une maintenance »)
2. Saisissez un **titre** et une **description**, choisissez les **sondes concernées** et les **pages d'état** où afficher le message, puis la **stratégie** : **Créneau de maintenance unique** (date de début et de fin, avec le **fuseau horaire**), récurrence (jours de la semaine ou du mois), ou activation manuelle. **Sauvegardez** :

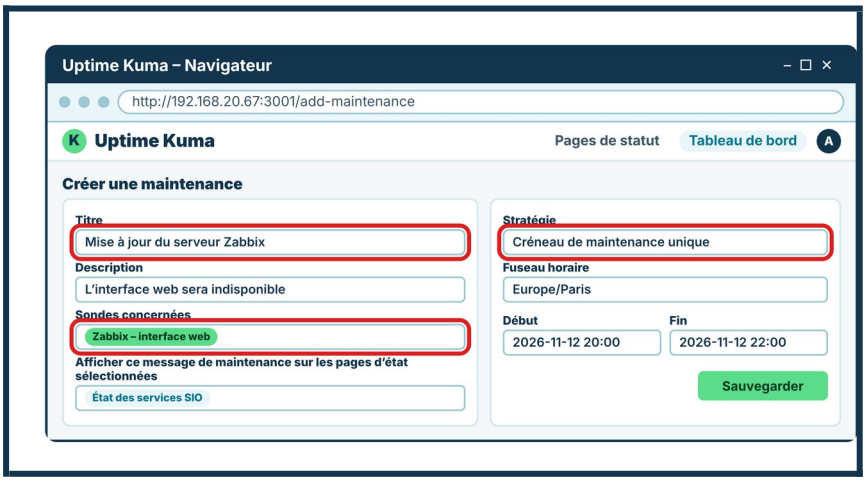



*Figure 19 – Création d'une maintenance*

**À SAVOIR**

Pendant le créneau, les sondes concernées affichent l'état « **Maintenance** » (bleu) : elles continuent d'être vérifiées mais **n'envoient aucune alerte** et ne dégradent pas la disponibilité affichée.

## C – Sauvegarder et mettre à jour

Tout ce qui compte (base de données, sondes, notifications, comptes) se trouve dans le dossier `/opt/uptime-kuma/data`. L'export JSON des sondes n'existe plus en version 2 : la sauvegarde recommandée est celle du **dossier de données**. Arrêtez le conteneur le temps de la copie pour obtenir une base cohérente (quelques secondes), puis redémarrez-le :

TERMINAL

```
cd /opt/uptime-kuma
sudo docker compose stop
sudo tar -czf /var/backups/uptime-kuma-$(date +%F).tgz data
sudo docker compose start
```

La mise à jour consiste à **télécharger la nouvelle image** puis à **recréer le conteneur** : les données, elles, restent dans `data`. **Sauvegardez avant chaque mise à jour** et lisez les notes de version pour les changements de version majeure :

TERMINAL

```
cd /opt/uptime-kuma
sudo docker compose pull
sudo docker compose up -d --force-recreate
sudo docker image prune -f
```




*Figure 20 – Sauvegarde du dossier data et mise à jour*

**À SAVOIR**

`docker image prune -f` supprime les anciennes images devenues inutiles. Copiez l'archive **hors du serveur** (autre machine, partage) : une sauvegarde sur le même disque ne protège pas d'une panne du disque.

**REMARQUE IMPORTANTE**

Avec la balise `2`, Compose suit les versions **2.x** : vous recevez les correctifs mais pas le passage à une future version majeure. Pour **figer** une version précise (par exemple en production), indiquez-la dans le fichier (`louislam/uptime-kuma:2.5.7`) et changez-la volontairement à chaque mise à jour.

## D – Publier en HTTPS derrière Traefik

En l'état, l'interface est en **HTTP** : le mot de passe administrateur circule en clair. Pour un usage réel, on place Uptime Kuma derrière un **proxy inverse** qui assure le **HTTPS**. Si vous disposez du proxy **Traefik** du tutoriel « Installer Zabbix derrière Traefik » (certificat délivré par SIO-CA), ajoutez un **fichier dynamique** pour Kuma, par exemple `/etc/traefik/dynamic/kuma.yml` (adaptez le nom du point d'entrée à votre configuration) :

**/etc/traefik/dynamic/kuma.yml**

```
http:
  routers:
    kuma:
      rule: "Host(`kuma.sio.local`)"
      entryPoints:
        - websecure
      tls: {}
      service: kuma
  services:
    kuma:
      loadBalancer:
        servers:
          - url: "http://192.168.20.67:3001"
```

1. Enregistrez le nom `kuma.sio.local` dans le DNS du domaine, vers l'adresse du serveur Traefik, et délivrez-lui un certificat comme pour les autres services
2. Dans Uptime Kuma, ouvrez **Paramètres › Proxy inverse** et réglez **Proxy de confiance** sur **Oui** : Kuma pourra ainsi lire la vraie adresse des clients dans les en-têtes transmis par le proxy
3. Une fois le HTTPS validé, **limitez l'accès direct au port 3001** (pare-feu en amont, ou publication sur `127.0.0.1` si le proxy est sur le même serveur)

**À SAVOIR**

Uptime Kuma utilise les **WebSockets** pour mettre à jour l'interface en direct : un proxy qui les bloque donne une page qui charge mais reste « en connexion ». Traefik les prend en charge sans réglage. Avec un autre proxy (nginx, Apache), il faut transmettre les en-têtes `Upgrade` et `Connection`, et réserver à Kuma un nom de domaine dédié.

## E – Dépanner

| Symptôme | Cause probable | Solution |
|---|---|---|
| **Interface inaccessible sur `:3001`** | Conteneur arrêté ou port filtré | `docker compose ps` et `docker compose logs uptime-kuma` ; vérifier le pare-feu et la ligne `ports` |
| **Le conteneur redémarre en boucle** | Droits ou volume de données invalides | Lire le journal ; vérifier que `./data` est un dossier local (pas un NFS) et que la syntaxe du fichier est correcte |
| **Sonde HTTPS en erreur de certificat** | Autorité de certification interne inconnue du conteneur | Fournir l'autorité avec `NODE_EXTRA_CA_CERTS` (étape 5), plutôt que d'ignorer l'erreur |
| **Pas de courriel reçu** | Réglages SMTP incorrects ou serveur qui refuse l'expéditeur | Utiliser **Tester** ; contrôler port, sécurité, identifiants ; lire le journal du conteneur |
| **Fausses alertes de courte durée** | Réseau instable ou cible lente | Augmenter **Tentatives** et le **délai d'expiration** de la sonde |
| **Une sonde Push passe « Hors ligne » alors que la tâche tourne** | Intervalle trop court ou horloge décalée | Régler l'intervalle sur la période de la tâche **plus une marge** ; vérifier l'heure des deux machines |
| **Derrière un proxy, l'interface reste en connexion** | WebSockets bloqués | Transmettre `Upgrade/Connection` ; **Proxy de confiance** = Oui |
| **Mot de passe administrateur perdu** | — | Réinitialiser en ligne de commande dans le conteneur : `docker compose exec uptime-kuma npm run reset-password` |

## Pour aller plus loin

Uptime Kuma sait aussi surveiller un **conteneur Docker** (accès au socket Docker, à réserver à un réseau de confiance : ce socket donne des droits équivalents à `root`), un **mot-clé** dans une page, une **requête Json** ou une **base de données**, et regrouper des sondes dans un **groupe**. Pour des mesures détaillées sur les serveurs eux-mêmes (processeur, mémoire, disque), complétez-le avec **Zabbix** ; pour comprendre ce que fait un poste sur le réseau, voir le tutoriel « **Analyser le trafic réseau avec Sniffnet** ».

✔ POINT DE CONTRÔLE

- Une **page de statut** est publiée et visible sans connexion
- Une **maintenance** planifiée met les sondes en état « Maintenance » sans alerte
- Vous disposez d'une **archive du dossier data** conservée hors du serveur
- Vous savez **mettre à jour** l'image et expliquer pourquoi les données sont conservées
- Vous savez comment publier Kuma en **HTTPS** et réinitialiser le mot de passe administrateur