

Des scripts plein la tête !

- [Surveiller vos sites HTTPS automatiquement avec Python et recevez des alertes sur votre téléphone](#)

Surveiller vos sites HTTPS automatiquement avec Python et recevez des alertes sur votre téléphone

Vous gérez des sites web pour vos clients et vous voulez être alerté immédiatement en cas de problème ? Dans cet article, je vous montre comment créer un système de surveillance automatique qui vérifie l'accessibilité de vos sites et la validité de leurs certificats SSL, avec des notifications directement sur votre téléphone.

Pourquoi surveiller vos sites HTTPS ?

Quand vous gérez plusieurs sites WordPress pour des artisans et PME locales, vous devez vous assurer que :

- Les sites sont accessibles 24h/24
- Les certificats SSL sont valides (sinon les navigateurs affichent des alertes effrayantes)
- Vous êtes prévenu rapidement en cas de problème

Un certificat SSL expiré, c'est l'assurance d'un client mécontent qui vous appelle en panique parce que son site affiche "Connexion non sécurisée".

Les bases : HTTP vs HTTPS

Avant de plonger dans le code, un rappel rapide. **HTTP**, c'est comme envoyer une carte postale : tout le monde peut lire ce qui transite. **HTTPS**, c'est une lettre scellée dans une enveloppe avec un sceau officiel (le certificat SSL).

Le certificat SSL garantit deux choses :

- Le site est bien celui qu'il prétend être
- Les données échangées sont chiffrées

Ce dont vous avez besoin

Côté technique :

- Python 3.7 minimum
- Deux bibliothèques : `requests` et `python-dotenv`
- Un compte Ntfy (gratuit) pour les notifications

Installation rapide :

```
pip install requests python-dotenv
```

Configuration : le fichier .env

Créez un fichier `.env` avec vos paramètres :

```
HOSTS=https://site-client1.fr|Client 1,https://site-client2.fr|Client 2
TIMEOUT=10
NTFY_TOPIC=monitoring-votre-topic-unique
```

Pour Ntfy, allez sur ntfy.sh, choisissez un nom de topic unique (par exemple `monitoring-Ab4yteBE35Fwx`), et notez-le dans votre `.env`.

Le cœur du script : vérifier HTTPS et SSL

Voici comment le script vérifie un site :

```
def check_https_and_ssl(url, timeout=10):
    result = {
```

```

        'accessible': False,
        'ssl_valid': False,
        'ssl_expires': None,
        'ssl_days_remaining': None,
        'http_code': None,
        'error': None
    }

# Étape 1 : Tester l'accessibilité
try:
    response = requests.get(url, timeout=timeout, verify=True)
    result['http_code'] = response.status_code
    result['accessible'] = (200 <= response.status_code < 400)
except requests.exceptions.SSLError:
    result['error'] = "Certificat SSL invalide ou expiré"
    return result

# Étape 2 : Analyser le certificat SSL
parsed = urlparse(url)
hostname = parsed.hostname

context = ssl.create_default_context()
with socket.create_connection((hostname, 443), timeout=timeout) as sock:
    with context.wrap_socket(sock, server_hostname=hostname) as secure_sock:
        cert = secure_sock.getpeercert()
        expires = datetime.strptime(cert['notAfter'], '%b %d %H:%M:%S %Y %Z')
        days_remaining = (expires - datetime.now()).days

        result['ssl_valid'] = True
        result['ssl_expires'] = expires.strftime('%Y-%m-%d')
        result['ssl_days_remaining'] = days_remaining

    return result

```

Le script fait deux choses : il se connecte au site via HTTPS pour vérifier l'accessibilité, puis il récupère et analyse le certificat SSL pour vérifier sa validité et sa date d'expiration.

Recevoir les alertes sur votre téléphone

Pour les notifications, on utilise Ntfy. C'est simple et efficace :

```
def send_ntfy(topic, title, message, priority='default'):
    url = f"https://ntfy.sh/{topic}"
    headers = {
        'Title': title,
        'Priority': priority,
        'Tags': 'warning,rotating_light'
    }
    requests.post(url, data=message.encode('utf-8'), headers=headers)
```

Installez l'app Ntfy sur votre téléphone, abonnez-vous à votre topic, et vous recevrez instantanément les alertes quand un site tombe ou se rétablit.

Automatiser la surveillance

Sur Linux ou macOS, utilisez Cron pour lancer le script toutes les 5 minutes :

```
crontab -e
```

Ajoutez cette ligne :

```
*/5 * * * * cd /chemin/vers/script && python3 check_https.py >> logs/cron.log 2>&1
```

Sur Windows, utilisez le Planificateur de tâches avec un fichier `.bat` qui lance le script.

Ce que vous obtenez

Quand le script tourne, vous recevez sur votre téléphone :

- 🚨 Une alerte urgente si un site devient inaccessible ou si son certificat SSL est invalide
- 🟢 Une notification de récupération quand le site redevient accessible
- ⚠️ Un avertissement si un certificat expire dans moins de 30 jours

Le script garde également un historique des incidents dans des fichiers JSON, pratique pour suivre la fiabilité de votre hébergeur.

Mon cas d'usage

Je gère plusieurs sites WordPress pour des clients locaux, hébergés chez différents prestataires. Ce script me permet de dormir tranquille : je suis alerté immédiatement si un site tombe, et je peux prévenir le client avant même qu'il ne s'en aperçoive. Ça fait toute la différence en termes de service client.

Aller plus loin

Vous pouvez facilement adapter le script pour :

- Vérifier des API REST plutôt que des sites web
- Surveiller des serveurs internes via VPN
- Monitorer des temps de réponse et alerter en cas de lenteur
- Intégrer d'autres canaux de notification (email, Slack, Discord)