

Bonnes pratiques de monitoring

Voici la **synthèse claire et structurée** de l'usage de **Grafana** et **Prometheus** dans le contexte de **surveillance réseau et sécurisation pendant l'usage de visio.workeezconnect.fr**.

Suivie des **bonnes pratiques**, puis des **propositions d'évolutions automatiques**.

Synthèse : Grafana & Prometheus pour visio.workeezconnect.fr

Objectif : Assurer la **surveillance active**, la **visualisation des performances**, et la **détection d'anomalies** durant l'usage en production de la plateforme de visio `visio.workeezconnect.fr`.

Architecture fonctionnelle

Composant	Rôle principal
Prometheus	Collecte les métriques (temps réel, périodique) via scraping HTTP
Blackbox Exporter	Teste la disponibilité (HTTP, TCP, DNS, ICMP) de <code>visio.workeezconnect.fr</code>
Node Exporter	Collecte CPU, RAM, disque, uptime sur le serveur Jitsi
Telegraf	Supervision système élargie, relais Prometheus + logs
Grafana	Visualisation en tableaux de bord et génération d'alertes visuelles
Wazuh (optionnel)	Corrélation comportementale (SIEM), alerte avancée

Bonnes pratiques pour surveillance visio + sécurité

1. Supervision technique du serveur

- CPU/RAM/Disque/Uptime via `node_exporter`
- Temps de réponse DNS/HTTP via `blackbox_exporter`
- Disponibilité du port 443 ou spécifique (`tcp_connect`)
- Surveillance du port vidéo (UDP ou WebRTC relay)

2. Alerting Prometheus






- Alerte HTTP (status != 2xx ou latence > 3s)
- Alerte DNS (échec de résolution ou `lookup_time` > 1s)
- Alerte TCP (connexion impossible)
- Bannière SSH absente (spoof ou altération)

3. Dashboards Grafana

- Panels clairs par **job** : HTTP / DNS / Node
- Coloriage dynamique : vert = OK, orange = lent, rouge = KO
- Statistiques uptime, incidents sur 24h/7j
- Corrélation entre modules : `probe_success`, `scrape_duration_seconds`, etc.

Évolutions automatiques possibles

À inclure prochainement

Fonction	Détail
 Génération auto du <code>alert.rules.yml</code>	Basé sur les modules actifs dans Prometheus
 Export CSV/JSON des métriques	Par jour / semaine, vers <code>~/logs/</code>
 Couleur terminal (tput)	Vert = OK, Rouge = FAIL dans les scripts
 Intégration n8n / Slack / Wazuh	Notification instantanée par webhook / SIEM
 Script de charge / saturation	Simule 10/50/100 connexions + failover

Automatisation : Voici ce qui peut être générer :

Option	Usage	Commande d'appel	Intérêt
Makefile	Chaînage manuel logique	<code>make monitor</code>	Structuration, test multi-étape
.service	Exécution au boot	<code>systemctl enable monitor-check</code>	Démarrage avec la machine
cron	Exécution périodique	<code>crontab -e</code> (ex : toutes les 15 min)	Planification récurrente

On automatise un check capacitaire ou un audit régulier, notamment avant chaque test de charge.

A savoir pour Makefile : Vérification des permissions pour Makefile et scripts liés

1. Le Makefile lui-même

Aucune permission d'exécution (`chmod +x`) n'est requise sur un Makefile car il est interprété par la commande `make`.

- ♦ Il suffit qu'il soit lisible :

`-rw-r--r-- 1 root root ... Makefile`  OK

2. Les scripts appelés par le Makefile (dans scripts/)

Les scripts .sh doivent en revanche être exécutables :

✓ Pour tous les scripts :

`chmod +x /home/ubuntu/monitoring/scripts/*.sh`

3. Résumé des permissions recommandées

Élément	Commande recommandée	Justification
<code>/home/ubuntu/monitoring/Makefile</code>	<code>chmod 644 (déjà OK)</code>	Fichier lu par make, pas exécuté
<code>/home/ubuntu/monitoring/scripts/*.sh</code>	<code>chmod +x</code>	Exécution des scripts appelés

4. Bonus : test rapide du Makefile

`make -C /home/ubuntu/monitoring/`