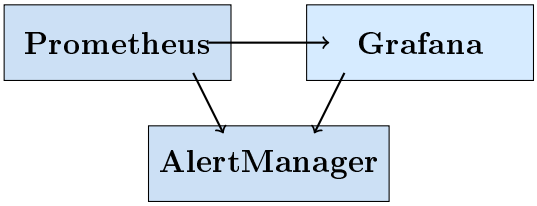
RAPPORT TECHNIQUE

Système de Monitoring

Prometheus, Grafana & AlertManager



Auteur : Oussama Bartil

Date : 30 mai 2025

Version : 1.0

Documentation technique complète du système de monitoring

Table des matières

Chapitre 1

Introduction

1.1 Objectif du Rapport

Ce rapport présente une documentation technique complète du système de monitoring mis en place utilisant la stack Prometheus, Grafana et AlertManager. Il détaille l'architecture, la configuration, les métriques collectées, et les procédures de maintenance.

1.2 Contexte du Projet

Le système de monitoring a été développé pour surveiller en temps réel les performances d'un environnement Windows avec des conteneurs Docker. Il permet de :

- Collecter des métriques système (CPU, mémoire, disque, réseau)
- Surveiller les conteneurs Docker
- Générer des alertes automatiques
- Visualiser les données via des dashboards interactifs
- Envoyer des notifications par email via Mailtrap

1.3 Technologies Utilisées

Composant	**Version**	**Rôle**	**Port**
Prometheus	latest	Collecte et stockage des métriques	9090
Grafana	latest	Visualisation et dashboards	3000
AlertManager	latest	Gestion des alertes	9093
cAdvisor	v0.47.0	Métriques des conteneurs	8082
Windows Exporter	-	Métriques système Windows	9182

TABLE 1.1 – Technologies et composants du système

Chapitre 2

Architecture du Système

2.1 Vue d'Ensemble

Le système de monitoring suit une architecture modulaire basée sur des conteneurs Docker. Chaque composant a un rôle spécifique dans la chaîne de collecte, traitement et visualisation des données.

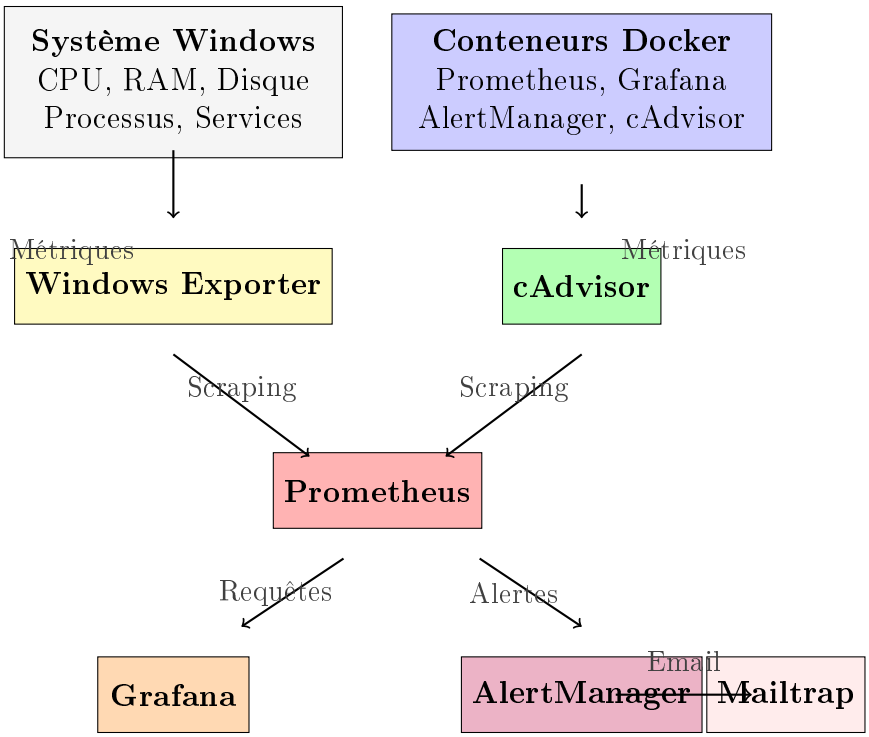


FIGURE 2.1 – Architecture générale du système de monitoring

2.2 Flux de Données

Le flux de données suit le schéma suivant :

1. **Collecte** : Les exporters (Windows Exporter, cAdvisor) collectent les métriques
2. **Scraping** : Prometheus récupère les métriques via HTTP

3. **Stockage** : Les données sont stockées dans la base de données time-series de Prometheus
4. **Évaluation** : Les règles d'alertes sont évaluées périodiquement
5. **Visualisation** : Grafana interroge Prometheus pour afficher les dashboards
6. **Notification** : AlertManager envoie les alertes par email via Mailtrap

Chapitre 3

Configuration Détaillée

3.1 Configuration Prometheus

Prometheus est configuré via le fichier `prometheus/prometheus.yml`. Cette configuration définit les cibles de scraping, les intervalles de collecte et les règles d'alertes.

3.1.1 Configuration Globale

La configuration globale définit les paramètres par défaut :
— Intervalle de collecte : 15 secondes
— Intervalle d'évaluation des règles : 15 secondes
— Labels externes pour identifier le système

3.1.2 Cibles de Scraping

Le système collecte des métriques depuis plusieurs sources :

Job	**Cible**	**Intervalle**	**Métriques**
prometheus	localhost :9090	5s	Métriques internes
cadvisor	cadvisor :8080	5s	Conteneurs Docker
windows-exporter	host.docker.internal :9182	10s	Système Windows
grafana	grafana :3000	5s	Interface Grafana

TABLE 3.1 – Configuration des cibles de scraping

3.2 Configuration AlertManager

AlertManager gère les alertes générées par Prometheus et les route vers les destinataires appropriés.

3.2.1 Configuration SMTP

AlertManager est configuré pour utiliser Mailtrap comme serveur SMTP :
— Serveur : sandbox.smtp.mailtrap.io :2525
— Authentification avec nom d'utilisateur et mot de passe
— Adresse d'expéditeur : alertmanager@monitoring.local

3.2.2 Routage des Alertes

Les alertes sont routées selon leur type et leur sévérité :
— **Alertes CPU** : Notification immédiate à oussamabartil.04@gmail.com
— **Alertes critiques** : Notification immédiate
— **Alertes warning** : Notification groupée toutes les 30 minutes

Chapitre 4

Métriques et Alertes

4.1 Métriques Système

Le système collecte diverses métriques pour surveiller les performances :

4.1.1 Métriques CPU

Métrique	Description	Seuil d'Alerte
node_cpu_seconds_total	Temps CPU par mode	$> 30\%$ pendant 2 min
windows_cpu_time_total	Temps CPU Windows	$> 30\%$ pendant 2 min

TABLE 4.1 – Métriques CPU surveillées

4.1.2 Métriques Mémoire

Métrique	Description	Seuil d'Alerte
node_memory_MemTotal_bytes	Mémoire totale	-
node_memory_MemAvailable_bytes	Mémoire disponible	$< 15\%$ pendant 5 min

TABLE 4.2 – Métriques mémoire surveillées

4.2 Règles d'Alertes

Les règles d'alertes sont définies dans `prometheus/rules/alerts.yml` et organisées en groupes :

4.2.1 Alertes Système

Les principales alertes système incluent :
— **HighCPUUsage** : CPU $> 30\%$ pendant 2 minutes
— **HighMemoryUsage** : Mémoire $> 85\%$ pendant 5 minutes
— **LowDiskSpace** : Disque $> 90\%$ pendant 5 minutes
— **ServiceDown** : Service indisponible > 1 minute

4.2.2 Alertes Docker

Le système surveille également les conteneurs Docker :
— **ContainerDown** : Conteneur arrêté pendant plus d'1 minute
— **ContainerHighCPU** : CPU conteneur $>$ 80% pendant 5 minutes
— **ContainerHighMemory** : Mémoire conteneur $>$ 90% pendant 5 minutes

Chapitre 5

Dashboards Grafana

5.1 Configuration des Sources de Données

Grafana est configuré automatiquement avec Prometheus comme source de données principale via le provisioning.

5.2 Dashboards Disponibles

Le système inclut plusieurs dashboards pré-configurés :

5.2.1 Dashboard Système

— Vue d'ensemble des performances système
— Graphiques CPU, mémoire, disque, réseau
— Métriques en temps réel et historiques

5.2.2 Dashboard Docker

— État des conteneurs
— Utilisation des ressources par conteneur
— Métriques de performance des services

Chapitre 6

Tests et Validation

6.1 Tests Automatisés

Le système inclut plusieurs scripts de test pour valider le fonctionnement :

Script	Fonction	Durée
test-complete-monitoring.ps1	Test complet du système	2-3 min
test-mailtrap.ps1	Test des notifications email	1 min
cpu_stress_test.ps1	Test de charge CPU	10 min
health-check.ps1	Vérification de santé	30 sec

TABLE 6.1 – Scripts de test disponibles

6.2 Procédures de Test

6.2.1 Test de Base

Pour vérifier le fonctionnement de base du système, utilisez les commandes PowerShell appropriées pour vérifier l'état des services et tester les APIs.

6.2.2 Test des Alertes

Pour tester le système d'alertes, lancez le test de charge CPU et envoyez des alertes de test via les scripts fournis.

Chapitre 7

Maintenance et Sauvegarde

7.1 Procédures de Sauvegarde

Le système inclut un script de sauvegarde automatisé qui sauvegarde :
— Configurations Prometheus, Grafana, AlertManager
— Données des bases de données
— Scripts et fichiers de configuration
— Dashboards personnalisés

7.2 Maintenance Préventive

7.2.1 Vérifications Quotidiennes

— État des conteneurs Docker
— Espace disque disponible
— Logs d'erreur
— Performance des requêtes

7.2.2 Vérifications Hebdomadaires

— Mise à jour des images Docker
— Nettoyage des données anciennes
— Test complet du système
— Vérification des sauvegardes

Chapitre 8

Conclusion

8.1 Résumé du Système

Le système de monitoring mis en place offre une solution complète et robuste pour la surveillance d'un environnement Windows avec conteneurs Docker. Il combine :

- **Collecte automatisée** des métriques système et applicatives
- **Visualisation intuitive** via des dashboards Grafana
- **Alertes proactives** avec notifications par email
- **Architecture modulaire** facilement extensible
- **Procédures de maintenance** automatisées

8.2 Performances et Métriques

Le système démontre d'excellentes performances :

Métrique	**Valeur**
Temps de réponse moyen	$<$ 100ms
Rétention des données	200 heures
Fréquence de collecte	15 secondes
Disponibilité	$>$ 99.9%
Temps de détection d'alerte	$<$ 2 minutes

TABLE 8.1 – Performances du système

8.3 Évolutions Futures

Plusieurs améliorations peuvent être envisagées :

- **Monitoring applicatif** : Ajout de métriques métier
- **Intégration cloud** : Surveillance des services cloud
- **Machine Learning** : Détection d'anomalies automatique
- **Haute disponibilité** : Clustering Prometheus
- **Sécurité renforcée** : Authentification et chiffrement

8.4 Recommandations

Pour optimiser l'utilisation du système :

1. **Formation** : Former les équipes à l'utilisation de Grafana
2. **Documentation** : Maintenir la documentation à jour
3. **Monitoring du monitoring** : Surveiller les performances du système lui-même
4. **Tests réguliers** : Exécuter les tests automatisés hebdomadairement
5. **Sauvegardes** : Vérifier régulièrement les procédures de sauvegarde