Contents

Journal de bord du projet

Supervision d'une application web avec Prometheus & Grafana

Jour 1 : Lancement et cadrage

Objectif défini : Mettre en place une stack de monitoring professionnelle pour une application web Flask, incluant collecte de métriques, visualisation et alerting.

Choix techniques :

- Python (Flask) pour l'application
- Prometheus pour la collecte et le stockage des métriques
- Grafana pour la visualisation
- Docker Compose pour l'orchestration
- cAdvisor & Node Exporter pour les métriques système et conteneurs

Initialisation du dépôt local et création du repo GitHub.

Jour 2 : Développement de l'application et instrumentation

Développement de l'app Flask avec endpoints de test (/, /api/health, /api/users, /api/products, /api/orders, /metrics, /error, /slow).

Ajout des métriques Prometheus dans le code (compteurs, histogrammes, gauges).

Création du Dockerfile pour containeriser l'application.

Test local de l'application (hors Docker) pour valider les endpoints et la génération de métriques.

Jour 3 : Orchestration Docker & configuration Prometheus

Rédaction du docker-compose.yml pour lancer tous les services (Flask, Prometheus, Grafana, cAdvisor, Node Exporter, Alertmanager).

Écriture du fichier prometheus.yml pour configurer les jobs de scraping (app Flask, cAdvisor, Node Exporter).

Ajout des règles d'alerting dans prometheus-rules.yml (erreurs, lenteurs, ressources).

Premier test de la stack complète avec docker-compose up.

Jour 4 : Provisioning automatique de Grafana

Création des fichiers de provisioning :

- datasources/datasource.yml pour connecter Prometheus à Grafana
- dashboards/web-app-dashboard.json pour le dashboard principal
- dashboards/dashboard.yml pour l'auto-import

Test du provisioning automatique : vérification de la présence du dashboard dans Grafana.

Correction d'un bug : dashboard non visible → redémarrage de Grafana, vérification des chemins et permissions.

Jour 5 : Sécurité, scripts et tests de charge

Ajout du fichier .env pour la gestion des secrets (clé PROMETHEUS_HEX).

Écriture des scripts de démarrage :

- start.sh (Linux/Mac)
- start.bat (Windows)

Développement du script load-test.py pour générer du trafic et tester les alertes.

Tests de charge : validation du déclenchement des alertes dans Grafana et Alertmanager.

Jour 6 : Documentation et rapport

Rédaction du README.md détaillé (installation, utilisation, dépannage).

Création du rapport de projet (PDF dans le dossier Document).

Ajout d'un guide d'utilisation et d'un journal de bord pour la soutenance.

Jour 7 : Gestion du dépôt Git et branches

Initialisation du dépôt distant GitHub.

Push du projet complet sur la branche main (hors dossier Document).

Création de la branche docs pour ne pousser que le dossier Document (rapport PDF).

Gestion d'un incident : suppression accidentelle des fichiers → restauration via git checkout main && git reset --hard origin/main.

Difficultés rencontrées & solutions

Provisioning Grafana non fonctionnel : correction des chemins, redémarrage du service.

Problème d'encodage du .env sous Windows : utilisation de PowerShell pour forcer l'ASCII.

Suppression accidentelle de fichiers : récupération grâce à Git.

Alertes Prometheus non déclenchées : ajustement des règles et des seuils.

Livrables finaux

Stack Docker opérationnelle (Flask, Prometheus, Grafana, cAdvisor, Node Exporter, Alertmanager), Dashboard Grafana auto-provisionné, Scripts de démarrage et de test de charge, Rapport PDF et documentation complète, Journal de bord détaillé.

Bilan et perspectives

Projet abouti : stack de monitoring professionnelle, automatisée, prête pour la production ou la démonstration.

Perspectives : déploiement cloud, CI/CD, extension multi-applications, intégration d'autres exporters.