Exporteur de métriques d'une plateforme BI

Service de supervision mis en production : il interroge périodiquement l'API d'administration d'une plateforme de restitution et expose des métriques consommables par un collecteur, avec alerting jusqu'à la console d'astreinte.

Résultat mesuré : surveillance d'environ 470 modèles, taux de succès de rafraîchissement rendu mesurable donc pilotable, délai de détection des pannes ramené de plusieurs heures à quelques minutes.

Fiches de méthode associées : `../../docs/14-observabilite-metriques.md`, `../../docs/13-docker-deploiement.md`, `../../docs/07-integration-api-rest.md`

Le problème résolu

La plateforme de restitution n'avait aucune alerte native sur échec de rafraîchissement. Les pannes étaient découvertes par un utilisateur métier ouvrant un rapport aux données périmées, souvent plusieurs jours après.

La décision d'architecture qui structure tout

Exposer des métriques plutôt qu'écrire les statuts dans une table de l'entrepôt.

Écrire dans l'entrepôt	Exposer des métriques
couple la supervision à ce qu'elle surveille	plan d'observabilité indépendant
si l'entrepôt tombe, la supervision tombe	la supervision survit à la panne qu'elle doit signaler
tableaux de bord à construire	outillage de visualisation et d'alerte déjà en place
historisation à gérer	rétention gérée par la plateforme de métriques

L'argument qui tranche, et qui convainc immédiatement une équipe d'exploitation : une supervision qui dépend du système surveillé ne signale pas la panne de ce système.

Architecture



Ce qui vaut d'être regardé

1. Le client HTTP robuste (`clients/base.py`)

Trois protections combinées, réutilisables pour toute API externe à quota :

Protection	Mise en œuvre

Limiteur de débit	seau à jetons, avec une horloge insensible aux ajustements système
Disjoncteur	fermé, puis ouvert après N échecs, puis semi-ouvert pour un essai
Reprise	uniquement sur erreurs transitoires, jamais sur 4xx métier

Points qui font la différence :

- le 401 déclenche un seul renouvellement de jeton, pas une boucle ;
- le 429 respecte l'en-tête de temporisation fourni par le serveur plutôt qu'un délai arbitraire ;
- les identifiants variables des chemins d'API sont normalisés avant de devenir des étiquettes, sinon chaque appel crée une série temporelle distincte.

2. La gestion du jeton (`secrets/token_manager.py`)

Jeton gardé en mémoire uniquement, avec renouvellement proactif avant expiration. Jamais persisté : un jeton écrit dans un fichier survit au processus.

Le champ du jeton est exclu de la représentation textuelle de l'objet : sans cela, une trace d'exception affichant l'objet fuite le jeton.

3. Le flot du coffre-fort (`secrets/vault_client.py`, `deployment/vault-secret-id.j2`)

```
1. identité de rôle (statique)      variable d'environnement du service
2. secret d'authentification        déposé sur disque par le pipeline
3. lecture au démarrage
4. échange contre un jeton d'accès
5. suppression immédiate du fichier de secret
6. lecture des secrets métier
7. relecture périodique             ← permet la rotation sans redéploiement
```

Deux propriétés obtenues : aucun secret ne persiste sur disque au-delà du démarrage, et la rotation d'un secret ne demande pas de redéploiement.

4. La discipline de cardinalité (`metrics/definitions.py`)

La contrainte qui structure toute conception de métriques. Chaque combinaison unique de valeurs d'étiquettes crée une série temporelle.

Type	Étiquettes autorisées
Jauge, compteur	identifiants stables et de faible cardinalité uniquement
Information	métadonnées variables (nom lisible, espace, propriétaire), jointes par l'identifiant

Ce découplage permet de renommer un objet sans créer une nouvelle série, et évite qu'une étiquette à forte variabilité rende la plateforme de métriques inexploitable.

5. Les trois sondes (`server/http.py`, `healthcheck.py`)

Sonde	Vérifie	Ne doit pas vérifier
vivacité	le processus répond	ses dépendances
disponibilité	dépendances joignables, premier cycle effectué	rien de coûteux
métriques	expose le registre	rien

La sonde de vivacité ne dépend délibérément d'aucune dépendance externe. Sinon une indisponibilité momentanée de l'API surveillée provoque un redémarrage en boucle du collecteur, au pire moment.

6. La stratégie d'optimisation de quota

(`collectors/refresh_status.py`) Endpoint agrégé préféré (un appel paginé pour tout le périmètre), endpoint détaillé en repli activable par option (un appel par élément, coûteux).

Principe généralisable : chercher l'endpoint « vue agrégée » avant l'endpoint « détail par élément », qui reste un filet de sécurité, pas le chemin par défaut.

7. La collecte découplée du scrutin (`scheduler/adaptive.py`)

Un planificateur interne déclenche la collecte à intervalle fixe, indépendamment des requêtes entrantes. Deux raisons : le collecteur externe interroge souvent, ce qui dépasserait le quota de l'API ; et une collecte lente ferait dépasser le délai du collecteur, interprété comme une panne.

L'intervalle est dicté par le quota de l'API source, pas par la fréquence de scrutin. Avec des fenêtres de fréquence élevée configurables pour les heures ouvrées.

8. Le déploiement (`distribution/`, `deployment/`, `.gitlab-ci.yml`)

Construction multi-étapes durcie, avec proxy paramétrable au moment de la construction (jamais codé en dur, sinon l'image est inutilisable ailleurs). Pipeline branché sur les gabarits partagés de l'organisation. Déploiement en validation manuelle, avec attente active du contrôle de santé puis test de fumée, et enregistrement auprès de l'annuaire de services.

C'est la partie qui a demandé le plus d'apprentissage réel : le proxy d'entreprise et les gabarits partiellement documentés sont les vrais obstacles d'une première mise en production, pas le code.

Vérification après anonymisation

Contrôle	Résultat
Tests	65 passent
Gabarits de déploiement	expressions de templating uniquement
Motifs sensibles résiduels	aucun

Ce qui a été retiré

Retiré	Raison
`build/`, `htmlcov/`, `.coverage`, `*.egg-info`	artefacts de construction et de couverture
`graphify-out/`	graphe de connaissance généré
`context.xml`	export du dépôt pour assistant
`docs/rapport_alternance.md`	nominatif, et redondant avec `docs/22` du portfolio
`.claude/`	permissions locales personnelles
historique de contrôle de version	contient ce qui a été supprimé

Le nom d'hôte du proxy interne, présent dans le gabarit de configuration, a été remplacé par un marqueur.

Note : ce projet ne contenait aucun identifiant de locataire, de client ni de secret en dur. Toute la configuration sensible passe par le coffre-fort ou par des variables d'environnement, ce qui est le comportement attendu et se vérifie ici directement dans le code.