Engineer

Prometheus

base de données qui relève et stocke des métriques

Ce que c’est

Prometheus est un système de supervision open source centré sur sa propre base de séries temporelles : toutes les quelques secondes, il demande aux programmes surveillés leurs valeurs actuelles par HTTP, conserve chaque valeur avec un horodatage et permet d’interroger les données conservées, de les représenter graphiquement et de définir des alertes à leur sujet. Le nom désigne le serveur seul, mais tout aussi souvent la petite famille de composants qui s’est développée autour.

Il a été créé chez SoundCloud, où le travail a commencé en 2012, et appartient désormais à la Cloud Native Computing Foundation. Il l’a rejointe en 2016 comme deuxième projet accueilli par la fondation, après Kubernetes, et a obtenu le statut de projet mature (« graduated ») en 2018. Prometheus 3.0, annoncé en novembre 2024, était sa première version majeure en sept ans. Le code est principalement écrit en Go et publié sous licence Apache 2.0.

Tout ce qu’il conserve est une série temporelle, nommée par une métrique et un ensemble de labels : des paires clé-valeur qui indiquent quelle instance d’un élément a été mesurée, comme dans `api_http_requests_total{method="POST", handler="/messages"}` . Chaque échantillon de cette série est une valeur flottante sur 64 bits, ou un histogramme natif, avec un horodatage en millisecondes. Une valeur différente pour un seul label crée une autre série. C’est ce qui rend les données faciles à découper par hôte, route ou code d’état, mais aussi le système facile à surcharger.

Les programmes exposent leurs valeurs sur un point de terminaison HTTP, généralement `/metrics` , au moyen d’une bibliothèque cliente compilée dans le code ou d’un exportateur : un petit adaptateur qui traduit les données d’un système que Prometheus ne peut pas instrumenter directement, comme un hôte

Linux ou une base de données. Le serveur trouve ses cibles par découverte de services ou dans une liste statique et y collecte les valeurs à intervalles réguliers. Un traitement par lots qui se termine avant la collecte suivante peut plutôt pousser ses valeurs vers une passerelle de réception distincte (push gateway). Les règles d'alerte s'exécutent sur le serveur et les alertes déclenchées sont envoyées à Alertmanager, qui les déduplique et les regroupe, applique les mises en silence et les achemine vers le courrier électronique, PagerDuty, OpsGenie et des destinataires similaires.

Il est généralement comparé aux collecteurs fondés sur l'envoi des données par les applications, dans la tradition de Graphite et StatsD, aux bases de séries temporelles généralistes comme InfluxDB et aux services de métriques hébergés. Grafana est son interface habituelle. Il conserve des nombres plutôt que des événements : une ligne de journal ou une trace doit donc aller ailleurs. Le projet précise qu'il privilégie la fiabilité plutôt que l'exhaustivité : la facturation par requête, où chaque comptage doit être exact, relève d'un autre système.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Prometheus ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Une supervision et des alertes qui signalent les incidents tant qu'ils restent mineurs, avec les chiffres qui montrent à quelle fréquence ils le restent.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Premier tableau de bord d'observabilité — première vue de l'entreprise sur ses propres systèmes

Premier tableau de bord d'observabilité Contactez-nous →

Pourquoi c'est important

Le choix auquel Prometheus répond est d'exploiter sa propre base de métriques ou d'en louer une, et, dans le premier cas, de déterminer sa taille.

Son exploitation n'entraîne aucun droit de licence. Le coût est une machine et les heures de la personne chargée de son bon fonctionnement. Un service hébergé prend ce travail en charge et le remplace par une facture qui croît avec les données envoyées : chaque nouveau label devient alors une ligne de facture. Pour quelques dizaines de machines, le serveur autogéré est généralement la solution la moins chère et la plus simple. L'équilibre change lorsque les séries se comptent par millions et que l'équipe préfère payer plutôt qu'affecter du personnel à leur gestion.

Les besoins révélés par le calcul de dimensionnement sont plus modestes qu'on ne l'imagine souvent. La documentation du projet sur le stockage donne une moyenne d'un à deux octets par échantillon. Mille séries collectées toutes les 15 secondes et conservées pendant les 15 jours de rétention par défaut représentent 86,4 millions d'échantillons, soit environ 86–173 Mo sur disque. Le disque est rarement ce qui manque. C'est la mémoire : le bloc en cours de remplissage y réside, protégé d'un plantage par un journal d'écriture anticipée, et chaque série active y consomme des ressources, qu'elle soit interrogée ou non.

Chaque serveur est délibérément autonome. Il ne dépend d'aucun stockage réseau et continue donc à répondre pendant que l'infrastructure surveillée s'effondre, précisément au moment où il est le plus utile. En contrepartie, le stockage local n'est ni organisé en cluster ni répliqué. La haute disponibilité demande deux serveurs identiques qui collectent les mêmes cibles. Conserver un historique de plus de quelques semaines demande d'en envoyer une copie avec `remote_write` vers un stockage conçu pour la garder.

Aller chercher les valeurs plutôt que les recevoir présente aussi un avantage pratique. Puisque le serveur pose les questions, il remarque qu'une cible ne répond plus et l'enregistre par le passage de `up` à zéro. Un ingénieur peut aussi ouvrir la même page `/metrics` dans un navigateur pour voir exactement ce que le serveur voit, ce qui permet de résoudre rapidement un désaccord entre le graphique et l'application.

À quoi cela ressemble en pratique

Dans les travaux de la fondatrice, Prometheus apparaît comme le collecteur derrière un tableau de bord : une fois pour un ensemble de serveurs de bases de données et une autre dans la deuxième version d'une vue que tout le bureau regardait.

Le cas des bases de données concernait un ensemble de dix serveurs virtuels Ubuntu 20.04 LTS hébergeant PostgreSQL 14 et MS SQL Server 2019. Prometheus et Grafana suivaient en temps réel le processeur, la mémoire, les entrées-sorties disque et les performances des requêtes, à côté des sauvegardes automatisées, des règles de pare-feu et des scripts d'analyse de journaux sur mesure. Sur les dix serveurs, la disponibilité s'est maintenue à 99,9 %. Les hôtes de bases de données se prêtent bien à la collecte, car les valeurs qui y annoncent

des problèmes, un disque qui se remplit ou une requête qui ralentit, évoluent sur des heures. Un échantillon toutes les quelques secondes les révèle bien avant qu'un utilisateur ne les remarque.

Le tableau de bord du bureau montre le passage d'un collecteur maison à Prometheus. Dans un rôle antérieur, la fondatrice a construit le premier tableau de bord d'observabilité d'une organisation. Il a commencé par relever par SSH les mesures de processeur, de RAM, de disque, de température et d'autres métriques de serveurs Linux distants, et surveillait les conteneurs Docker de la même façon. Une deuxième version a été reconstruite avec Prometheus et Grafana, autour des métriques choisies avec les équipes DevOps et d'ingénierie : utilisation du processeur, mémoire utilisée, disponibilité des services et latence des API, avec des alertes lors des dépassements de seuil. Avec ce tableau de bord, les équipes détectaient et résolvaient les incidents 40 % plus vite. Interroger les machines par SSH est un bon début pour quelques machines ; un collecteur avec des labels et un langage de requête permet ensuite à la même vue de continuer à grandir.

Où les problèmes surviennent

La mémoire augmente après la publication d'une version qui n'a rien changé d'important. Quelqu'un a ajouté un label dont les valeurs ne sont pas bornées : identifiant d'utilisateur, adresse électronique, chemin d'URL complet ou identifiant de requête. Chaque nouvelle valeur est devenue une série, et le serveur en contient maintenant des millions au lieu de milliers. Le guide de nommage met précisément en garde contre cela. Limitez les valeurs des labels à de petits ensembles connus, envoyez le détail par requête vers les journaux ou les traces et repérez les pires cas avec une requête comme celle-ci :

```
topk(10, count by (__name__) ({__name__=~".+"}))
```

Un graphique montre une chute brutale ou une pointe qui n'a jamais eu lieu. Un compteur ne fait qu'augmenter et retombe à zéro au redémarrage du processus. Représenter sa valeur brute ou faire les soustractions à la main transforme chaque redémarrage en chute ou en pic. `rate()` comprend les remises à zéro : calculez donc d'abord le taux, puis la somme, sur une fenêtre large de plusieurs intervalles de collecte :

```
sum by (job) (rate(http_requests_total{status=~"5.."}[5m]))
```

Une règle d'enregistrement qui additionne un compteur avant de calculer le taux cache les remises à zéro dans la somme et fait réapparaître les fausses chutes.

Un service s'arrête et son alerte d'erreurs reste au vert. Une règle comme un taux d'erreurs supérieur à un seuil est évaluée sur les séries existantes. Lorsque la cible disparaît, ses séries aussi ; l'expression renvoie un résultat vide et l'alerte se résout d'elle-même au pire moment. Déclenchez une alerte sur `up == 0` pour chaque cible de collecte, utilisez `absent()` pour toute série qui doit toujours exister et faites vérifier par un système extérieur au serveur que le serveur lui-même fonctionne.

Les chiffres du trimestre précédent ont disparu quand quelqu'un les demande. Le stockage local conserve 15 jours par défaut, en un exemplaire sur un seul disque. Il convient comme fenêtre sur le comportement récent, mais déçoit comme archive. Augmentez la rétention avec prudence, envoyez ce qui doit durer vers un stockage à long terme par `remote_write` et gardez les chiffres dont dépendent des sommes d'argent dans une base faisant autorité.

Pour aller plus loin

- Présentation de Prometheus — La présentation du projet : ce qu'il est, ses composants et les cas auxquels il ne convient pas.
- Modèle de données — Les noms de métriques, les labels et les échantillons, et pourquoi chaque valeur de label crée une série.
- Types de métriques — Compteur, jauge, histogramme et résumé, et les questions auxquelles chacun répond.
- Bases des requêtes — Les sélecteurs PromQL, les vecteurs couvrant un intervalle de temps et les types renvoyés par une expression.
- Stockage — Les blocs de deux heures, le journal d'écriture anticipée, la rétention et l'interface remote-write.
- Nommage des métriques et des labels — Les unités de base, le suffixe `_total` et la mise en garde contre les labels à forte cardinalité.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que Prometheus ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/prometheus/