Engineer

OpenTelemetry

Envoi de télémétrie indépendant des fournisseurs

Ce que c’est

OpenTelemetry est un standard open source, accompagné de bibliothèques et d’outils, pour produire des traces, des métriques et des journaux à partir de logiciels en cours d’exécution, puis les transmettre au système qui les stocke et les affiche. La plupart des ingénieurs l’appellent OTel. C’est un projet de la Cloud Native Computing Foundation, né de la fusion de deux initiatives antérieures, OpenTracing et OpenCensus.

Le plus utile est de savoir ce qu’il laisse de côté. OpenTelemetry n’est pas un backend d’observabilité : il ne conserve pas de données, n’affiche pas de tableaux de bord et n’envoie pas d’alertes. Jaeger, Prometheus et les plateformes commerciales s’en chargent. La question « OpenTelemetry ou Prometheus ? » compare donc un conduit à une destination. La réponse habituelle est les deux : OpenTelemetry là où les données sont produites, et un backend choisi par l’équipe là où elles sont conservées.

Le projet comprend plusieurs éléments qui s’articulent :

- une **spécification** qui définit l’API, le SDK et les données, pour tous les langages ;
- **OTLP**, le protocole qui transporte la télémétrie sur gRPC ou HTTP, encodée avec Protocol Buffers ;
- des **conventions sémantiques**, des noms communs pour les mêmes choses, afin qu’une route HTTP ou une instruction de base de données soit étiquetée de la même manière en Go, Java et Python ;
- une **API et un SDK par langage**, avec des bibliothèques d’instrumentation et une instrumentation automatique pour les frameworks courants ;
- le **Collector**, un processus séparé qui reçoit la télémétrie, la traite et l’exporte vers la suite de la chaîne.

Parmi les signaux transportés, les traces demandent le plus d'explications. Une trace suit une requête dans chacun des services qu'elle traverse. Chaque étape est un segment, ou span : un nom, une heure de début, une durée, un ensemble d'attributs et l'identité du segment qui l'a provoqué. Lorsque la requête passe d'un service au suivant, cette identité l'accompagne dans un en-tête ; sans autre configuration, c'est celui que définit W3C Trace Context :

```
traceparent: 00-a0892f3577b34da6a3ce929d0e0e4736-f03067aa0ba902b7-01
```

Les quatre champs sont la version du format, l'identifiant de trace commun à tous les segments de la requête, l'identifiant du segment appelant et un ensemble d'indicateurs, où la valeur finale `01` indique que la trace est échantillonnée. Les métriques et les journaux complètent le tableau. Un quatrième signal, le bagage (baggage), transporte de petites informations sous forme de paires clé-valeur avec la requête, afin qu'un service situé trois sauts plus loin puisse encore les lire.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'OpenTelemetry ? » 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 stockage est le coût visible de l'observabilité. Le coût le plus difficile à éviter ensuite est celui de l'instrumentation : quelqu'un doit ajouter les appels à chaque service, dans chaque langage, et les tenir à jour lorsque le code change. Quand chaque fournisseur livrait son propre agent et sa propre API, changer de fournisseur obligeait à tout refaire. Avec OpenTelemetry, le code appelle une seule API, et le choix du backend passe dans la configuration : soit un exportateur du SDK, soit une route du Collector. Le projet énonce clairement ce principe : les données qu'un système produit appartiennent à celui qui l'exploite.

La propriété de ces données a des effets pratiques autant que contractuels. Le Collector se place entre les applications et le backend. C'est donc l'endroit où filtrer, échantillonner, renommer ou retirer des attributs avant tout envoi, et d'où transmettre le même flux à deux backends pendant une migration. Passer d'un fournisseur à un autre devient une modification de la configuration du Collector plutôt que de cent services.

Le volume est le coût à prévoir. Chaque requête peut produire des dizaines de segments et, avec un trafic réel, la facture de leur conservation intégrale peut dépasser celle du service observé. L'échantillonneur préconfiguré du SDK, `ParentBased(root=AlwaysOn)` , échantillonne chaque nouvelle trace et suit la décision de l'appelant pour les autres : avec ces réglages, tout est donc conservé. L'équipe doit décider délibérément de la quantité à garder.

La maturité varie selon le signal et le langage. OTLP est stable pour les traces, les métriques et les journaux, tandis que le SDK de chaque langage progresse à son rythme. La page d'état du projet indique où chacun en est. Lisez-la avant de retenir cette solution pour un service écrit dans un langage moins courant.

L'effort n'est pas toujours justifié. Une application seule, avec des journaux structurés et quelques métriques, gagne peu à faire du traçage distribué. OpenTelemetry devient rentable lorsqu'une requête franchit les frontières entre processus, à travers un frontend, une API, une file d'attente et une base de données, et que répondre à « où le temps est-il passé ? » demande une réunion.

À quoi cela ressemble en pratique

Sur une plateforme, la supervision a été construite plutôt qu'achetée, et OpenTelemetry en constitue la partie traçage. Les traces passent par OTLP : une requête lente est suivie à travers le frontend, l'API et la base de données, au lieu de faire l'objet de suppositions. À côté fonctionne un pipeline d'erreurs interne, composé d'un service errmon et d'un service d'ingestion. Il assainit les données reçues avant leur stockage, regroupe les erreurs en problèmes récurrents, détecte les pics et les régressions avec une période de temporisation pour qu'un mauvais déploiement ne déclenche pas quarante appels à la personne de garde, résout les symboles des traces de pile minifiées du frontend et exécute un signal de vie synthétique pour prouver que le pipeline lui-même fonctionne.

Les résultats arrivent dans la base de données de la plateforme elle-même sous forme d'événements d'erreur, de groupes d'erreurs, d'une boîte de réception, de latences d'API et d'échantillons de pile. Les opérateurs utilisent 11 vues sur ces données, dont une consacrée aux objectifs de niveau de service. La page est claire sur le prix : la mise en place de cette supervision a demandé du temps de travail et ajouté un élément à maintenir, tandis qu'un outil de suivi acheté aurait fonctionné dès le même après-midi. En échange, les données sensibles restent sur la plateforme. Les règles d'alerte ont, elles aussi, été écrites pour cette plateforme plutôt que pour une plateforme générique.

Où les problèmes surviennent

La trace s'arrête à une frontière. Le contexte traverse facilement les appels HTTP, car les bibliothèques d'instrumentation écrivent et lisent l'en-tête des deux côtés. Une file de messages, une tâche planifiée ou un proxy qui retire les en-têtes inconnus brise la chaîne si personne ne transmet le contexte manuellement. La trace se divise en deux, et la partie lente paraît sans rapport avec la requête qui l'a causée. Placez le contexte dans les métadonnées du message et vérifiez que chaque service utilise le même propagateur.

Les attributs divulguent des informations ou se multiplient sans contrôle. Un attribut de segment est facile à ajouter, et les valeurs les plus faciles à ajouter sont les mauvaises : adresses de courriel, URL complètes avec des jetons dans la chaîne de requête, SQL contenant encore ses valeurs littérales. Pour les métriques, le danger est la cardinalité. Un attribut qui prend des millions de valeurs distinctes, comme un identifiant d'utilisateur, multiplie le nombre de séries temporelles et la facture avec lui. Suivez les conventions sémantiques, n'incluez pas d'identifiants dans les métriques et masquez les données sensibles dans le Collector comme seconde ligne de défense.

L'échantillonnage se fait par accident. L'échantillonnage en tête décide au début d'une requête, avant de savoir si elle sera lente ou échouera. Conserver 1 % des traces conserve donc aussi environ 1 % des rares requêtes lentes. L'échantillonnage en fin de trace décide après avoir examiné ensemble les segments d'une trace et peut conserver chaque erreur et chaque requête lente, mais le composant qui s'en charge doit garder toutes ces données en attendant. C'est un service avec état, dont il faut prévoir et surveiller la capacité propre.

On fait aveuglément confiance au pipeline. OTLP accuse réception un saut à la fois ; la livraison de bout en bout dépasse son périmètre. Un Collector qui manque de mémoire ou prend du retard dans sa file d'exportation perd de la télémétrie, tandis que les applications qui lui en envoient continuent comme si tout allait bien. Surveillez les métriques du Collector lui-même, exécutez-en plusieurs instances, et traitez un trou dans les traces comme un défaut à expliquer plutôt que comme une heure calme.

Pour aller plus loin

- Qu'est-ce qu'OpenTelemetry ? — la définition donnée par le projet, ses composants et ce qu'il choisit de ne pas être.
- Spécification OTLP — le protocole sur le réseau : transports, encodage, compression et accusés de réception.
- Trace Context — la recommandation du W3C sur laquelle repose l'en-tête `traceparent` .
- Propagation du contexte — comment une trace suit une requête au-delà des frontières entre processus.
- Conventions sémantiques — les noms communs des attributs, métriques et ressources.
- Échantillonnage — échantillonnage en tête et en fin de trace, et coût de fonctionnement de chacun.

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 qu'OpenTelemetry ? » 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/opentelemetry/