Engineer

Observabilité

Comprendre un système de l'extérieur par ce qu'il émet

Ce que c'est

L'observabilité mesure à quel point une équipe peut comprendre de l'extérieur ce que fait un système en fonctionnement, en utilisant uniquement les données qu'il émet, sans l'ouvrir ni savoir à l'avance ce qui a mal tourné. L'introduction d'OpenTelemetry l'exprime en questions : un système observable permet de l'interroger sans connaître son fonctionnement interne, y compris sur ce à quoi personne ne pensait lors de l'écriture du logiciel. L'introduction résume ainsi la question à laquelle il devrait pouvoir répondre : « Pourquoi cela se produit-il ? »

L'observabilité est donc une propriété du système, pas un produit. Un outil peut stocker et afficher les données, mais elles doivent d'abord venir de l'application. La documentation OpenTelemetry est explicite : pour être observable, un système doit avoir des composants instrumentés, dont le code émet des signaux. Elle fixe ensuite un critère pratique : une application est correctement instrumentée quand ses signaux existants permettent aux développeurs d'enquêter sur un problème sans modifier d'abord l'instrumentation.

Ces données s'appellent la télémétrie ; OpenTelemetry les classe en signaux :

- **Les traces** enregistrent le trajet d'une requête dans le système. Une trace est composée de segments, ou spans, chacun étant une unité de travail avec un nom, des heures de début et de fin et des attributs comme la méthode HTTP et la route. Le premier segment est la racine et couvre toute la requête.
- **Les métriques** sont des mesures prises à l'exécution et agrégées dans le temps : taux d'erreurs, débit de requêtes, utilisation du processeur.
- **Les journaux** sont des enregistrements horodatés d'événements, dont beaucoup ne sont liés à aucune requête particulière.
- **Le bagage contextuel (baggage)** est un ensemble d'informations contextuelles transmis avec une requête pour être utilisé par les autres signaux.

La page des signaux cite aussi les événements, type particulier d'enregistrement de journal, et les profils, enregistrements de l'utilisation des ressources au niveau du code, comme étant en développement ou au stade de proposition.

Ce qui transforme trois tas de données en un ensemble interrogeable, c'est leur corrélation. OpenTelemetry propage le contexte entre services : un segment connaît ainsi sa trace et les identifiants de la trace et du segment actifs peuvent être ajoutés à chaque enregistrement de journal. Chaque signal porte aussi une ressource : des attributs identifiant l'entité qui l'émet, comme le nom du service, l'hôte ou le conteneur. Avec ces clés communes, une équipe passe d'une métrique aux requêtes correspondantes, puis d'une requête aux lignes de journal qu'elle a produites.

L'observabilité et la supervision se recoupent sans se confondre. La supervision recherche des conditions jugées importantes à l'avance et déclenche une alerte si l'une apparaît. L'observabilité est ce dont dispose l'équipe une fois l'alerte déclenchée, quand la cause n'avait été prévue par personne.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que l'observabilité ? » 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

Les systèmes modernes échouent de façons difficiles à prévoir. Une requête peut traverser une passerelle, plusieurs services, une file d'attente et une base de données avant la réponse ; un défaut dans l'un peut apparaître comme une lenteur ailleurs. L'introduction juge le traçage distribué essentiel pour ces systèmes, dont les problèmes sont souvent non déterministes ou trop complexes à reproduire sur le poste d'un développeur. Quand un échec est impossible à reproduire, la télémétrie enregistrée pendant sa survenue est la seule preuve disponible.

Tableaux de bord et alertes répondent aux questions anticipées. C'est utile et cela couvre les échecs déjà vus par l'équipe. Les incidents difficiles sont les nouveaux : la question n'est plus de savoir si le taux d'erreurs a augmenté, mais pourquoi les requêtes d'une région, passant par une version vers un prestataire de paiement, ont commencé à échouer. Si répondre exige d'écrire du nouveau code, de le déployer et d'attendre que le problème se reproduise, l'indisponibilité dure autant que cet aller-retour.

L'observabilité tient aussi compte de l'utilisateur. L'introduction définit la fiabilité comme la capacité du service à faire ce qu'attendent ses utilisateurs, avec cet exemple : une boutique peut être toujours disponible et rester peu fiable si cliquer pour ajouter des chaussures noires au panier n'ajoute pas toujours des chaussures noires. Une métrique d'hôte ne le montrerait jamais ; une télémétrie par requête le peut. Elle fonde les indicateurs de niveau de service : mesures prises du point de vue de l'utilisateur, que les objectifs de niveau de service rattachent ensuite aux besoins de l'activité.

Enfin, elle change le coût d'un incident. Une équipe capable de suivre une requête lente jusqu'à l'appel précis qui l'a ralentie passe son temps à corriger plutôt qu'à deviner ; le compte rendu ultérieur peut citer des preuves plutôt que des souvenirs.

À quoi cela ressemble en pratique

Prenons une boutique en ligne fictive. Depuis 14 h 00, une petite part des parcours de commande dépasse deux secondes. Rien n'est en panne, aucune alerte n'a été déclenchée et aucun tableau de bord n'a été prévu pour cela. Voici comment une équipe disposant de signaux corrélés enquêterait. Tous les noms, identifiants et nombres qui suivent sont inventés pour l'exemple.

La première étape est l'histogramme de latence du parcours de commande, mesurée en secondes. Dans le format textuel OpenMetrics, une tranche peut porter après `#` un exemplaire (exemplar) : référence à une donnée extérieure à la métrique, généralement l'identifiant d'une trace tombée dans cette tranche. Deux tranches donnent :

```
checkout_bucket{le="1.0"} 95
checkout_bucket{le="2.5"} 97 # {trace_id="3f1c9a77e2b04d18a6c5e0b9d4127f60"} 2.3
```

La deuxième étape suit cet identifiant de trace. La trace montre où sont passées les 2,3 secondes, segment par segment :

```
POST /checkout                      2.30 s   service checkout
  SELECT cart                       0.04 s   service checkout
  POST /authorise                   2.21 s   service payments
    event: provider timeout, retrying
  INSERT order                      0.03 s   service checkout
```

La troisième étape récupère les lignes de journal portant le même identifiant de trace. La spécification OpenTelemetry pour les journaux écrits dans d'autres formats que son propre protocole demande de conserver les identifiants comme champs JSON de premier niveau nommés `trace_id` et `span_id`, en minuscules et encodés en hexadécimal :

```
{
    "timestamp": 1793110051.402,
    "severity": "WARN",
    "body": "provider timeout after 2000 ms, retrying",
    "provider": "acquirer-b",
    "trace_id": "3f1c9a77e2b04d18a6c5e0b9d4127f60",
    "span_id": "a91c04e7d2f3b658"
}
```

Trois étapes ont répondu à la question imprévue : l'un des deux prestataires de paiement a commencé à dépasser les délais et chaque parcours de commande qui lui était adressé attendait une nouvelle tentative. Le même attribut, `provider`, permet maintenant de regrouper toutes les traces récentes pour confirmer ce comportement avant toute modification du code.

Rien de cela ne fonctionne si les services ne s'identifient pas. Les SDK OpenTelemetry définissent par défaut l'attribut de ressource `service.name` à `unknown_service` ; la documentation recommande de le fixer explicitement, dans le code ou par variable d'environnement. D'autres attributs de ressource s'ajoutent de la même façon :

```
env OTEL_SERVICE_NAME=checkout \
    OTEL_RESOURCE_ATTRIBUTES=deployment.environment.name=production \
    ./checkout
```

Sur ce site, trois pages du portfolio montrent la construction progressive de l'observabilité. L'une décrit le premier tableau de bord d'observabilité d'une entreprise, qui collectait par SSH les métriques système de serveurs Linux distants, comme processeur, mémoire et disque ; une deuxième version a ensuite été bâtie sur Grafana et Prometheus. Le travail pour réduire le risque opérationnel avec Grafana et Prometheus a instrumenté les services clés et choisi des métriques utiles plutôt que flatteuses. Une page ultérieure, supervision des erreurs et traçage en propre, a ajouté le traçage OpenTelemetry sur OTLP, en parallèle d'un pipeline d'erreurs qui assainit les contenus avant stockage. Une requête lente peut ainsi être suivie dans le frontend, l'API et la base, au lieu de faire l'objet de suppositions.

Où les problèmes surviennent

Acheter l'observabilité au lieu de l'intégrer. Un nouvel outil dans lequel aucune donnée utile n'arrive est une pièce vide. L'observabilité vient de l'instrumentation du code et des bibliothèques : prévoyez le travail d'ingénierie pour ajouter segments, attributs et champs de journal, pas seulement la licence.

Trois signaux, trois îlots. Les métriques dans un outil, les journaux dans un autre et les traces dans un troisième, sans clé commune, obligent l'équipe à copier des horodatages d'un écran à l'autre. Ajoutez les identifiants de trace et de segment à chaque enregistrement de journal, attachez des exemplaires aux histogrammes de latence si la pile le permet et donnez à tous les signaux les mêmes attributs de ressource.

Tous les services s'appellent unknown_service. C'est la valeur par défaut du SDK : les services d'une flotte qui la partagent ne peuvent être distingués les uns des autres. Définissez `OTEL_SERVICE_NAME` pour chaque service. La ressource est attachée à la création des fournisseurs de traces et de métriques et ne peut plus changer ensuite : elle doit donc être correcte au démarrage.

Seules les frontières sont instrumentées. L'instrumentation sans code est un bon début : elle enregistre ce qui se passe aux frontières d'une application, dans ses bibliothèques et son environnement d'exécution. Les questions d'un incident portent généralement sur le métier : niveau d'offre du client, prestataire, valeur d'un indicateur de fonctionnalité. Cela exige des attributs ajoutés dans le code. OpenTelemetry présente l'instrumentation par code comme complément de celle sans code ; les deux peuvent fonctionner ensemble.

L'instrumentation se fait pendant l'incident. Si répondre exigeait l'ajout d'une ligne de journal et un déploiement, le système n'était pas correctement instrumenté selon le critère de l'introduction. Posez systématiquement la question « Qu'avons-nous dû ajouter ? » dans chaque retour d'incident et faites de chaque réponse une action de suivi.

Des détails sans limite vont dans le mauvais signal. Les identifiants d'utilisateurs, URL complètes et corps des requêtes ont leur place dans les segments et journaux, pas dans les étiquettes de métriques, où chaque nouvelle valeur crée une série temporelle. La norme OpenMetrics plafonne le total des noms et valeurs d'étiquettes d'un exemplaire à 128 caractères, précisément pour empêcher d'y loger des données de segment, et permet à un système d'ingestion d'écarter tous les exemplaires. Vérifiez que les liens survivent sur tout le trajet, de l'exportateur à l'écran, avant de vous y fier.

La télémétrie divulgue des données. Segments et journaux enregistrent le contenu des requêtes, qui peut inclure adresses, noms et jetons. Décidez des attributs enregistrables, retirez le reste avant stockage et protégez la télémétrie avec autant de soin que la base qu'elle décrit.

Pour aller plus loin

- Introduction à l'observabilité — définition OpenTelemetry, avec fiabilité, SLI et SLO, segments et traces distribuées.
- Signaux — traces, métriques, journaux et bagage contextuel, ainsi que les signaux encore en développement.
- Instrumentation — instrumentation avec et sans code, et propagation du contexte reliant les signaux.
- Ressources — attributs de ressource, valeur `unknown_service` par défaut et variables d'environnement pour les définir.
- Contexte de trace dans les formats de journal autres qu'OTLP — spécification stable pour enregistrer les identifiants de trace et de segment dans syslog, le texte brut et JSON.
- OpenMetrics 1.0 — norme du format d'exposition, dont les exemplaires et les conseils de 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 l'observabilité ? » 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/observability/