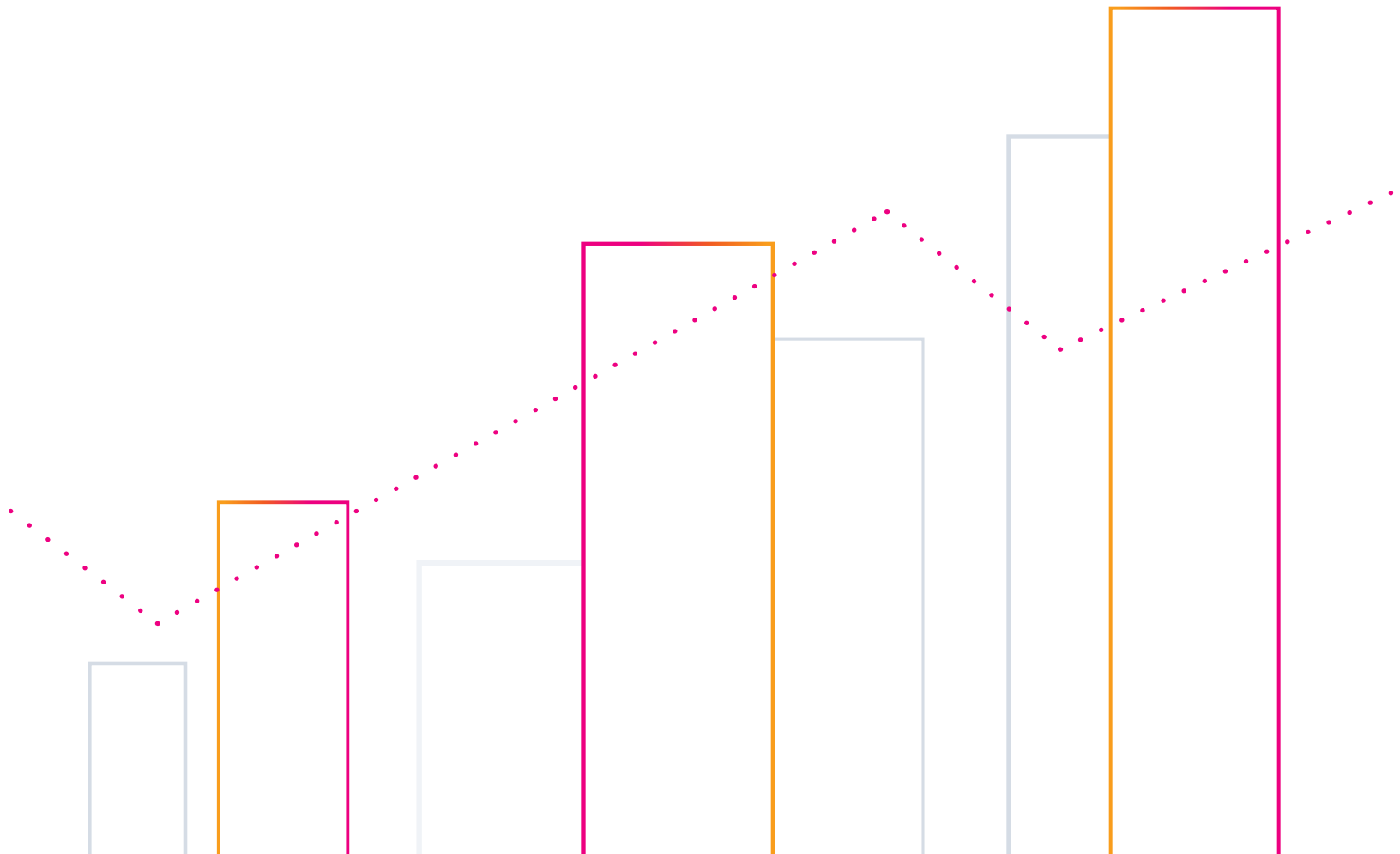
Les API en action

Guide de supervision des performances des API



Sommaire

Introduction

Qu'il s'agisse de l'essor des microservices, de la propagation des applications Web à page unique ou de la domination constante des applications mobiles natives, les API sont les héros méconnus du Web moderne.

Au début, les API offraient un moyen pour les entreprises de se différencier et de s'intégrer à d'autres systèmes, mais certains affirment qu'elles ne se sont véritablement répandues qu'avec l'essor des réseaux sociaux. Avec le développement des réseaux sociaux et l'élargissement de leur base d'utilisateurs, les entreprises ont commencé à créer et à exploiter des API pour permettre de partager et d'intégrer du contenu dans les flux d'interactions des réseaux sociaux.

Aujourd'hui, nous utilisons des API pour connecter plusieurs plateformes de réseaux sociaux (entre autres pour partager sur Facebook ou Twitter une publication d'Instagram) ou pour nous connecter à des ensembles de données, par exemple pour enregistrer notre position ou commander un trajet en se géolocalisant. De nombreuses applications Web et mobiles que nous utilisons aujourd'hui reposent sur d'autres API dont elles dépendent étroitement ; ces API exécutent et renvoient des données correctement, rapidement, en toute sécurité et de manière fiable.

Comme les API sont le moteur des applications et des infrastructures modernes, il est essentiel de superviser leurs performances.

Dans ce livre blanc, nous allons vous aider à comprendre les fonctions de base des API, pourquoi il est important de superviser les API et quels types de fonctionnalités rechercher lors de la mise en œuvre de systèmes de supervision des performances des API.

Comprendre les API

Une API est un ensemble d'instructions et de normes de programmation permettant d'accéder à une application logicielle ou à un service web. Les API donnent aux développeurs des instructions sur les modalités d'interaction avec les services et peuvent être utilisées pour connecter des données entre différents systèmes. Une API décrit les fonctionnalités disponibles à partir d'un service, comment l'utiliser et y accéder, et à quels formats il accèdera pour obtenir des entrées et des sorties.

Aujourd'hui, des entreprises et des applications entières reposent sur des API ouvertes pour échanger des données entre différents systèmes. Vous pouvez voir une API comme un serveur au restaurant. Au restaurant, vous pouvez faire votre choix parmi un menu. Peut-être avez-vous aussi des instructions spécifiques quant à la préparation de votre commande. Lorsque le serveur vous demande votre commande, vous dites : « Je voudrais le filet mignon, à point, avec des épinards à la crème et une pomme de terre au four bien garnie sans ciboulette ».

Votre serveur note votre commande, l'apporte en cuisine, récupère votre plat lorsqu'il est prêt et vous le sert à table.

Dans ce scénario simple, le rôle du serveur s'apparente à celui d'une API, sauf qu'au lieu de livrer un délicieux repas, une API fournit des données. Comme un serveur, l'API prend un ensemble d'instructions (une requête) auprès d'une source (une application ou un développeur), amène cette requête à une base de données, récupère les données ou facilite certaines actions, puis renvoie une réponse à la source. Les API sont des messagers qui assurent la connexion des systèmes.

Lorsqu'il s'agit d'envoyer et de recevoir des messages d'API, il faut savoir qu'il existe des API privées, utilisées au sein d'une organisation interne, et des API publiques, tournées vers les consommateurs. Les API privées rendent les entreprises plus agiles, flexibles et puissantes.

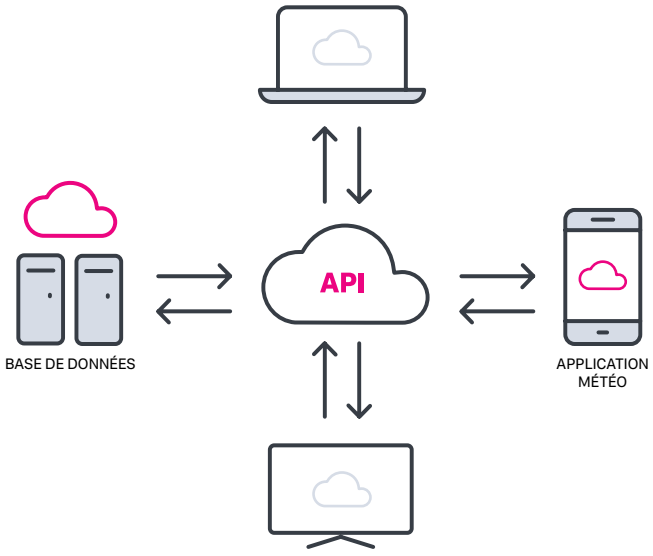
Les API publiques aident les entreprises à se connecter pour proposer de nouvelles intégrations et établir de nouveaux partenariats.

Les API en action

Prenons l'exemple d'une entreprise bien connue qui utilise des API publiques.

The Weather Company collecte des données météorologiques à partir de millions de terminaux et de sources à travers le monde. L'entreprise stocke ces données et les met à disposition d'applications destinées aux consommateurs via des centaines d'API différentes.

Pensez à un widget, sur votre écran d'accueil ou votre bureau, qui affiche toujours la température et les conditions météorologiques actuelles en fonction de votre emplacement. Votre appareil ne possède pas forcément de thermomètre, de baromètre ou de technologie intégrée capable de détecter ou prédire les conditions météorologiques, mais il peut envoyer des informations sur votre position à une API et obtenir en retour des données correctes comme la température et les prévisions météo en fonction de votre position. Même si vous n'êtes pas sur le site Web de The Weather Company ou que vous n'utilisez pas l'une des applications mobiles natives de The Weather Company, vous pouvez interagir avec les données d'une API de The Weather Company répondant à des requêtes simples.



Comparons maintenant ce scénario d'API publique à un exemple où l'API est utilisée pour déclencher des tâches afin d'améliorer des processus métier internes.

Splunk Web Optimization capture un certain nombre de métriques de performance pour les entreprises numériques et fournit des suggestions intelligentes pour l'amélioration des performances des sites. L'outil d'optimisation peut aider les équipes à identifier quand des modifications apportées à un site Web entraînent une dégradation des performances, indiquant qu'un site ou une application web présente du contenu qui ralentit son chargement.

À l'aide de l'API de l'outil d'optimisation, l'équipe de Splunk peut intégrer des tests de performances dans les déploiements et identifier les dégradations de performances introduites. Une requête adressée à l'API d'optimisation a été intégrée au processus de déploiement ; elle déclenche une analyse des performances dans l'environnement de staging de Splunk et publie les résultats dans notre canal de discussion Slack. Nous pouvons maintenant utiliser notre configuration ChatOps existante pour informer l'équipe lorsque ce que nous développons en staging ralentit l'application.

Dans ce scénario, l'API ne se contente pas de renvoyer des données à la suite d'une requête, elle effectue une action. L'API Weather Company comme l'API Splunk pourraient être publiques ou destinées aux consommateurs, ce qui signifie que d'autres systèmes pourraient en dépendre pour renvoyer des données ou déclencher des actions spécifiques.

CHAPITRE 1

SOAP, REST et JSON

À la base, les API sont conçues pour demander et recevoir des données d'un service distant. Dans cet e-book, nous allons nous pencher sur les API web, avec lesquelles la demande et la réponse se font via HTTP. Il existe de nombreuses façons de mettre ces demandes et réponses en forme. Si vous lisez des informations sur les API, vous rencontrerez un large éventail d'acronymes et de termes spécifiques. La plupart du temps, ils ne sont pas utiles pour comprendre les concepts essentiels des API, leur importance et les méthodes permettant de les tester et de superviser leurs performances.

Voici quelques grandes lignes qui vous aideront à vous y retrouver :

- les différents formats d'API peuvent appliquer différentes structures au corps des demandes et des réponses ; SOAP est très structuré et utilise XML. REST peut utiliser n'importe quoi, mais utilise généralement JSON ;
- les différents formats d'API peuvent utiliser différentes méthodes HTTP lors de la création de requêtes. Une API comme SOAP utilisera presque toujours POST, tandis que REST peut utiliser GET, POST ou même des méthodes moins courantes comme PUT ou DELETE ;
- ces différents formats peuvent transmettre leurs informations d'identification ou d'authentification de différentes manières. Ils peuvent employer des cookies, des en-têtes HTTP spéciaux, ou même des paramètres de chaîne de requête.

Si tout cela vous semble confus, rappelez-vous simplement : les API sont conçues pour envoyer une demande et récupérer des données. Tout le reste découle de cette idée simple.

CHAPITRE 2

Pourquoi superviser les API

Un peu plus tôt, nous avons abordé quelques exemples de systèmes qui dépendent des données des API ou dont certaines fonctionnalités de base reposent sur des API tierces.

Les API sont merveilleuses car elles connectent les services et nous permettent de transmettre des données entre des systèmes gérés indépendants, mais cette connectivité et cette interdépendance créent des vulnérabilités. En ce qui concerne les performances et la disponibilité, il est important de superviser les API dont nous dépendons et celles que nous fournissons aux autres.

L'importance des performances des API

Lorsqu'une API est défaillante et perturbe les performances ou l'expérience utilisateur de votre site, cet échec se répercute sur votre entreprise. Les utilisateurs finaux et les clients ne sauront probablement pas qu'un tiers peut être en faute. Et selon l'importance de cette API dans un processus de transaction, cet échec peut avoir un impact immédiat sur vos résultats.

Par exemple, si un élément clé du processus de paiement sur votre site web est une recherche basée sur la localisation et que vous comptez sur une API tierce pour l'exécuter, lorsque cette API ne fonctionne pas correctement, vos clients potentiels ne peuvent pas finaliser leur achat.

Pour prendre un autre exemple, imaginez que vous avez développé une application qui nécessite une authentification à partir d'une plateforme de réseaux sociaux. Si l'API de cette plateforme subit une interruption, il se peut que vos utilisateurs ne puissent pas se connecter à votre système.

En tant que développeur ou propriétaire de site, vous pouvez décider que l'avantage offert par le service tiers l'emporte sur le risque de ce type d'échec. Afin d'évaluer le risque avec précision et d'avoir une visibilité sur l'impact de ces services au fil du temps, il est crucial de superviser la partie du flux utilisateur de votre site qui repose sur une API.

Si vous avez créé une API ouverte et que vous l'avez mise à la disposition de partenaires ou de développeurs, vous avez la responsabilité de vous assurer que l'API est disponible et fonctionne comme prévu.

Disons par exemple que vous avez développé une nouvelle API interne qui transmet les données de commande d'un appareil mobile à un système de votre entrepôt. Il est peut-être essentiel que les données soient transmises à l'entrepôt en deux minutes, sans quoi tout le planning de production sera perturbé. Lorsque vous avez développé l'API et que vous l'avez testée lors de la phase de staging, les données ont toujours été transmises en une minute, mais lorsque vous lancez l'API et commencez à traiter des demandes réelles, vous remarquez que le temps de réponse réel se rapproche de plus en plus de ce seuil de deux minutes. Sans une supervision active de l'API en production, votre équipe peut penser que l'API développée est suffisamment rapide sur la base des tests de pré-production.

Remarque : si vous hébergez une API utilisée par d'autres personnes, veillez à superviser activement cette API dans les environnements de pré-production et de production.

Qu'il s'agisse de garder un œil sur vos propres API ou de voir l'impact des services externes que vous utilisez, il est important de superviser la disponibilité, la fonctionnalité, la vitesse et les performances des API. Si vous savez que vous utilisez une API qui a été instable par le passé et que vous ne la supervisez pas activement, parlez sans attendre à votre équipe et élaborez un plan pour commencer à superviser cette API aujourd'hui.

Ce qu'il faut superviser

Maintenant que vous comprenez pourquoi il est crucial de superviser les performances des API, vous savez également que vous devez prendre en compte à la fois :

- les API sur lesquelles votre site web ou votre application native s'appuie pour obtenir des données ou accomplir des processus critiques, et
- les API que vous gérez et sur lesquelles des clients, des utilisateurs finaux ou des développeurs s'appuient pour obtenir des données ou accomplir des processus.

Pour superviser ces deux types d'API, il est important de tester :

- **la disponibilité.** Ce point de terminaison d'API est-il actif ? Est-ce qu'il renvoie une erreur ?
- **le délai de réponse.** À quelle vitesse l'API renvoie-t-elle des réponses ? Le temps de réponse se dégrade-t-il avec le temps ? Le temps de réponse est-il moins bon en production qu'en pré-production ?
- **la validation des données.** L'API renvoie-t-elle les bonnes données au bon format ?
- **les processus en plusieurs étapes.** Puis-je enregistrer et réutiliser avec succès une variable à partir de cette API ? L'authentification fonctionne-t-elle comme prévu ? Puis-je effectuer une transaction avec les données de cette API ?

Ce ne sont que les aspects de base que votre équipe doit examiner pour superviser les performances d'une API. Dans la section suivante, nous verrons comment implémenter techniquement la supervision des API et quelles caractéristiques sont essentielles pour créer des tests de performances robustes et flexibles.

CHAPITRE 3

Comment superviser les API

Si vous avez accès à un système de supervision externe et proactif, superviser la réponse d'une API pour vérifier sa disponibilité peut être assez simple et facile à exécuter, au moyen de contrôles basiques de disponibilité ou d'un ping. Définissez le protocole ou installez un point de terminaison, configurez le test pour qu'il s'exécute fréquemment à partir de différentes régions du monde et créez des alertes en cas de mauvais code de réponse ou de latence.

Mais que faire si vous devez superviser plus que la disponibilité ?

Votre solution de supervision devra fournir un certain nombre de fonctionnalités clés pour tester pleinement les transactions par API. Voici les composants de requêtes API typiques que vous devrez configurer dans vos tests au-delà de la simple question « Quel point de terminaison dois-je atteindre ? »

En-têtes de requête

Selon le fonctionnement d'un site ou d'une application, les en-têtes de requête peuvent constituer une partie cruciale des spécifications de la configuration d'un test de supervision active, afin de simuler efficacement une transaction. Par exemple, si nous devons tester activement le fonctionnement d'un processus de paiement lorsqu'un visiteur reçoit un cookie, nous aurons besoin d'un moyen d'assigner un en-tête de requête à ce cookie pour développer un test actif sur cette transaction.

Quand nous parlons d'en-têtes de requête, nous faisons référence aux champs transmis dans les sections d'en-tête des requêtes HTTP. Les en-têtes de requête peuvent inclure des règles et des paramètres qui définissent le fonctionnement d'une transaction HTTP.

Il existe un ensemble standard de types d'en-tête de requête, et ils ont des noms et des objectifs spécifiques. Voici quelques exemples courants d'en-têtes de requête :

- **Authorization :** envoie des identifiants pour une authentification HTTP de base afin d'autoriser l'accès.
- **Cache-Control :** indique au navigateur pendant combien de temps une ressource peut être mise en cache et réutilisée.
- **Content-Type :** indique à un serveur le type MIME du corps d'une requête afin qu'il sache comment interpréter les données.
- **Cookie :** définit un cookie à stocker dans le navigateur afin de suivre un état ou une session.

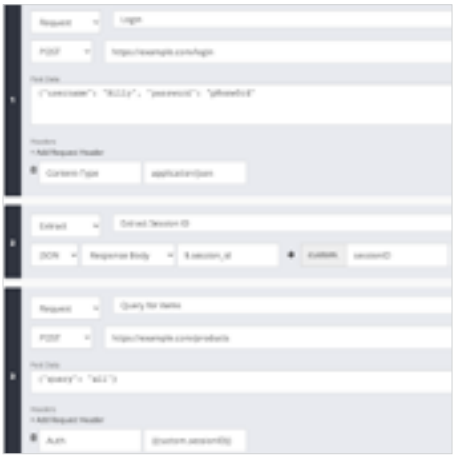
Certains développeurs peuvent également implémenter des en-têtes de requête personnalisés aux noms spécifiques. Il est courant de voir des en-têtes de requête personnalisés avec le préfixe « X » : par exemple, X-Http-Method-Override permet de remplacer une méthode de requête (comme POST) par une autre (comme PUT ou DELETE).

Les en-têtes de requête dans les vérifications d'API

API Check de Splunk nous aide à superviser la disponibilité, le temps de réponse et la qualité des données dans les transactions avec API. Avec une vérification d'API, nous pouvons définir des en-têtes pour chaque requête d'une transaction.

Imaginons que nous devions envoyer via POST un nom d'utilisateur et un mot de passe pour accéder à certaines informations. Ensuite, une fois connectés au point de terminaison, nous devons stocker et définir un ID de session afin de renseigner préalablement d'autres composants spécifiques à notre session.

Lors de la création des étapes d'une vérification d'API, nous pouvons cliquer sur + Add a Request Header (Ajouter un en-tête de requête) pour fournir un ou plusieurs en-têtes à chaque étape de requête.



Dans l'exemple ci-dessus, nous utilisons la vérification d'API de Splunk pour :

1. demander l'envoi via POST du nom d'utilisateur et du mot de passe à un point de terminaison pour vous connecter ;
2. extraire un ID de session du corps de la réponse à l'aide du chemin JSON et enregistrer cet ID en tant que variable réutilisable dans les étapes suivantes ;
3. demander un envoi via POST à un autre point de terminaison, en insérant l'ID de session dans les en-têtes de requête.

Nous pourrions ajouter d'autres étapes fonctionnelles à cette transaction ou une étape Assert pour confirmer que l'ID de session est défini comme prévu.

Gestion de l'authentification

Dans l'exemple ci-dessus, nous avons utilisé des en-têtes de requête pour envoyer un nom d'utilisateur et un mot de passe à des fins d'authentification. Prenons une minute pour nous concentrer sur l'aspect sécurité des API et sur sa prise en compte lors de la création de tests de performances.

Pour une API, l'authentification définit qui est autorisé à accéder à des données ou à des points de terminaison sécurisés. C'est particulièrement important pour les API qui partagent des informations sensibles, celles qui permettent aux utilisateurs finaux d'apporter des modifications et pour les entreprises qui facturent l'accès aux données via l'API. Et bien qu'il soit déjà complexe de sécuriser une API pour un utilisateur final humain, il faut tenir compte de facteurs supplémentaires lorsque nous devons authentifier un nombre croissant d'entités non humaines auprès d'un système.

Les API devenant plus sécurisées, les systèmes de supervision proactive s'adaptent pour permettre d'accéder à des systèmes sécurisés depuis l'extérieur.

Authentification directe

L'authentification de base HTTP offre un bon exemple d'authentification directe. L'authentification de base HTTP fait partie intégrante de HTTP et peut être utilisée pour les points de terminaison d'API ou toute URL HTTP. Vous envoyez simplement un nom d'utilisateur et un mot de passe (encodés ensemble en base64) dans le cadre de votre requête d'API. L'avantage de l'authentification de base HTTP est qu'elle est facile à mettre en œuvre et qu'elle est généralement incluse dans les frameworks standards. Par contre, l'authentification de base HTTP n'offre aucune option avancée et peut être facilement décodée.

Pour donner un autre exemple d'authentification directe, citons les clés ou les jetons API. Les clés API ne sont qu'une longue chaîne de chiffres hexadécimaux (par exemple 34d83d84f28d146aeae0e32f7803c88d) qui peut être envoyée à la place d'un nom d'utilisateur ou d'un mot de passe pour authentifier l'accès à un point de terminaison API. Les clés API s'apparentent essentiellement à un ensemble regroupant un nom d'utilisateur et un mot de passe, mais elles fournissent une couche d'abstraction utile. Par exemple, plusieurs utilisateurs finaux peuvent partager une même clé API.

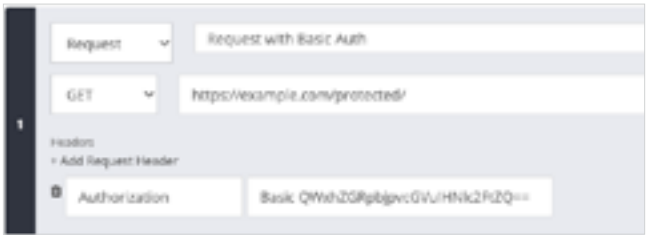
Lorsque vous utilisez n'importe quel type d'authentification directe, il est important que vous utilisiez également SSL/TLS ou https:// au début de l'URL du point de terminaison de l'API. L'utilisation de SSL/TLS évite d'exposer les informations d'authentification HTTP de base ou les clés API dans l'URL.

Vous souhaitez en savoir plus sur SSL/TLS et comment optimiser les performances ? Vous trouverez à ce sujet **plusieurs excellents articles sur le blog Zoompf**.

Authentification HTTP de base

Si vous utilisez l'authentification de base pour sécuriser vos API, il est très facile de l'inclure lors de la configuration d'un moniteur externe pour vérifier les performances de l'API.

Le moyen le plus courant et le plus fiable de configurer une demande de supervision avec une authentification HTTP de base consiste à encoder la valeur username:password (nom d'utilisateur:mot de passe) en base64 et à envoyer cette valeur dans un en-tête d'autorisation :



Il est important de noter que s'il est facile d'encoder des noms d'utilisateur et des mots de passe en base64, il est également très simple de les décoder pour authentifier une requête. Vous pouvez essayer vous-même avec **un encodeur/décodeur base64 en ligne**. Étant donné l'accessibilité de base64, il est important de protéger ce type d'authentification directe avec SSL/TLS.

Clés d'API

Du point de vue de la supervision, il est assez simple de reproduire le processus consistant à atteindre un point de terminaison avec une clé d'API dans l'URL ou avec des en-têtes de requête. Fournissez la clé et rappelez-vous simplement que si elle change, vous devrez également mettre à jour la configuration de votre test de supervision.

Notez que différents systèmes peuvent accepter les clés API de différentes manières (par exemple, dans le cadre des données POST plutôt que dans un en-tête de requête) ; vérifiez l'API que vous supervisez pour comprendre comment transmettre correctement la clé d'API.

Authentification basée sur des tickets

Bien que l'authentification directe présente des avantages pratiques indéniables, nous devrons peut-être ajouter une couche de sécurité supplémentaire à nos API. Les systèmes d'authentification basés sur des tickets s'appuient sur des serveurs d'authentification centraux qui agissent comme intermédiaires : ils acceptent les informations d'identification des utilisateurs finaux, puis renvoient des tickets, des jetons ou des clés qui permettent à un utilisateur final spécifique d'accéder uniquement à des données sécurisées particulières. L'authentification basée sur les tickets est idéale pour tout scénario dans lequel vous protégez des informations sensibles, car elle permet à une API de créer des objets ou d'apporter des modifications. Elle est aussi très pratique si vous facturez l'utilisation de votre API.

Comprendre les systèmes basés sur les tickets

On peut comparer l'authentification basée sur les tickets à la façon dont on peut obtenir des clés pour essayer une voiture.

Imaginons que je vends mon vieux tacot sur leboncoin. Vous me retrouvez au café près de chez moi pour faire un test de conduite avec ma voiture. Vous me montrez votre permis de conduire. Je dis : « Ah ! Vous êtes bien la personne sympathique que je viens de rencontrer sur Internet. Je vous fais entièrement confiance. Voici la clé. Faites un tour et ramenez-la dans quelques minutes. » Cette approche est comparable à l'authentification directe dans le sens où je vous donne la clé de voiture directement, sur la seule base de votre nom.

Imaginez maintenant que je vous vends un véhicule de luxe chez un concessionnaire. Vous me retrouvez chez le concessionnaire et vous me présentez votre permis de conduire. Je n'ai pas de clé capable de faire démarrer toutes les voitures du parking. Je dois prendre votre permis de conduire et l'utiliser pour enregistrer vos informations dans un système informatique qui vérifiera que vous êtes bien une personne honnête. Cette opération déverrouille ensuite une boîte dont je pourrai sortir une clé qui ne fonctionnera que pour le véhicule que vous devez essayer. Cette méthode s'apparente à un système basé sur ticket dans le sens où je m'appuie sur un système centralisé pour remettre une clé qui est maintenant liée à vous via le ticket.

OAuth, Kerberos, l'authentification unique et les formulaires Web sont autant d'exemples de systèmes d'authentification basés sur des tickets. Vous pouvez même développer votre propre système d'authentification personnalisé. Bien qu'il existe de nombreuses manières d'implémenter ce type de protocole, la plupart des systèmes basés sur des tickets partagent une structure similaire : vous effectuez d'abord une demande de ticket ou de jeton, puis vous utilisez ce ticket ou ce jeton pour accéder à des données ou à des terminaux sécurisés.

Dans les systèmes d'authentification basés sur les tickets, les tickets ressemblent beaucoup aux clés API dont nous avons parlé plus haut. La principale différence est que les tickets sont éphémères. Ils ne sont valables que pour une courte période et peuvent être facilement révoqués, ce qui offre une couche de sécurité supplémentaire.

Supervision avec l'authentification basée sur des tickets

Afin de superviser efficacement une API qui utilise l'authentification par ticket, vous devez pouvoir effectuer plusieurs étapes et enregistrer le ticket ou le jeton dans une variable réutilisable dans les étapes suivantes.

Pour prendre un exemple simple, il s'agit de faire une requête avec un nom d'utilisateur, un mot de passe et un certain type de spécification dans l'en-tête, puis de récupérer un jeton dans le système, d'enregistrer ce jeton en tant que variable, puis de faire une autre requête auprès d'un point de terminaison avec ce jeton comme en-tête.

Si vous n'avez pas encore implémenté une authentification pour votre API, vous devez impérativement le faire pour protéger vos données et vos systèmes. Et lorsque vous augmentez la sécurité, assurez-vous que vos systèmes de supervision externes ont également la permission et la capacité de superviser les performances et la fiabilité de votre système. Si vous ne supervisez que les performances de votre API du côté de l'application, vous risquez de passer à côté de toutes sortes de problèmes de connectivité empêchant vos utilisateurs finaux d'accéder aux données ou de faire des modifications via votre API.

Supervision et validation des données

Lorsque nous supervisons un site Web dans un navigateur, nous voulons en savoir plus que le seul code de réponse et confirmer que certains contenus ou images se chargent sur la page. Si la page renvoie « 200 OK » mais qu'elle est complètement vide, il faut s'en occuper sans attendre. Le même principe s'applique lorsque nous supervisons les points de terminaison de l'API. Lorsqu'on supervise les points de terminaison de l'API, il faut non seulement confirmer que le code de réponse est conforme aux attentes, mais aussi que les bonnes données sont renvoyées dans le bon format. Examinons un scénario d'utilisation simple de l'utilisation des options de base d'extraction (extract) et d'assertion (assert) pour vérifier qu'une API renvoie des données au format correct.

Exemple de validation de données

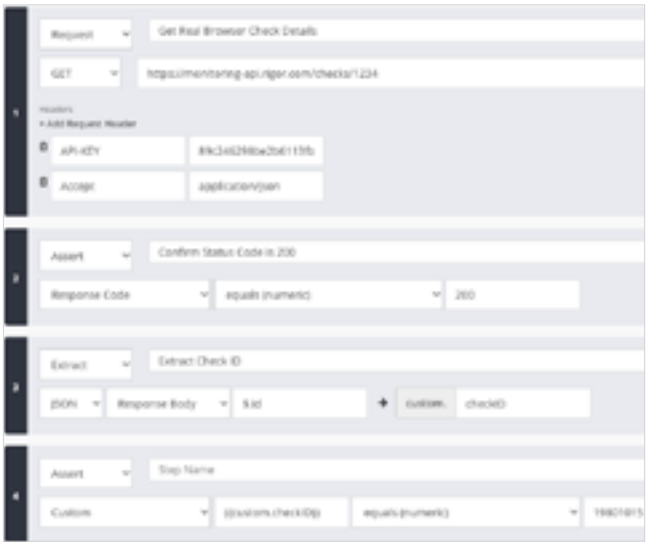
Splunk Optimization propose **une API ouverte** qui permet d'extraire régulièrement des données à des fins de rapport. Lorsqu'un utilisateur Splunk atteint le point de terminaison pour effectuer une vérification avec une clé API, nous devons renvoyer un code de réponse « 200 OK » et l'ensemble de données du bon ID qui correspond au point de terminaison.

Nous pouvons créer un test synthétique externe pour atteindre le point de terminaison de contrôle à une fréquence définie et à partir de plusieurs emplacements, et confirmer que :

1. code de réponse = 200, et
2. l'ID de vérification inclus dans la sortie JSON correspond au point de terminaison d'URL que nous atteignons.

Dans l'exemple qui suit, nous utilisons une vérification d'API Splunk pour :

1. faire une requête avec une clé d'API au point de terminaison de l'API Splunk pour obtenir de vraies données de vérification du navigateur ;
2. confirmer que le code de réponse contient la valeur « 200 » ;
3. extraire l'ID de vérification du JSON à l'aide du chemin JSON.



4. vérifier que l'ID de vérification extrait du chemin JSON est la valeur attendue. Ce flux utilisateur très simple nous permet de tester :

- **la disponibilité.** La vérification échouera si l'API renvoie un code de réponse qui n'est pas 200 OK ;
- **le format des données.** Si les données reviennent de l'API dans un format autre que JSON, l'étape d'extraction d'une valeur à l'aide du chemin JSON échouera à la vérification ;
- **la qualité des données.** Si nous parvenons à extraire une valeur pour l'ID mais qu'elle ne correspond pas à la valeur attendue, l'étape d'assertion échouera à la vérification.

Par exemple, si nous recevons une alerte indiquant que notre moniteur externe n'a pas pu extraire l'ID de vérification du JSON, nous allons analyser notre alerte et inspecter le corps du résultat ou de la réponse à partir du point de terminaison de l'API. En examinant le corps de la réponse, nous pouvons rapidement voir si le format était incorrect ou si la valeur id manquait dans la sortie. Ces informations nous aident à commencer le dépannage immédiatement.

Ceci n'est qu'un exemple simple de la mise en œuvre d'une supervision robuste pour une API. Si vos tests API actuels ne supervisent que le code et le temps de réponse, il est peut-être temps d'ajouter des critères supplémentaires pour le format et la qualité des données.

Écrire des tests de performance en envisageant l'échec

Jusqu'à maintenant, nous avons abordé la supervision axée sur les en-têtes de requête, l'authentification et la validation des données. Lorsqu'il s'agit de rédiger des tests de performances, une stratégie consiste à écrire des tests de manière à ce que le système appelle une API et ne reçoive pas de données.

Lorsque l'on rédige du code contenant de nombreux appels locaux, un wrapper qui appelle une API externe passe souvent inaperçu dans le contexte d'une application. Si votre test est conçu pour vous alerter lorsqu'aucune donnée n'est présente, vous vous assurez de ne pas manquer d'erreurs critiques. N'oubliez pas de rendre votre code résilient pour qu'il continue de fonctionner lorsqu'il reçoit un message d'erreur, des données falsifiées ou aucune réponse.

CHAPITRE 4

Supervision des SLA

Plus tôt, nous avons évoqué les vulnérabilités qui découlent des systèmes interdépendants qui doivent transmettre des données dans les deux sens sur le Web. De plus en plus d'applications sont structurées sur des API tierces, il est donc essentiel pour les entreprises que les API partenaires fonctionnent de manière rapide, fiable et sûre.

Les entreprises ont développé un moyen de gérer ce risque, et nous appelons généralement ces contrats des « SLA ».

Qu'est-ce qu'un SLA ?

Un accord de niveau de service (communément appelé SLA) est un accord entre deux parties sur les services qui seront fournis par une partie à l'autre. Au sens large, cet accord peut inclure n'importe quel type de services, du délai de réponse de l'assistance personnalisée à la livraison des produits.

Souvent, lorsque deux fournisseurs de technologies ou de logiciels établissent un SLA, l'accord décrit à la fois :

- **la disponibilité.** Quel taux de disponibilité peut être garanti par le partenaire ? Combien de temps à l'avance le partenaire doit-il être informé d'une coupure ou d'une maintenance planifiée ?
- **la réactivité.** Dans quels délais mon système peut-il attendre une réponse de la part du système du partenaire ?

L'une des parties peut prétendre à un crédit, à un remboursement ou à une résiliation anticipée selon que ces SLA sont respectés ou non.

Par exemple, imaginons une application Web de covoiturage qui s'appuie fortement sur les données d'un tiers spécialisé dans la cartographie. Lorsque cette application web de covoiturage accepte de travailler exclusivement avec un excellent fournisseur de cartographie, ce fournisseur de cartographie peut garantir : « Votre application de covoiturage aura accès à nos données cartographiques 99,99 % du temps et nous informerons votre équipe au moins trois semaines avant la prochaine maintenance planifiée ».

Et l'équipe qui gère l'application de covoiturage pourrait répondre : « Parfait ! Ça nous semble très honnête. Nos utilisateurs tolèrent certains problèmes et ils ne s'en plaignent jamais sur Twitter, donc 99,99 % de disponibilité est un taux largement suffisant. Et trois semaines nous laissent amplement le temps d'informer nos clients des temps d'arrêt à venir. Nous acceptons les conditions, mais si votre API est disponible moins de 99,99 % du temps, nous devrons être intégralement remboursés. »

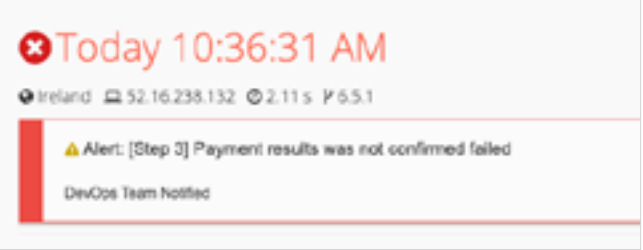
Tout le monde signe sur les lignes pointillées et se serre la main. L'application de covoiturage intègre une nouvelle fonctionnalité qui se lie à l'API pour extraire les données du fournisseur de cartographie.

Comment vous assurer que vous respectez vos SLA

En tant que responsables pour le fournisseur de cartographie, nous pourrions dire : « Nous avons besoin de données pour anticiper les problèmes afin d'avoir les moyens de respecter notre part du contrat. Et idéalement, nous voudrions partager ces données publiquement avec nos partenaires sur l'application de covoiturage afin qu'ils sachent qu'ils peuvent faire confiance à notre service ».

Nous pourrions compter sur le contrôle interne de notre application, mais le tableau serait incomplet. Comment savoir si nos données cartographiques sont disponibles à partir de notre API pour un utilisateur final extérieur à notre système ? Comment confirmer non seulement que les données sont disponibles, mais également dans le bon format ?

Nous pouvons créer un moniteur externe synthétique pour tester l'extraction des données de notre propre API et mettre en place des alertes pour que nos ingénieurs soient immédiatement avertis en cas de problème susceptible de nous faire enfreindre notre accord.



L'exemple ci-dessus montre une alerte émise par une vérification d'API dans Splunk Synthetic Monitoring.

Grâce aux données proactives qui simulent de vrais utilisateurs finaux interagissant avec notre système de cartographie, nous pouvons :

1. anticiper les problèmes de performances avant qu'ils n'affectent nos vrais utilisateurs, et
2. transmettre des rapports à notre partenaire pour démontrer que notre disponibilité est parfaitement conforme à notre engagement.

Comment faire appliquer les SLA par vos partenaires

En tant que responsables de l'application Web de covoiturage, nous pourrions dire : « Merci pour tous ces rapports, amis fournisseurs de cartographie, mais nous devons vraiment faire preuve de rigueur et comparer certaines données externes à vos rapports ». Nous pourrions utiliser des contrôles d'API externes pour superviser les performances de l'API de cartographie et confirmer qu'elle est bien opérationnelle 99,99 % du temps.

Si nous constatons que la disponibilité devient inférieure au SLA ou si nous observons un temps d'arrêt prolongé qui n'a pas été communiqué trois semaines à l'avance, nous pourrions utiliser nos rapports comme base pour discuter avec notre partenaire de la rectification de la violation du contrat. Nous pouvons également utiliser des vérifications d'API pour mieux comprendre comment les problèmes touchant l'API de notre partenaire peuvent affecter nos utilisateurs réels.

Les contrôles d'API peuvent être utilisés par les deux parties d'un accord de niveau de service pour confirmer le respect de ses conditions. Pour les responsables d'API, la supervision proactive peut aider à détecter les problèmes avant qu'ils n'affectent les partenaires et permet de produire des rapports faciles à partager. Pour les utilisateurs finaux de l'API, la supervision proactive peut alerter des problèmes externe affectant les utilisateurs et offrir un niveau supplémentaire de confiance dans la capacité des partenaires à respecter leur contrat.

Dans l'exemple qui suit, une vérification en plusieurs étapes portant sur l'API du fournisseur de cartographie confirme que les données se situent dans la plage attendue et suit la disponibilité de ces demandes.

N'oubliez pas : les contrôles d'API peuvent être utilisés pour superviser à la fois la disponibilité et la réactivité. Tous les scénarios d'utilisation que nous élaborons pour les vérifications d'API doivent correspondre à nos SLA. Si notre accord est basé uniquement sur la disponibilité, il suffit de configurer une vérification qui va simplement atteindre le point final, puis de se fier au taux de disponibilité. Si notre accord est basé sur la vitesse à laquelle une API renvoie des données, nous pouvons créer une vérification en plusieurs étapes qui extrait des données de l'API, puis comparer le temps de réponse moyen aux normes convenues.

CHAPITRE 5

Les contrôles d'API de Splunk

Chez Splunk, nous avons développé un cadre de tests de performances qui peut aider les entreprises à évaluer si les API fonctionnent comme prévu. Avec les vérifications d'API, nous pouvons :

- suivre la disponibilité des API critiques ;
- capturer et analyser le temps de réponse d'une API ;
- assurer la fonctionnalité de l'API en validant les valeurs et la structure de sa réponse ;
- produire des alertes sur toutes les conditions indiquant une API cassée ou peu performante, pour permettre aux équipes d'anticiper les problèmes avant qu'ils n'affectent les utilisateurs ou ne violent les SLA.

Les vérifications d'API de Splunk reposent sur quatre concepts simples :

1. faire des requêtes HTTP ;
2. extraire les données des réponses ;
3. enregistrer les données à utiliser dans des requêtes ou des analyses supplémentaires ;
4. faire des assertions sur les données et leur format.

Ces composants sont simples, mais puissants. Comme une fractale, des choses parfois belles et compliquées peuvent émerger de l'organisation des blocs de construction les plus élémentaires. La vérification d'API de Splunk utilise des éléments simples qui peuvent être assemblés pour produire des tests capables de superviser et de vérifier des flux d'API d'une grande complexité.

Les ingénieurs ont travaillé en étroite collaboration avec les clients pendant le développement et le test de la vérification d'API pour s'assurer que cette approche simple facilite la vérification tout en couvrant les besoins des clients. Lookout, un fournisseur leader de solutions de sécurité mobile, faisait partie de nos bêta-testeurs. Voici ce que Dean Ross-Smith, Ingénieur des opérations cloud chez Lookout, peut nous en dire :

« La flexibilité de la nouvelle vérification d'API de Splunk a permis à Lookout de configurer un contrôle et de superviser un élément critique de l'infrastructure. Jusqu'ici, aucune autre solution ne nous permettait de le faire. »

Conçu pour les entreprises

Qu'une API alimente un site web, une application mobile ou même l'infrastructure centrale de votre entreprise, la vérification d'API de Splunk garantit la performance, la disponibilité et la fonctionnalité de l'API par des moyens qui sont à la portée d'utilisateurs de tout niveau technique.

Notre vérification d'API répond notamment aux besoins métier suivants :

- tester des flux d'API complexes en plusieurs étapes ;
- superviser la disponibilité et le temps de réponse à partir de différentes régions du monde ;
- suivre et appliquer les SLA de performance des API tierces ;
- vérifier l'exactitude des réponses de l'API ;
- tester l'intégralité du cycle de vie CRUD d'un objet de données via une API ;
- gérer des systèmes d'authentification API complexes basés sur des jetons ;
- superviser des pages d'état des applications.

Points à retenir

Vous avez une application qui dépend d'API internes ou tierces ? Vous fournissez des données à vos clients via une API ? Vous avez besoin d'une grande transparence sur la disponibilité, la fonctionnalité et les performances d'une API ? La réponse à toutes ces questions est la nouvelle vérification d'API pour Splunk Synthetic Monitoring. Pour en savoir plus sur les produits de performance Web de Splunk ou pour bénéficier d'une démonstration, contactez-nous dès aujourd'hui.

Pour obtenir toutes les informations et un guide technique détaillé, consultez API Check sur **Splunkbase.**

splunk>

En savoir plus : www.splunk.com/asksales

www.splunk.com

21-16715-SPLK-APIs-In-Action-TWP-106