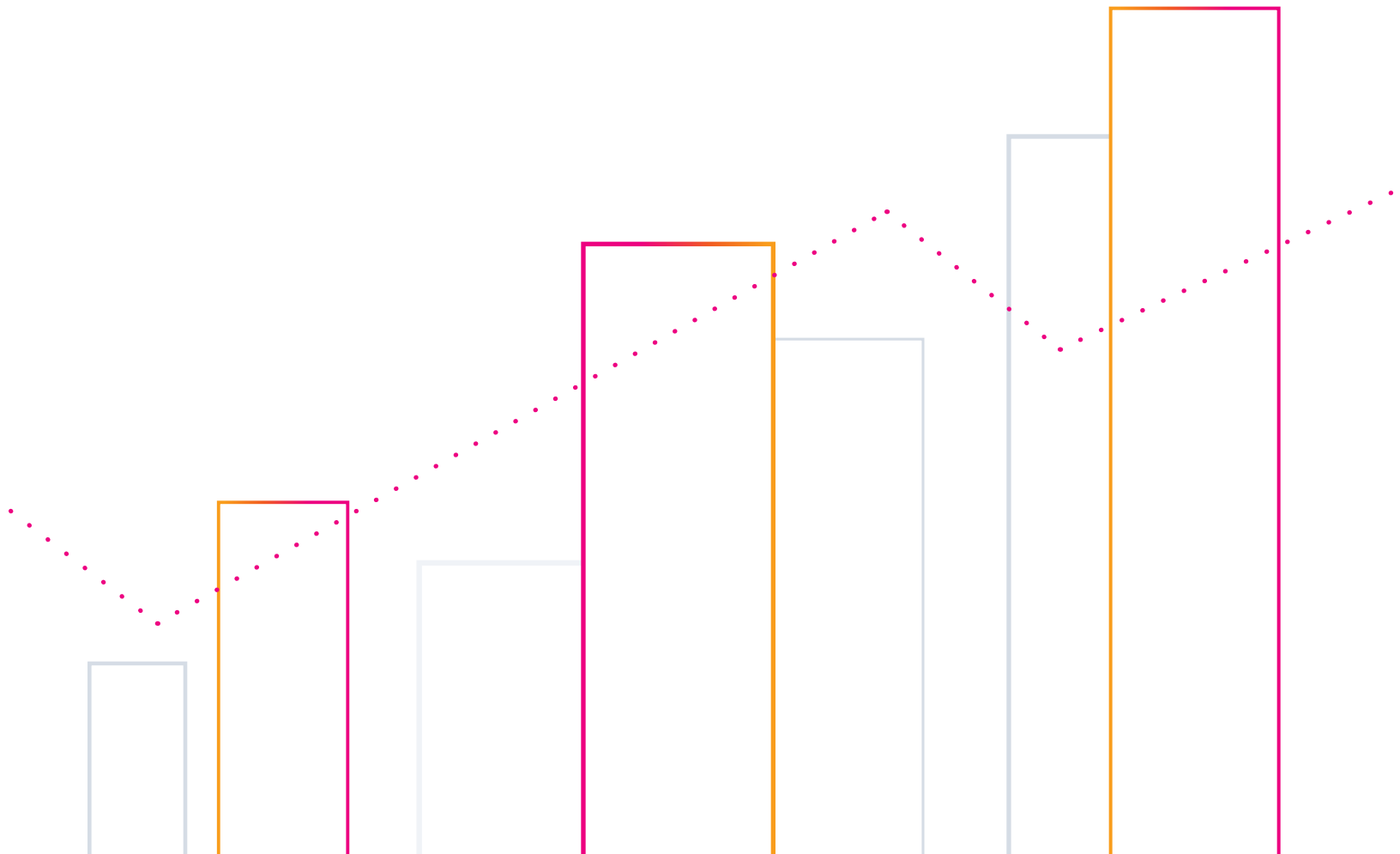
splunk> turn data into doing™

WHITEPAPER

APIs in Aktion

Ein Leitfaden zum Monitoring der API-Performance



Inhalt

Einleitung

Der Siegeszug von Microservices, die Verbreitung von Single-Page-Web-Anwendungen und die ungebrochene Dominanz von nativen Mobil-Apps haben APIs zu den Dreh- und Angelpunkten im modernen Web gemacht.

APIs waren auch früher schon eine Möglichkeit für Unternehmen, sich von den Mitbewerbern abzusetzen und Integrationsmöglichkeiten mit anderen Systemen zu bieten. Es gibt aber auch die Meinung, dass APIs sich erst mit dem Aufkommen der sozialen Medien allgemein durchgesetzt haben: Mit der zunehmenden Größe und Benutzerzahl der sozialen Netzwerke begannen die Unternehmen, APIs zu entwickeln und zu nutzen, sodass Anwender Inhalte teilen und in die Datenströme der Social-Media-Interaktionen einbetten können.

Heute nutzen wir APIs als Verbindungen zwischen mehreren Social-Media-Plattformen, damit wir beispielsweise Nachrichten aus Instagram an Facebook oder Twitter posten können. APIs werden außerdem für Verbindungen zu Datenbeständen verwendet, wenn wir etwa unseren Standort auf einer Karte markieren, damit wir einchecken oder eine Mitfahrgelegenheit finden können. Viele Web- und Mobil-Apps, die wir heute nutzen, basieren auf externen APIs auf und sind darauf angewiesen, dass diese APIs ordnungsgemäß funktionieren und Daten korrekt, schnell, sicher und zuverlässig liefern.

Da APIs ein derart wichtiger Faktor moderner Anwendungen und Infrastrukturen sind, ist das Monitoring der API-Performance unbedingt notwendig.

In diesem Whitepaper erläutern wir die Grundfunktionen von APIs und erklären, warum API-Monitoring erfolgsentscheidend ist und auf welche Funktionen Sie beim Implementieren von Systemen zum Monitoring der API-Performance Wert legen sollten.

Grundlegendes zu APIs

Bei einer API handelt es sich um eine Gruppe von Programmieranweisungen und -standards für den Zugriff auf eine webbasierte Software-App oder einen Service. APIs enthalten Anweisungen, mit denen Entwickler die Interaktion mit den Services herstellen und Daten zwischen den einzelnen Systemen austauschen können. Eine API beschreibt die verfügbare Funktionalität eines Services, ihre Verwendung, den Zugriff darauf und die Formate für Ein- und Ausgaben.

Heutzutage basieren ganze Unternehmen und Anwendungen auf offenen APIs und deren Fähigkeit, Daten zwischen Systemen zu übertragen. Sie können sich eine API wie einen Kellner im Restaurant vorstellen. Sie nehmen Platz und bekommen eine Speisekarte, aus der Sie Getränke und Gerichte bestellen können. Vielleicht haben Sie besondere Wünsche, was die Zubereitung der Speisen angeht. Wenn der Kellner nach Ihrer Bestellung fragt, sagen Sie z. B.: „Ich hätte gerne das Filetsteak, medium gebraten, mit Rahmspinat und einer Ofenkartoffel mit Sour Cream, aber ohne Schnittlauch."

Der Kellner notiert Ihre Bestellung, gibt sie an die Küche weiter, holt Ihr Essen dort ab, wenn es fertig ist, und bringt es an den Tisch.

Bei diesem simplen Szenario übernimmt der Kellner die Rolle einer API, natürlich mit dem Unterschied, dass die API kein Essen, sondern Daten liefert. Wie ein Kellner nimmt die API eine Reihe von Anweisungen (eine Anforderung) von einer Quelle (von einer Anwendung oder von Entwicklern) entgegen, übermittelt diese Anforderung an eine Datenbank, holt die Daten ab oder erleichtert bestimmte Aktionen und gibt dann eine Antwort an die Quelle zurück. APIs sind Boten, die Systeme miteinander verbinden.

Im Zusammenhang mit dem Senden und Empfangen von API-Nachrichten sollte man wissen, dass es zwei Arten von APIs gibt: private Programmierer-APIs, die intern innerhalb einer Organisation verwendet werden, und öffentliche APIs, die für Anwender zugänglich sind.

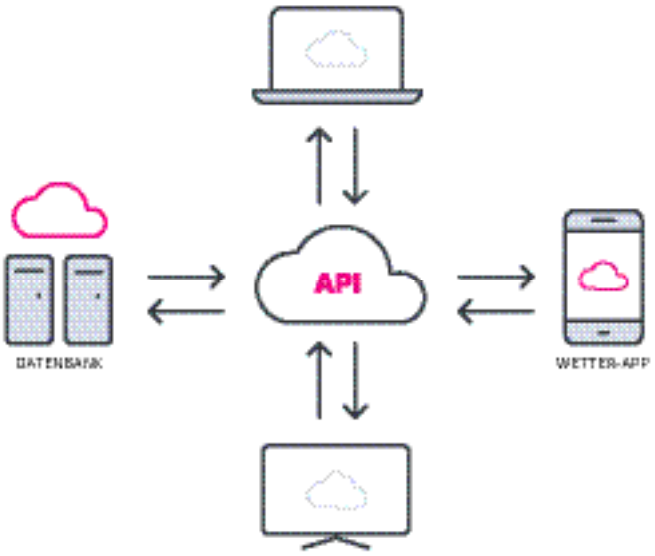
Private APIs machen Unternehmen agiler, flexibler und leistungsfähiger. Öffentliche APIs machen es Unternehmen einfacher, Verbindungen herzustellen, um neue Integrationen anzubieten und neue Partnerschaften aufzubauen.

APIs in Aktion

Sehen wir uns ein bekanntes Unternehmen an, das öffentliche APIs einsetzt.

The Weather Company sammelt Wetterdaten von Millionen von Endpunkten und Quellen rund um den Erdball. Das Unternehmen speichert diese Daten und macht sie über Hunderte unterschiedlicher APIs für Endverbraucher-orientierte Anwendungen zugänglich.

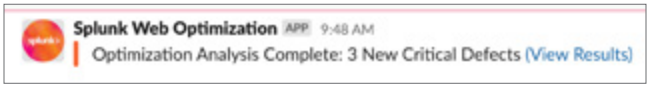
Nehmen wir einmal an, Sie haben ein Widget auf Ihrem Startbildschirm oder Ihrem Desktop, das immer die aktuelle Temperatur und das aktuelle Wetter an Ihrem Standort zeigt. Sie werden in Ihrem Gerät wohl kaum ein Thermometer, Barometer oder sonstige Technologie zur Erkennung und Vorhersage von Wettermustern eingebaut haben, doch es kann Informationen über Ihren Standort an eine API senden und auf diese Weise die passenden Daten ausgeben: die Temperatur, die örtliche Wettervorhersage etc. Selbst wenn Sie nicht auf der Weather-Company-Website sind oder eine der nativen Mobil-Apps des Unternehmens nutzen, interagieren Sie unter Umständen mit Daten aus einer Weather-Company-API, die einfache Anforderungen beantwortet.



Vergleichen wir das Szenario dieser öffentlichen API nun mit einem Beispiel dafür, wie APIs zum Auslösen von Aufgaben eingesetzt werden können, um einen internen Geschäftsprozess zu verbessern.

Splunk Web Optimization erfasst eine Reihe von Performance-Metriken für digitale Unternehmen und gibt intelligente Empfehlungen zur Verbesserung der Site-Performance. Mit diesem Optimierungstool können Teams leicht erkennen, ob Änderungen an einer Website zu einer Verschlechterung der Leistung führen, ob also eine Site oder Web-App etwas enthält, das zur Folge hat, dass sie langsamer geladen wird.

Mit der API des Optimierungstools kann das Splunk-Team solche Performance-Tests in Bereitstellungen integrieren und so etwaige Performance-Verschlechterungen identifizieren. Es wurde eine Anforderung in den Bereitstellungsprozess der Optimierungs-API eingefügt, die einen Performance-Scan in der Staging-Umgebung von Splunk auslöst und die Scan-Ergebnisse in unserem Chat-Kanal in Slack postet. Jetzt können wir unsere bestehende ChatOps-Einrichtung verwenden, um das Team zu informieren, falls neue Elemente unsere Anwendung in der Staging-Umgebung langsamer machen.



diesem Szenario gibt die API nicht nur Daten auf eine forderung zurück, sondern führt zudem anforderungs-hängig Aktionen aus. Sowohl die Weather-Company-API auch die Splunk-API könnten öffentlich bzw. End-braucher-orientiert sein, sodass also auch andere steme diese APIs dafür nutzen könnten, Daten einzu-dern oder bestimmte Aktionen auszulösen.

KAPITEL 1

SOAP, REST und JSON

Im Kern geht es bei APIs darum, Daten von einem Remote-Service anzufordern und zu empfangen. In diesem E-Book liegt der Schwerpunkt auf webbasierten APIs, bei denen Anforderung und Antwort über HTTP erfolgen. Es gibt viele unterschiedliche Möglichkeiten, diese Anforderungen und Antworten zu formatieren. Wer sich in API-Beschreibungen einliest, wird dabei auf unzählige verschiedene Abkürzungen und Bezeichnungen stoßen. Im Wesentlichen spielen sie aber im Hinblick auf das Gesamtkonzept von APIs, ihre Bedeutung und die Vorgehensweisen bei Performance-Monitoring und -Tests keine Rolle.

Hier nur die wichtigsten Punkte zur Orientierung:

- Einzelne API-Formate verwenden eventuell unterschiedliche Methoden zur Strukturierung von Anforderungs- und Antworttext; SOAP ist sehr strukturiert, hier wird XML verwendet. REST ist sehr flexibel, verwendet aber meist JSON.
- Bei den verschiedenen API-Formaten werden eventuell unterschiedliche HTTP-Methoden für Anforderungen verwendet. Bei Formaten wie SOAP wird möglicherweise immer POST verwendet, während für REST vor allem GET und POST oder auch weniger gängige Methoden wie PUT oder DELETE verwendet werden.
- Anmelde- bzw. Authentifizierungsdaten werden je nach Format eventuell unterschiedlich übermittelt. Es können beispielsweise Cookies oder spezielle HTTP-Header verwendet werden oder sogar Parameter der Abfragezeichenfolge.

Wenn Sie das Gefühl haben, dass Sie den Überblick verlieren, denken Sie einfach daran: Bei APIs geht es darum, eine Anforderung zu senden und Daten zurückzubekommen. Alles andere baut auf diesem einfachen Prinzip auf.

KAPITEL 2

Gründe für das API-Monitoring

Wir haben bereits einige Beispiele für Systeme kennengelernt, die von API-Daten abhängen oder wichtige Funktionen haben, die auf APIs von Drittanbietern aufsetzen.

APIs sind extrem praktisch, da sie Services verbinden und es uns ermöglichen, Daten zwischen individuell verwalteten Systemen auszutauschen. Leider entstehen aber durch all diese Verbindungsmöglichkeiten und gegenseitigen Abhängigkeiten auch Schwachstellen. Im Hinblick auf Performance und Verfügbarkeit ist das Monitoring der APIs, auf die wir selbst angewiesen sind, und der APIs, die wir für andere bereitstellen, von enormer Bedeutung.

Bedeutung der API-Performance

Wenn eine API ausfällt und die Leistung oder User Experience Ihrer Website beeinträchtigt, fällt dies auf Ihr Unternehmen zurück. Endanwendern und Kunden ist höchstwahrscheinlich nicht klar, dass die Ursache bei einem Drittanbieter liegt. Und falls diese API obendrein für einen Ihrer Transaktionsprozesse notwendig ist, kann sich der Ausfall sogar unmittelbar auf Ihr Geschäftsergebnis auswirken.

Nehmen wir an, dass eine Schlüsselkomponente des Bezahlvorgangs auf Ihrer Website eine Standortsuche ist, für die Sie eine Drittanbieter-API nutzen. In diesem Fall können potenzielle Kunden ihren Einkauf gar nicht erfolgreich abschließen, wenn diese API nicht richtig funktioniert.

Oder stellen Sie sich vor, Sie hätten eine Anwendung entwickelt, die die Authentifizierung durch eine Social-Media-Plattform erfordert. Fällt die API zu dieser Social-Media-Plattform aus, können sich Ihre Benutzer nicht mehr bei Ihrem System anmelden.

Als Entwickler oder Website-Inhaber entscheiden Sie möglicherweise, dass die Vorteile des Drittanbieter-Services das Ausfallrisiko überwiegen. Damit Sie aber das Risiko genau einschätzen und sich ein Bild davon machen können, wie sich der Service im zeitlichen Verlauf auswirkt, sollten Sie den betreffenden Benutzer-Flow auf Ihrer Website unbedingt überwachen.

Wenn Sie eine offene API für Partner oder Entwickler zur Verfügung stellen, liegt es in Ihrer Verantwortung, sicherzustellen, dass die API verfügbar ist und ordnungsgemäß funktioniert.

Noch ein Szenario: Nehmen wir an, Sie haben eine neue interne API entwickelt, die Bestelldaten von einem Mobilgerät an das System in Ihrem Produktlager übermittelt. Vielleicht müssen die Daten unbedingt innerhalb von zwei Minuten übertragen werden, da sonst der gesamte Produktionszeitplan hinfällig ist. Beim Entwickeln und Testen der API in der Staging-Umgebung sind die Daten immer erfolgreich innerhalb einer Minute übertragen worden. Doch sobald Sie die API nach dem Produktivstart zur Verarbeitung realer Anforderungen einsetzen, müssen Sie feststellen, dass die tatsächliche Antwortzeit sich immer mehr der Zweiminutenmarke nähert. Ohne aktives Monitoring der API in der Produktion würde Ihr Team aufgrund der Vorproduktionstests wohl einfach annehmen, dass die neue API schon schnell genug ist.

Hinweis: Wenn Sie eine API hosten, auf die andere Personen angewiesen sind, sollten Sie diese API unbedingt sowohl in Vorproduktions- als auch Produktionsumgebungen aktiv überwachen.

Ganz gleich, ob Sie eigene APIs im Blick behalten oder die Auswirkungen externer, für Sie notwendiger Services kontrollieren möchten – das Monitoring von APIs auf Verfügbarkeit, Funktionalität, Geschwindigkeit und Performance ist enorm wichtig. Falls Sie bereits wissen, dass Sie eine API verwenden, die sich in der Vergangenheit als nicht stabil erwiesen hat, und dass diese API nicht aktiv überwacht wird, trommeln Sie am besten noch heute Ihr Team zusammen und entwickeln gemeinsam einen Plan für das Monitoring dieser API.

Was sollte überwacht werden?

Sie verstehen jetzt, warum das Monitoring der API-Performance so wichtig ist. Und Sie wissen auch, dass Sie beide API-Arten berücksichtigen müssen:

- APIs, auf die Ihre Website oder native Anwendung für kritische Daten oder Prozesse angewiesen ist, und
- von Ihnen verwaltete APIs, die Kunden, Endanwender oder Entwickler für Daten oder Prozesse benötigen.

Beim Monitoring dieser beiden API-Arten sollten Sie unbedingt folgende Kriterien kontrollieren:

- **Verfügbarkeit:** Ist dieser API-Endpunkt verfügbar? Meldet er einen Fehler?
- **Antwortzeit:** Wie schnell reagiert die API? Verschlechtert sich die Antwortzeit mit der Zeit? Ist die Antwortzeit im Produktionssystem länger als in der Vorproduktion?
- **Datenvalidierung:** Gibt die API die richtigen Daten im richtigen Format zurück?
- **Mehrstufige Prozesse:** Lässt sich eine Variable aus dieser API erfolgreich speichern und wiederverwenden? Funktioniert die Authentifizierung wie erwartet? Kann ich eine Transaktion mit Daten von dieser API abschließen?

Dies sind nur die grundlegenden Fragestellungen, die Ihr Team beim Monitoring der API-Performance prüfen sollte. Im folgenden Abschnitt befassen wir uns damit, wie API-Monitoring technisch umgesetzt wird und welche Funktionen für Performance-Tests wichtig sind, die ebenso solide wie flexibel sind.

KAPITEL 3

Vorgehensweise beim API-Monitoring

Wenn Sie Zugriff auf ein externes, proaktives Monitoring-System haben, lässt sich eine API-Antwort durch einfache Uptime- oder Ping-Tests relativ leicht auf Verfügbarkeit prüfen. Definieren Sie das Protokoll oder geben Sie einen Endpunkt an, legen Sie fest, dass der Test wiederholt von Standorten rund um den Globus ausgeführt werden soll, und lassen Sie sich bei Antwortfehlercodes oder Latenzen automatisch benachrichtigen.

Was aber, wenn Sie mehr als die bloße Verfügbarkeit kontrollieren wollen?

Es gibt eine Reihe wichtiger Funktionen, die Ihre Monitoring-Lösung mitbringen sollte, damit Sie API-Transaktionen in vollem Umfang testen können. Die folgenden Komponenten gehören zu typischen API-Anfragen, die Sie in Ihren Tests konfigurieren müssen – jenseits der Frage „Welchen Endpunkt soll ich ansprechen?“.

Anforderungsheader

Je nach der Funktionsweise einer Website bzw. Anwendung sind Request Header unter Umständen ein wichtiger Bestandteil, der in die Konfiguration der Monitoring-Tests gehört. Falls wir etwa aktiv testen möchten, wie ein Bezahlvorgang bei einem Besucher mit Cookie funktioniert, dann brauchen wir eine Möglichkeit, dieses Cookie mit einem Anforderungsheader festzulegen, wenn wir einen aktiven Test für diese Transaktion erstellen.

Wenn wir hier von Anforderungsheadern sprechen, meinen wir Felder, die in den Header-Abschnitten von HTTP-Anforderungen übertragen werden. Anforderungsheader können Regeln und Einstellungen enthalten, mit denen definiert wird, wie eine HTTP-Transaktion ablaufen soll.

Es gibt einen Standardsatz an unterstützten Request-Header-Typen mit eigenen Namen und festgelegten Zwecken. Gängige Beispiele von Anforderungsheadern sind:

- **Authorization:** Übertrage Anmeldedaten für die HTTP-Basisauthentifizierung als Zugriffsberechtigung.
- **Cache-Control:** Informiere den Browser, wie lange eine Ressource im Cache gespeichert und wiederverwendet werden soll.
- **Content-Type:** Übertrage den MIME-Typ des Hauptteils einer Anforderung an den Server, damit dieser weiß, wie die Daten geparst werden sollen.
- **Cookie:** Lege ein im Browser zu speicherndes Cookie fest, damit Zustände oder Sessions nachverfolgt werden können.

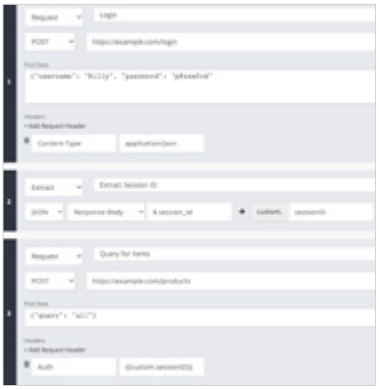
Manche Entwickler implementieren auch benutzerdefinierte Anforderungsheader mit individuell festgelegten Bezeichnungen. Es ist gängige Praxis, solchen Request Header ein X als Präfix voranzustellen. Der Anforderungsheader X-Http-Method-Override könnte beispielsweise dazu dienen, die Anforderungsmethode zu ändern, z. B. POST in PUT oder DELETE.

Anforderungsheader in API-Prüfungen

Mit dem API Check von Splunk können wir Verfügbarkeit, Antwortzeit und Datenqualität bei API-Transaktionen überwachen. Bei einer API-Prüfung können wir bei jeder Anforderung Request Header als Teil der Transaktion festlegen.

Stellen Sie sich ein Szenario vor, bei dem wir Benutzername und Passwort mit POST übermitteln müssen, um Zugriff auf bestimmte Informationen zu bekommen. Nach der Anmeldung beim Endpunkt müssen wir eine Session-ID speichern und festlegen, um andere, sessionspezifische Komponenten vorab einzutragen.

Beim Erstellen einer API-Prüfung können wir einfach auf „+ Add a Request Header“ klicken und bei jedem Anforderungsschritt einen oder mehrere Header angeben.



Im obigen Beispiel verwenden wir die API-Prüfung von Splunk für Folgendes:

1. Absetzen einer Anforderung, um zwecks Anmeldung Benutzername und Kennwort per POST an einen Endpunkt zu übertragen.
2. Extraktion einer Session-ID aus dem Antworttext mit „JSON path“; diese ID wird als Variable gespeichert, die wir bei späteren Schritten gebrauchen können.
3. Absetzen einer Anforderung für die POST-Übertragung an einen anderen Endpunkt, wobei die Session-ID in den Anforderungsheadern übertragen wird.

Wir können jetzt weitere funktionale Schritte zu dieser Transaktion hinzufügen oder einen „Assert"-Schritt verwenden, um zu bestätigen, dass die Session-ID wie erwartet festgelegt wurde.

Authentifizierung

Im obigen Beispiel haben wir Anforderungsheader für die Übertragung von Benutzername und Passwort verwendet. Jetzt sehen wir uns APIs einmal unter Sicherheitsaspekten an und überlegen, wie wir diese bei unseren Performance-Tests berücksichtigen können.

Durch die Authentifizierung für eine API wird festgelegt, wer auf sichere Daten oder Endpunkte zugreifen darf. Dies ist besonders wichtig bei APIs, die Zugriff auf sensible Informationen geben, bei APIs, die Endanwendern Änderungen erlauben, sowie für Unternehmen, die für den Zugriff auf Daten über eine API Gebühren erheben. Nun ist es eine Aufgabe, eine API für einen einzelnen menschlichen User zu sichern, eine ganz andere, die Authentifizierung bei Systemen für eine wachsende Anzahl nichtmenschlicher Entitäten zu regeln. Dazu müssen wir weitere Überlegungen anstellen.

APIs werden immer sicherer, und proaktive Monitoring-Systeme halten bei dieser Entwicklung Schritt, sodass sie den externen Zugriff auf sichere Systeme ermöglichen können.

Direkte Authentifizierung

Ein gutes Beispiel für die direkte Authentifizierung ist die HTTP-Basisauthentifizierung. Sie ist ein standardmäßiger HTTP-Bestandteil, der für API-Endpunkte oder HTTP-URLs verwendet werden kann. Man überträgt bei der Anforderung an die API einfach einen Benutzernamen und ein Passwort, die gemeinsam mit Base64 verschlüsselt werden. Die HTTP-Basisauthentifizierung hat den Vorteil, dass sie sich leicht implementieren lässt und bei Standard-Frameworks meist enthalten ist. Der Nachteil ist, dass die HTTP-Basisauthentifizierung keine erweiterten Optionen bietet und leicht entschlüsselt werden kann.

Ein weiteres Beispiel direkter Authentifizierung wären API-Schlüssel oder -Token. Ein API-Schlüssel ist einfach eine lange hexadezimale Zeichenfolge wie z. B. 34d83d84f28d146aeae0e32f7803c88d, die anstelle von Benutzername oder Passwort übermittelt werden, um den Zugriff zu authentifizieren. API-Schlüssel sind im Grunde dasselbe wie Anmeldedaten aus Benutzername und Passwort, sie bieten jedoch eine nützliche Abstraktionsebene. So könnten z. B. mehrere Endanwender einen gemeinsamen API-Schlüssel nutzen.

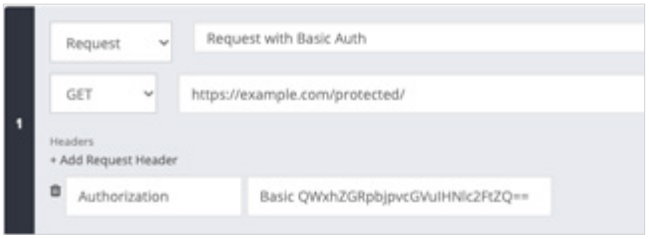
Bei jeder Art von direkter Authentifizierung ist es wichtig, dass Sie SSL/TLS verwenden oder am Anfang der URL für den API-Endpunkt https:// angeben. Durch die Verwendung von SSL/TLS wird sichergestellt, dass die Anmeldeinformationen bzw. API-Schlüssel für die HTTP-Basisauthentifizierung in der URL nicht offengelegt werden.

Möchten Sie mehr über SSL/TLS und Performance-Optimierung erfahren? Im Zoompf-Blog gibt es dazu **eine Reihe exzellenter Artikel**.

HTTP-Basisauthentifizierung

Wenn Sie die Basisauthentifizierung zur Sicherung Ihrer APIs verwenden, lässt sich diese Authentifizierung beim Konfigurieren einer externen Monitoring-Lösung zum Monitoring der API-Performance ganz leicht einbinden.

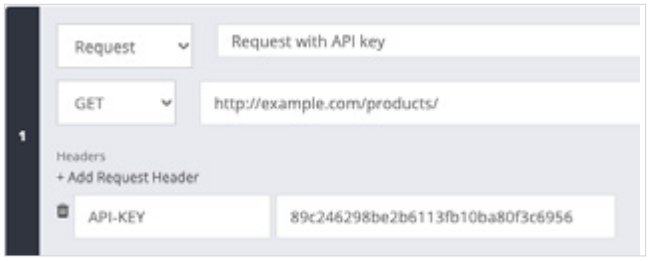
Die gängigste und zuverlässigste Methode der Erstellung einer Monitoring-Anforderung mit HTTP-Basisauthentifizierung besteht darin, den Wert aus Benutzername:Passwort in Base64 zu codieren und in einem Authorization-Header zu übertragen:



Dabei dürfen Sie nicht vergessen, dass sich Benutzernamen und Passwörter in Base64 nicht nur leicht codieren, sondern umgekehrt ebenso leicht decodieren lassen, damit das System eine Anforderung authentifizieren kann. Sie können dies mit einem **Online-Codierer/Decoder für Base64** selbst ausprobieren. Da der Base64-Zugriff so einfach ist, muss diese Art der direkten Authentifizierung unbedingt mit SSL/TLS geschützt werden.

API-Schlüssel

Aus Monitoring-Perspektive ist es ziemlich einfach, das Ansprechen eines Endpunkts mit einem API-Schlüssel in der URL oder mit Anforderungsheadern zu replizieren. Sie geben den Schlüssel an und behalten im Hinterkopf, dass Sie auch die Konfiguration Ihres Monitoring-Tests aktualisieren müssen, falls sich der Schlüssel einmal ändert.



Bitte beachten: Unterschiedliche Systeme akzeptieren API-Schlüssel eventuell auf unterschiedliche Weise, z. B. als Teil der POST-Daten anstatt als Anforderungsheader. Sie sollten dies also für die zu überwachende API prüfen, damit Sie den API-Schlüssel richtig übertragen.

Ticketbasierte Authentifizierung

Die Implementierung der direkten Authentifizierung bietet zwar zweifellos einige Vorteile, doch wir sollten eventuell eine zusätzliche Sicherheitsebene zu unseren APIs hinzufügen. Ticketbasierte Authentifizierungssysteme nutzen zentrale Authentifizierungsserver, die als Vermittler agieren, die Anmeldeinformationen von Endanwendern entgegennehmen und dann Tickets, Token oder Schlüssel zurücksenden, die dem jeweiligen Nutzer den Zugriff auf bestimmte geschützte Daten ermöglichen. Die ticketbasierte Authentifizierung eignet sich ideal für Fälle, in denen Sie sensible Informationen schützen müssen, einer API das Erstellen von Objekten oder das Durchführen von Änderungen gestatten wollen oder Gebühren für die API-Nutzung erheben.

Funktionsweise ticketbasierter Systeme

Sie können sich die ticketbasierte Authentifizierung so ähnlich wie das Prozedere vorstellen, das Sie durchlaufen, um den Schlüssel für eine Probefahrt mit einem Auto zu bekommen.

Nehmen wir an, ich verkaufe meine alte Schrottlaube über eine Online-Plattform. Sie interessieren sich für mein Auto und treffen sich mit mir im Café um die Ecke, um eine Probefahrt zu machen. Sie zeigen mir Ihren Führerschein. Ich sage: „Wunderbar. Sie sind offenbar der nette Interessent von der Online-Plattform. Ich vertraue Ihnen. Hier ist der Schlüssel. Machen Sie eine Probefahrt, und wir sehen uns dann in ein paar Minuten wieder hier." – So ähnlich ist es bei der direkten Authentifizierung. Ich erkenne Ihren Namen und händige Ihnen den Autoschlüssel direkt aus.

Nehmen wir nun aber an, dass ich Autohändler bin und in meiner Kfz-Niederlassung einen Luxuswagen verkaufe. Sie treffen mich im Laden und zeigen mir Ihren Führerschein. Ich habe keinen Hauptschlüssel, der für alle Autos auf dem Hof passt. Ich muss Ihre Daten also anhand Ihres Führerscheins in einem Computersystem erfassen, das dann überprüft, ob sie gültig sind. Ist dies der Fall, wird ein Kästchen entsperrt, dem ich einen Schlüssel entnehme, der ausschließlich zu dem Fahrzeug passt, für das Sie zur Probefahrt angemeldet sind. – Dies ist mit einem ticketbasierten System vergleichbar: Ich greife auf ein zentralisiertes System zurück, um einen Schlüssel auszugeben, der jetzt über das Ticket mit Ihnen verknüpft ist.

OAuth, Kerberos, Single Sign-on und Webformulare sind alles Beispiele für ticketbasierte Authentifizierungssysteme. Sie können sogar ein eigenes, benutzerdefiniertes Authentifizierungssystem entwickeln. Es gibt zwar viele verschiedene Methoden für die Implementierung dieser Art von Protokoll, aber die meisten ticketbasierten Systeme sind in ihrer Struktur einander ähnlich: Man fordert ein Ticket oder Token an und verwendet es dann für den Zugriff auf geschützte Daten oder Endpunkte.

Die Tickets von ticketbasierten Authentifizierungssystemen haben starke Ähnlichkeit mit den oben beschriebenen API-Schlüsseln. Ein wesentlicher Unterschied besteht jedoch darin, dass Tickets kurzlebig sind. Sie sind nur für eine kurze Zeitspanne gültig und können problemlos widerrufen werden, was die Sicherheit erhöht.

Monitoring mit ticketbasierter Authentifizierung

Ein effektives API-Monitoring mit ticketbasierter Authentifizierung setzt voraus, dass Sie in der Lage sind, mehrere Schritte durchzuführen und das Ticket oder Token in einer Variablen zu speichern, die Sie in nachfolgenden Schritten wiederverwenden können.

Ein einfaches Beispiel hierzu: Sie setzen eine Anforderung mit einem Benutzernamen und Passwort und einer Spezifikation im Header ab, rufen ein Token aus dem System ab, speichern dieses Token als Variable und senden dann eine weitere Anforderung an einen Endpunkt mit dem Token als Header.

Falls Sie nicht schon dabei sind, eine Authentifizierung für Ihre API zu implementieren, dann sollten Sie schleunigst damit anfangen, um Ihre Daten und Systeme zu schützen. Zugleich sollten Sie auch sicherstellen, dass Ihre externen Monitoring-Systeme dazu berechtigt und in der Lage sind, die Performance und Zuverlässigkeit Ihres Systems zu überwachen. Wenn Sie Ihre API-Performance nur auf der Anwendungsseite überwachen, könnten Ihnen alle möglichen Verbindungsprobleme entgehen, die Ihre Endanwender daran hindern, über Ihre API auf Daten zuzugreifen oder Änderungen vorzunehmen.

Monitoring und Prüfen von Daten

Beim Monitoring einer Website im Browser müssen wir mehr tun als nur den Antwortcode zu prüfen und zu checken, ob Inhalte oder Bilder auf der Seite geladen werden. Gibt die Seite „200 OK“ zurück, bleibt aber vollständig leer, dann werden wir das sofort untersuchen. Dasselbe gilt für das Monitoring von API-Endpunkten. Beim Monitoring von API-Endpunkten sollte man nicht nur sicherstellen, dass der Antwortcode den Erwartungen entspricht, sondern auch, dass die richtigen Daten im richtigen Format zurückgegeben werden.

Sehen wir uns die einzelnen Schritte eines einfachen Use Cases an, bei dem grundlegende Extract- und Assert-Optionen verwendet werden, um zu prüfen, ob eine API Daten im richtigen Format zurückgibt.

Datenvalidierung am Beispiel

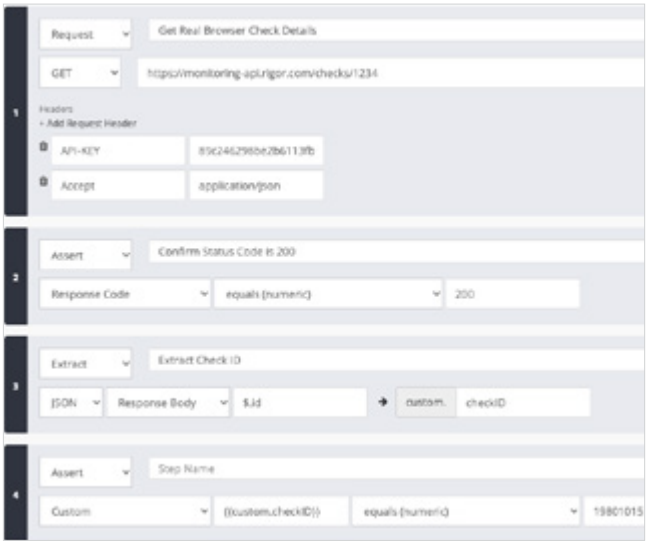
In **Splunk Optimization** gibt es eine **offene API**, die Splunk-Benutzer einsetzen, um regelmäßig Daten fürs Reporting abzurufen. Wenn ein Splunk-Benutzer zu Validierungszwecken einen API-Schlüssel an den Endpunkt übermittelt, ist es wichtig, dass wir den Antwortcode „200 OK“ und das Datenset für die richtige, dem Endpunkt entsprechende ID zurückgeben.

Wir können einen externen, synthetischen Test erstellen, um den Prüf-Endpunkt in einer festgelegten Häufigkeit von mehreren Standorten aus anzusprechen und Folgendes zu bestätigen:

1. Antwortcode = 200.
2. Die Prüf-ID in der JSON-Ausgabe entspricht dem angesprochenen URL-Endpunkt.

Im folgenden Beispiel verwenden wir eine API-Prüfung von Splunk für Folgendes:

1. Absenden einer Anforderung mit einem API-Schlüssel an den Splunk API-Endpunkt, um echte Browser-Prüfdaten zu bekommen.
2. Bestätigung, dass der Antwortcode den Wert „200“ enthält.
3. Extraktion einer Prüf-ID aus dem JSON-Code mit „JSON path“.
4. Bestätigung, dass die aus dem JSON-Pfad extrahierte Prüf-ID dem erwarteten Wert entspricht.



Dieser einfache Benutzer-Flow hilft uns beim Test auf …

- **Verfügbarkeit:** Die Prüfung schlägt fehl, wenn die API einen anderen Antwortcode als „200 OK“ zurückgibt.
- **Datenformat:** Gibt die API die Daten in einem anderen Format als JSON zurück, schlägt der Schritt zum Extrahieren eines Werts mit „JSON path“ und damit die gesamte Prüfung fehl.
- **Datenqualität:** Wenn wir einen Wert für die ID extrahieren können, dieser aber nicht dem erwarteten Wert entspricht, schlägt der Bestätigungsschritt und damit die Prüfung fehl.

Wenn wir z. B. eine Warnmeldung erhalten, dass unser externes Monitoring-System die Prüf-ID nicht aus JSON extrahieren konnte, sollten wir die Meldung aufrufen und die Ausgabe bzw. den Antworttext des API-Endpunkts unter die Lupe nehmen. Wenn wir uns den Antworttext ansehen, können wir schnell feststellen, ob das Format falsch war oder der ID-Wert nicht in der Ausgabe enthalten war. Anhand dieser Informationen können wir sofort mit dem Troubleshooting beginnen.

Dies war nur ein einfaches Beispiel, wie sich robustes Monitoring für eine API implementieren lässt. Falls Ihre API-Tests derzeit nur zum Monitoring von Antwortcode und Antwortzeit dienen, sollten Sie an diesem Punkt überlegen, ob Sie nicht zusätzliche Kriterien für Datenformat und -qualität hinzuzufügen.

Performance-Tests für Fehlerbedingungen schreiben

Bis jetzt haben wir uns angesehen, wie man beim Monitoring mit Anforderungsheadern, Authentifizierung und Datenvalidierung vorgeht. Wenn es nun um das Schreiben von Performance-Tests geht, so besteht eine Strategie darin, die Tests so zu schreiben, dass ein System eine API aufrufen und keine Daten erhalten kann.

Wenn Sie Code mit vielen lokalen Aufrufen schreiben, bleibt ein Wrapper, der eine externe API aufruft, im Kontext einer Anwendung oft unbemerkt. Wenn Ihr Test so konzipiert ist, dass er eine Warnmeldung ausgibt, falls keine Daten vorliegen, können Sie sicherstellen, dass Sie keine kritischen Fehler übersehen. Und vergessen Sie nicht, Ihren Code resilient zu gestalten, sodass er auch dann noch funktioniert, wenn er eine Fehlermeldung, beschädigte Daten oder gar keine Antwort erhält.

KAPITEL 4

Monitoring auf SLA-Einhaltung

Wir haben weiter oben schon kurz die Schwachstellen angesprochen, die sich aus verzahnten, voneinander abhängigen Systemen ergeben, die Daten über das Web senden und empfangen. Da immer mehr Anwendungen auf Drittanbieter-APIs aufbauen, ist es für Unternehmen von entscheidender Bedeutung, dass Partner-APIs schnell, sicher und zuverlässig funktionieren.

Unternehmen haben eine Methode entwickelt, um dieses Risiko in den Griff zu bekommen: durch Verträge, die in der Regel unter dem Kürzel SLA bekannt sind.

Was ist ein SLA?

Ein Service Level Agreement (SLA) ist eine Vereinbarung zwischen zwei Parteien, die regelt, welche Services von einer Partei für die andere erbracht werden sollen. Im Prinzip könnte eine solche Vereinbarung eine beliebige Anzahl von Services umfassen – von kundenspezifischen Support-Antworten bis hin zur Produktbereitstellung.

SLAs zwischen Technologie- oder Softwareanbietern haben meist folgende Punkte zum Gegenstand:

- **Verfügbarkeit:** Wie viel Uptime (in Prozent) kann der Partner garantieren? Wie viel Vorwarnzeit benötigt ein Partner vor einer geplanten Downtime oder Wartung?
- **Reaktionszeiten:** Wie schnell kann mein System eine Antwort vom Partnersystem erwarten?

Es kann außerdem vereinbart werden, dass eine Partei bei Nichteinhaltung bzw. Nichterfüllung des SLAs Anspruch auf eine Gutschrift bzw. Erstattung hat oder vom Vertrag zurückzutreten darf.

Sehen wir uns hierzu ein Beispiel an: Eine Web-App für die Vermittlung von Mitfahrgelegenheiten hängt stark von den Daten eines Drittanbieters ab, der auf Straßenkarten spezialisiert ist. Wenn die Betreiber der Mitfahr-App einwilligen, ausschließlich mit einem einzigen Kartenanbieter zusammenzuarbeiten, dann gibt der Kartenanbieter eventuell folgende Garantien: „Ihre App zur Mitfahrvermittlung wird zu 99,99 % der Zeit Zugriff auf unsere Kartendaten haben, und wir werden Ihr Team mindestens drei Wochen im Voraus über anstehende geplante Wartungsarbeiten informieren."

Das Team, das die Mitfahr-App betreut, könnte nun antworten: „Sehr schön! Das klingt sehr gut. Unsere Benutzer tolerieren ein gewisses Maß an Pannen und beklagen sich auch nicht gleich auf Twitter. 99,99 % Uptime ist also mehr als ausreichend. Und drei Wochen Vorlaufzeit genügen, um unsere Kunden über die geplante Downtime zu informieren. Wir stimmen den Bedingungen zu, doch falls Ihre API weniger als 99,99 % der Zeit zur Verfügung steht, haben wir Anspruch auf volle Kostenerstattung."

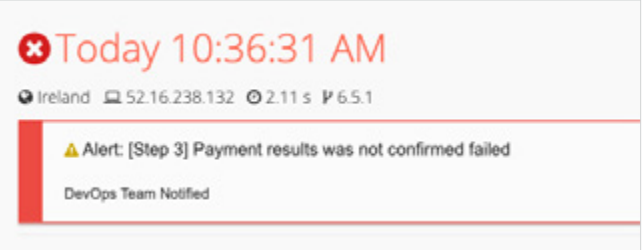
Alle setzen ihre Unterschriften unter die Vereinbarung, besiegeln das Ganze noch per Handschlag, und die Mitfahrvermittlung entwickelt für ihre Web-App eine neue Funktion, die eine Verbindung über die API des Kartenanbieters aufbaut und von dort Daten abruft.

SLA-Einhaltung sicherstellen

Als Geschäftsleitung des Kartenanbieters könnten wir jetzt sagen: „Wir brauchen unbedingt Daten zur Problemvorbeugung, damit wir unseren Teil der Vereinbarung ganz bestimmt erfüllen. Es wäre auch prima, wenn wir diese Daten unseren Partnern von der Mitfahrvermittlung offenlegen könnten, um ihnen damit zu zeigen, dass sie sich auf unseren Service verlassen können."

Wir könnten dazu das interne Monitoring unserer Anwendung nutzen, doch das würde uns vielleicht nur ein unvollständiges Bild liefern. Wie können wir feststellen, ob die Kartendaten über unsere API für Endbenutzer außerhalb unseres Systems zur Verfügung stehen? Wie können wir sicherstellen, dass die Daten nicht nur verfügbar sind, sondern auch im richtigen Format vorliegen?

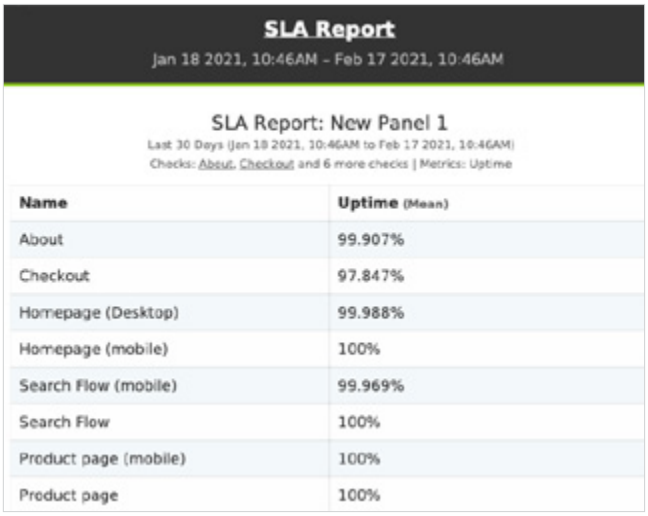
Wir können ein synthetisches, externes Monitoring einrichten, um den Abruf von Daten aus unserer eigenen API zu testen, und Warnmeldungen festlegen, damit unsere Techniker sofort informiert werden, wenn Probleme auftreten, die die Einhaltung unserer Service-Level-Vereinbarungen gefährden könnten.



Das obige Beispiel zeigt eine Warnmeldung einer API-Prüfung in Splunk Synthetic Monitoring.

Mit den proaktiven Daten, die Interaktionen echter Endanwender mit unserem Kartierungssystem simulieren, können wir

1. Performance-Probleme erkennen und beheben, bevor sie Folgen für unsere realen Anwender haben, und
2. Berichte an unsere Partner weitergeben, die zeigen, dass unsere Uptime tatsächlich die zugesagten Werte erreicht.



SLA Report

Jan 18 2021, 10:46AM – Feb 17 2021, 10:46AM

SLA Report: New Panel 1

Last 30 Days (Jan 18 2021, 10:46AM to Feb 17 2021, 10:46AM)
Checks: About, Checkout and 6 more checks | Metrics: Uptime

Name	Uptime (Mean)
About	99.907%
Checkout	97.847%
Homepage (Desktop)	99.988%
Homepage (mobile)	100%
Search Flow (mobile)	99.969%
Search Flow	100%
Product page (mobile)	100%
Product page	100%

SLAs bei Partnern durchsetzen

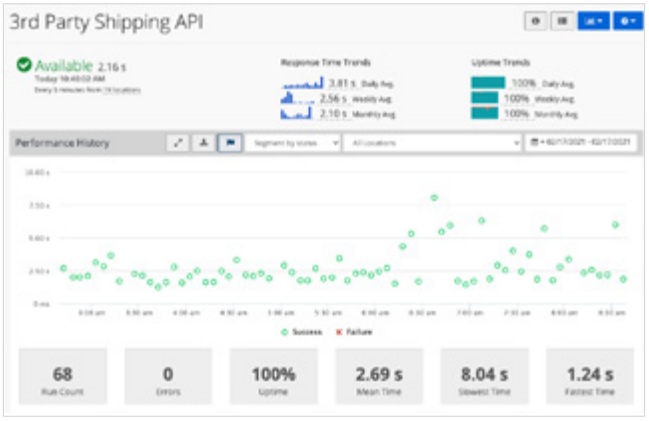
Die Geschäftsleitung der Mitfahrvermittlung könnte nun sagen: „Toll, dass ihr uns diese Berichte zur Verfügung stellt, liebe Kartenanbieter, aber wir müssen unseren eigenen Sorgfaltspflichten genügen und ein paar externe Daten mit euren Berichten vergleichen." Wir können zu diesem Zweck mit externen API-Prüfungen die Performance der Kartendaten-API überwachen und auf diese Weise herausfinden, ob sie wirklich 99,99 % der Zeit für Zugriffe zur Verfügung steht.

Falls wir feststellen, dass die Verfügbarkeit unter den zugesagten Werten der SLA liegt oder dass die API längere Zeit ausfällt, ohne dass dies drei Wochen im Voraus angekündigt wurde, können wir auf der soliden Basis unserer Berichte mit unserem Partner über den SLA-Verstoß und etwaige Konsequenzen sprechen. Dank der API-Prüfungen können wir auch besser verstehen, wie sich Probleme mit der API unseres Partners auf unsere Benutzer auswirken.

Beide SLA-Parteien können also mithilfe von API-Prüfungen herausfinden, ob die Bedingungen des Service Level Agreements eingehalten werden. Als API-Verantwortliche können Sie durch proaktives Monitoring außerdem Probleme abfangen, bevor sie sich auf Ihre Partner auswirken, und bekommen zudem die Möglichkeit, Berichte zu erstellen und Ihren Partnern vorzulegen. API-Endbenutzer können Sie durch proaktives Monitoring auf Drittanbieterprobleme aufmerksam werden, die sich auf Ihre eigenen Kunden auswirken könnten. Und Sie können sicher sein, dass Ihr Partner seine SLA-Zusagen auch einhält.

Im folgenden Beispiel sehen wir uns eine mehrstufige Prüfung der Kartendatenanbieter-API an. Wir kontrollieren dabei, ob die Daten im erwarteten Bereich liegen, und protokollieren die Verfügbarkeit bei Anforderungen.

Zur Erinnerung: Mit API-Prüfungen kann sowohl die Verfügbarkeit als auch die Reaktionszeit überwacht werden. Die Use Cases, die Sie für API-Prüfungen erstellen, sollten stets an den SLAs ausgerichtet sein. Wenn Ihre Vereinbarung nur die Verfügbarkeit betrifft, können Sie eine Prüfung konfigurieren, die den Endpunkt einfach anspricht, und sich dann den Prozentwert der Uptime ansehen. Beruht die Vereinbarung darauf, wie schnell eine API Daten zurückgibt, können Sie eine mehrstufige Prüfung einrichten, die Daten aus der API abruft, und dann die durchschnittliche Antwortzeit mit den vereinbarten Werten vergleichen.



KAPITEL 5

API-Prüfungen von Splunk

Bei Splunk haben wir ein Framework für Performance-Tests entwickelt, das Unternehmen hilft, festzustellen, ob APIs die erwartete Performance liefern.

Mit API-Prüfungen können Sie

- die Verfügbarkeit kritischer APIs nachverfolgen,
- API-Antwortzeiten in ihrer Entwicklung erfassen,
- die Funktion der API sicherstellen, indem Sie Antwortwerte und -struktur der API prüfen, sowie
- Warnmeldungen erhalten, sobald Bedingungen eintreten, die auf eine fehlerhafte oder nicht ausreichend performante API hindeuten; auf diese Weise fangen die Teams Probleme ab, noch bevor sich diese auf Anwender auswirken oder zu SLA-Verstößen führen.

Die API-Prüfungen von Splunk sind aus vier einfachen Komponenten aufgebaut:

1. HTTP-Anforderungen absetzen,
2. Daten aus Antworten extrahieren,
3. Daten speichern, zur Verwendung in weiteren Anforderungen oder Analysen, sowie
4. Befunde prüfen: ob Daten zurückgegeben werden und ob sie im richtigen Format vorliegen.

Diese Komponenten sind einfach, aber erstaunlich leistungsfähig. Wie bei Fraktalbildern können durch die Kombination ganz einfacher Komponenten wunderschöne, komplexe Strukturen entstehen. Die API-Prüfungen von Splunk verwenden einfache Bausteine, die sich zu Testszenarien zusammensetzen lassen, mit denen das Monitoring und Prüfen selbst der kompliziertesten API-Ströme möglich wird.

Unsere Techniker arbeiten bei der Entwicklung und den Beta-Tests der API-Prüfung eng mit Kunden zusammen, um sicherzustellen, dass die Prüfung mit diesem einfachen Ansatz gut verständlich ist und gleichzeitig die Anforderungen der Kunden erfüllt. Beim Beta-Testing war unter anderem Lookout beteiligt, ein führender Anbieter von Mobile-Security-Lösungen. Dean Ross-Smith, seines Zeichens Cloud Operations Engineer bei Lookout, äußert sich wie folgt über API Check:

„Dank der Flexibilität des neuen Splunk API Checks kann Lookout jetzt eine Prüfung konfigurieren und ein erfolgskritisches Element der Infrastruktur überwachen. Diese Möglichkeit hatten wir bisher noch bei keiner anderen Lösung."

Für Unternehmen gemacht

Ganz gleich, ob der Web-Auftritt, die Mobil-App oder die Kerninfrastruktur Ihres Unternehmens auf APIs basiert – der API Check von Splunk stellt sicher – und zwar auf eine Art und Weise, die für Benutzer mit jedem technischen Kenntnisstand nachvollziehbar ist –, dass die API funktioniert, verfügbar ist und die erwartete Leistung bringt.

Unsere API-Prüfung ist die Antwort auf eine ganze Reihe von Unternehmensanforderungen, unter anderem diese:

- Testen komplexer, mehrstufiger API-Abläufe
- Monitoring von Verfügbarkeit und Antwortzeiten von Standorten in aller Welt aus
- Nachverfolgung und Durchsetzen der Performance-SLAs von Drittanbieter-APIs
- Prüfung der Korrektheit von API-Antworten
- Testen des gesamten CRUD-Lebenszyklus eines Datenobjekts mittels API
- Umgang mit komplexen tokenbasierten API-Authentifizierungssystemen
- Monitoring von Anwendungsstatusseiten

Erkenntnisse

Gibt es bei Ihnen eine Anwendung, die von einer eigenen API abhängt oder auf eine Drittanbieter-API angewiesen ist? Stellen Sie selbst Ihren Kunden Daten über eine API bereit? Brauchen Sie mehr Transparenz in Bezug auf die Verfügbarkeit, Funktion und Performance einer API? Das neue API Check für Splunk Synthetic Monitoring ist die Antwort auf all diese Fragen.

Wenn Sie mehr über die Web-Performance-Produkte von Splunk erfahren oder einen Demo-Termin vereinbaren möchten, kontaktieren Sie uns – am besten gleich heute.

Weitere Informationen über API Check mit allen Einzelheiten und schrittweisen technischen Schritt-für-Schritt-Anleitungen finden Sie in der **Splunkbase.**

splunk>

Weitere Informationen: www.splunk.com/asksales

www.splunk.de

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