09 | 22 Sonderdruck für X-INTEGRATE Software & Consulting

IT Administrator
Das Magazin für professionelle System- und Netzwerkadministration

Ereignisgesteuerte Architekturen

www.it-administrator.de

Quelle: deshacam – 123RF

Ereignisgesteuerte Architekturen

Großes Event

von Thomas Zöller und Andreas Roth

Flexibilität ist das Mantra der Stunde: Unternehmen streben danach, Geschäftsmodelle schnell und agil auf neue Marktanforderungen umzustellen beziehungsweise neue Ansätze zu entwickeln. Dazu passt keine Softwarearchitektur, innerhalb derer Prozesse stark ineinandergreifen. Daher ist derzeit viel von ereignisgesteuerten Architekturen die Rede, die einzelne Softwarekomponenten entkoppeln und als kleinteilige Services verbinden. Daten lassen sich so asynchron und in Echtzeit verarbeiten. Wir zeigen die Herausforderungen für den Admin in solchen Infrastrukturen.

Anders als in monolithischen Softwarearchitekturen verläuft die Kommunikation in einer Event-Driven Architecture (EDA) asynchron. Ein Message-Broker orchestriert den Austausch der Events in Form von Nachrichten. Eine solche beschreibt jeweils ein Event, also ein für andere Services relevantes Ereignis innerhalb oder außerhalb des Systems. Ereignisse können entweder den Zustand eines Datenobjektes beschreiben oder Identifikatoren sein. Nachrichten sind in Topics und Partitionen organisiert, wobei jedes Topic aus vielen Partitionen bestehen kann. Jede Partition ist eine geordnete, unveränderbare Abfolge von Nachrichten, in die sich neue Nachrichten mit der Zeit integrieren.

Basis von EDAs ist häufig Apache Kafka. Über diese Plattform lassen sich Event-Streams in Echtzeit veröffentlichen, abonnieren, speichern und verarbeiten. Kafka minimiert die Notwendigkeit von Punkt-zu-Punkt-Integrationen für die gemeinsame Datennutzung in bestimmten Anwendungen und Latenzen reduzieren sich auf Millisekunden. Verfügbar ist die Open-Source-Plattform, die unter der Apache-2.0-Lizenz steht, unter [1].

EDA als Herausforderung für den Admin

Sowohl im Bereich Continuous Integration/Continuous Delivery (CI/CD) als auch im Monitoring und Logging stellen solche Architekturen die operativen IT-Verantwortlichen vor neue Herausforderungen. In einer EDA liegen wie beschrieben kleinteilige Microservices vor, die Event-basiert meist asynchron miteinander kommunizieren und querbeet in der Anwendungslandschaft und Infrastruktur verteilt sind. Um solche Konstrukte sinnvoll aufzubauen und zu betreiben, bedarf es einer sauberen konzeptionellen Grundlage im Bereich der Anwendungsunterstützung und Anwendungsverteilung (CI/CD). Aber auch die Überwachung wie auch generell die Auswahl der für eine solche Architektur geeigneten Produkte bedarf besonderer Aufmerksamkeit.

In einer verteilten Anwendungslandschaft müssen sich Administratoren von vornherein Gedanken über Softwareberatung und -verteilung machen. Anstatt großer monolithischer Applikationen, die siloartig auf Systemen laufen, sind viele kleine Services einzeln auszurollen und zu verwalten. Sie liegen auf lokalen Applikationsstacks oder Servern in der Cloud, können aber ebenso gut als gekapselter SaaS-Ansatz bei einem Drittanbieter gehostet sein.

Wie IT-Verantwortliche diese Services letztlich ausbringen, ist keine Aufgabe für die Entwicklung, sondern für DevOps beziehungsweise Infrastrukturteams. Build- und Deployment-Automatisierung sind heute ein gängiges Muster und einer der Kernaspekte hinter DevOps. In einer EDA ist es noch wichtiger, CI/CD konzeptionell früh einzubetten und sich zu überlegen, wie das Unternehmen Anwendungen baut und sie verteilt. Anderenfalls sind hinterher ein effizientes Management und die kontinuierliche Aktualisierung der Anwendungslandschaft kaum noch möglich.

Zentrales Monitoring

Nach ihrem Aufsetzen stellt sich die Frage, wie diese Art von Architektur zu überwachen ist: zum einen über Konzepte wie zentralisiertes Logmanagement und Monitoring. Denn bei vielen verteilten Diensten funktioniert es nicht mehr, sich kurz auf jedes System aufzuschalten, von denen jedes sein eigenes Logging mitbringt. Vielmehr bedarf es einer Zentralisierung, um das asynchrone Zusammenspiel mehrerer kleinteiliger Services überhaupt überwachen zu können.

Producer und Consumer [2] haben bei der EDA, anders als in einer klassischen Systemlandschaft, wenig miteinander zu tun, denn die verteilten, entkoppelten Services haben keine engeren Berührungspunkte. Tritt in einem Dienst ein Fehler auf, sind nachgelagerte Systeme davon zunächst unbenommen – im Gegensatz zur klassischen Struktur, wo beim Stillstand einer Komponente rasch der gesamte Datenfluss stockt. Weil also in einer

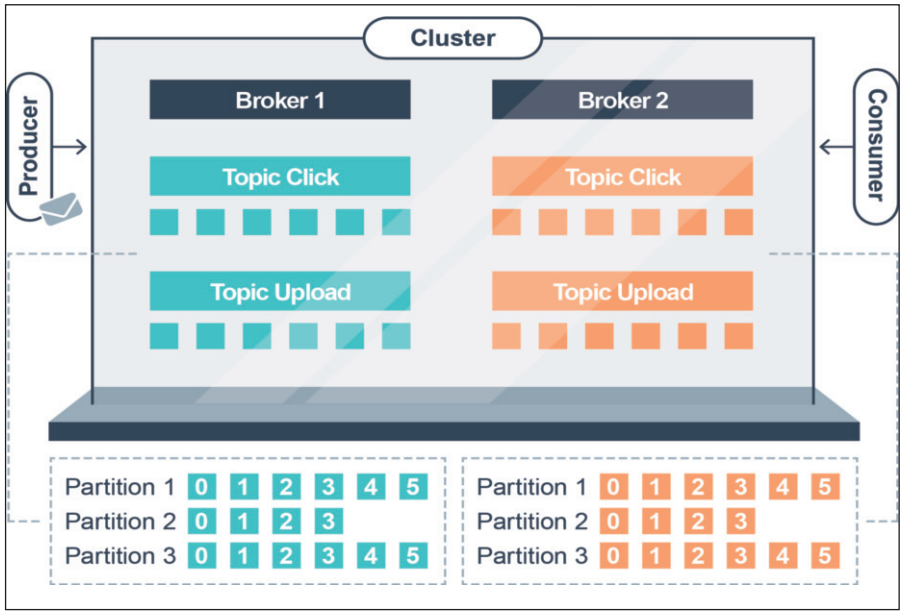
EDA kleinteilige Services jeder für sich gesehen weniger Funktionsumfang abdeckt, wird es immer schwieriger, aus dem großen Ganzen heraus zu erkennen, dass irgendwo ein Problem aufgetreten ist. Deshalb ist ein zentrales Monitoring wichtig, um alles im Blick zu behalten. Erkennt das Monitoring ein Problem, muss dieses im Zuge von CI/CD automatisch behandelt werden. CI/CD sollte daher die Überwachung abdecken und dynamisch verwalten.

Der Microservices-Ansatz ist schließlich auf schnelle und hohe Skalierbarkeit ausgelegt. Steigt daher die Anzahl der Dienste, sollte dies auch gleich in der Überwachung automatisch ankommen. Fazit: Überwachung und CI/CD sind in einer EDA miteinander verwoben, was Administrationsabteilungen vor die Herausforderung stellt, alles im Blick zu behalten.

Monitoring ist an dieser Stelle nicht mit Logmanagement zu verwechseln. Logs geben darüber Aufschluss, was in den einzelnen Komponenten tatsächlich geschehen ist. Das Monitoring dagegen erklärt, wie sich die Ressourcen über die Zeit verhalten – was relevant für den Admin ist, der seine Umgebung hinsichtlich der Auslastung überwachen möchte, insbesondere wenn er die Skalierbarkeit betrachtet. Das Monitoring verzeichnet, wie hoch die Auslastung war, ob einzelne Services nach oben skaliert wurden, um die Last besser abzufangen et cetera. So lassen sich erstens Rückschlüsse darüber ziehen, wie sich das gesamte System – nicht nur die einzelne Anwendung – über den Zeitraum verhalten hat, und zweitens, wie verfügbar einzelne Events sind. Dies geht aus dem Logmanagement nicht hervor. Dort müssten IT-Verantwortliche tatsächlich jede einzelne Bearbeitung protokollieren, was allein der großen Datenmengen wegen unpraktikabel ist.

Management verteilter Anwendungslandschaften

In traditionellen Anwendungen mit klassischer Datenverarbeitung ist dem Admin bekannt (weil alles synchron läuft), welche Anwendung mit welcher Subkomponente oder anderen Anwendung spricht und wie die Datenflüsse aussehen. In EDA dagegen hat er es mit einer losen Kopplung von Consumern und Producern zu tun. Umso wichtiger ist es, Möglichkeiten der Nachverfolgung im Monitoring einzubetten. So bleibt der Admin aussagefähig darüber, wie sich ein Prozess über die verschiedenen Eventverarbeitungen ausgestaltet.



Die Informationsverteilung mit Apache Kafka läuft über Nachrichten, die wiederum in Topics und Partitionen organisiert sind.

Die IT-Abteilung muss in einer verteilten Anwendungslandschaft zudem Qualitätskriterien für die einzelnen Services definieren, die in CI/CD abzufragen sind. Beispiel: Bedienen sich alle Services einer zentralisierten Fehler-Queue? Wenn durchweg asynchron kommuniziert wird, ist zu untersuchen, ob alle entwickelten Dienste hinterher an ein zentrales Fehlermanagement angeschlossen sind. Dies sollten IT-Verantwortliche beim Deployment der Anwendung bereits berücksichtigen und im Zweifelsfall prüfen.

Sind die Events vorab entsprechend definiert, lässt sich im Anschluss – ohne die Anwendungslandschaft und Verteilung der Komponenten im Vorhinein genau zu kennen – der Event-Fluss im Tracing visuell darstellen. Programme wie Thundra [3] sind auf das Monitoring solcher verteilten Infrastrukturen und die Event-Visualisierung spezialisiert. Sie sind auch in nicht Event-basierten Architekturen einsetzbar und überwachen dort keine Events, sondern andere Systemparameter. In einer EDA lassen sie sich einsetzen, um eine Recognition durchzuführen und den eigentlichen Nachrichten- und Datenfluss visuell darzustellen.

Fazit

EDA-Strukturen basieren auf Technologien, Lösungsansätzen und Komponenten, die in einer klassischen Anwendungslandschaft oder Infrastruktur bislang zum Teil nicht zu finden sind – weil sie nicht notwendig sind. Die Aufgabe von IT-Abteilungen beim Umbau hin zu einer EDA-Infrastruktur ist es, die richtigen Komponenten auszuwählen und zu entscheiden, welcher Service lokal und welcher in der Cloud läuft. Viele der neuen Komponenten existieren in dieser Form zudem noch nicht lange auf dem Markt. Deshalb stellt allein ihr Einsatz an sich bereits eine Herausforderung dar, da sie zunächst in die bestehende Infrastruktur eingebettet werden müssen. *(jp)* IT

Thomas Zöller und Andreas Roth sind Project Manager/Solution Architects bei X-INTEGRATE.

Link-Codes

[1] Apache Kafka
m9z01

[2] Producer und Consumer in Kafka
m9z02

[3] Thundra
m9z03