E-BOOK

Ein wahrer Krimi:

So kommen Sie mit Cloud Insights Problemen in Ihrer Stream-Processing-Kubernetes-Applikation auf die Spur

NetApp

Inhalt

Die Akteure	04
Die Tat	05
Der Router: Die Ermittlungen beginnen	06
Die Aufnahme: Ein neuer Verdächtiger	07
Der Message Broker: Der Mittelsmann	10
Der Prozessor: Wir kommen der Sache näher	11
Der Storage: Es wird spannend	12
Die Berechnung: Was machen unsere AWS EC2 Instanzen?	14
Fazit: Eigentlich lag es die ganze Zeit auf der Hand	16

Ein wahrer Krimi:

So kommen Sie mit Cloud Insights Problemen in Ihrer Stream-Processing-Kubernetes-Applikation auf die Spur

In den letzten paar Jahren hat sich die Applikationsentwicklung von Grund auf verändert. Durch die Übernahme von ereignisgesteuerten Microservice-Architekturen und die neuen Möglichkeiten der Applikationsskalierung ist es zu Komplexitäten gekommen, die noch immer schwer zu handhaben sind. Dementsprechend ist ein Bedarf an Monitoring-, Troubleshooting- und Planungslösungen entstanden, die uns einen Überblick darüber geben, wie sich die Komponenten unserer Applikationen verhalten, und die die Systemstabilität fördern.

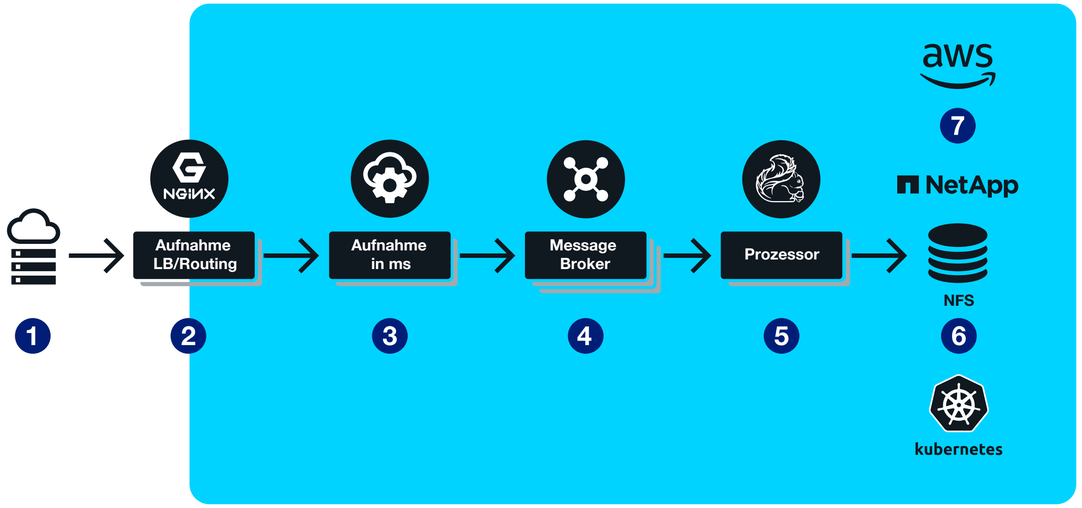
Wir bei NetApp setzen unsere eigene Lösung NetApp Cloud Insights ein, wenn wir neue Versionen von Cloud Insights und andere neue Produkte entwickeln, um sicherzustellen, dass sie genau so funktionieren, wie wir es uns wünschen.

In dieser Serie erzählen wir Ihnen von einem konkreten Beispiel für die Nutzung von Cloud Insights bei der Fehlerbehebung in unserem System. Es geht um die Ermittlung des Täters, um seine Tat (schlechte Performance) und um mehrere Verdächtige (Architekturkomponenten). Am Ende war es natürlich wie immer der Butler (die fehlerhafte, das Problem verursachende Konfiguration). Es treten auf: Load Balancer (NGINX), Microservices (K8s Pods), Message Broker (Kafka), Stream Processing (Flink), zugrunde liegender Storage (NetApp Cloud Volumes ONTAP FlexVol Technology) und sogar AWS EC2 Virtual Machines. Wie sich herausstellen wird, ist der Schuldige an einem Ort zu finden, den Sie normalerweise nicht mit Ihrer Umgebung in Verbindung bringen (Storage und Apps außerhalb Ihrer K8s-Umgebung, die sich auf die Abläufe in dieser Umgebung auswirken). Mit Cloud Insights gehen wir den verschiedenen Anhaltspunkten nach. Am Ende der Serie haben wir unsere gesamte Umgebung erfasst und sämtliche ihrer Komponenten untersucht.

Folgen Sie mir – das Spiel hat begonnen!

Die Akteure

Wir möchten Ihnen nun die Hauptakteure dieser Geschichte vorstellen – die Komponenten unserer Architektur. Wir entwickeln eine Stream-Processing-Applikation. Da diese Geschichte auf einer wahren Begebenheit beruht, haben wir die Namen zum Schutz der Betroffenen geändert und sie ein wenig gekürzt, damit auch alles auf Ihren Bildschirm passt. Diese App empfängt externe Anfragen, und die Daten passieren eine Reihe von Prozessen. Am Ende werden sie im Filesystem abgelegt.



Bei unserer Applikation handelt es sich um eine typische Stream-Processing-Applikation, die in Kubernetes (K8s) in einer AWS-Umgebung ausgeführt wird. Dies sind ihre Hauptkomponenten:

1. Der Agent
Ein externes Tool, das Datenpunkte erfasst und an unser System sendet.

2. Der Router
Ein Router/Load Balancer für eingehende Daten. Für diesen Zweck nutzen wir einen NGINX Ingress Controller. NGINX empfängt Anfragen, leitet sie basierend auf URL-Mustern weiter und führt einen Lastausgleich über die Pods eines Aufnahmeservices durch.

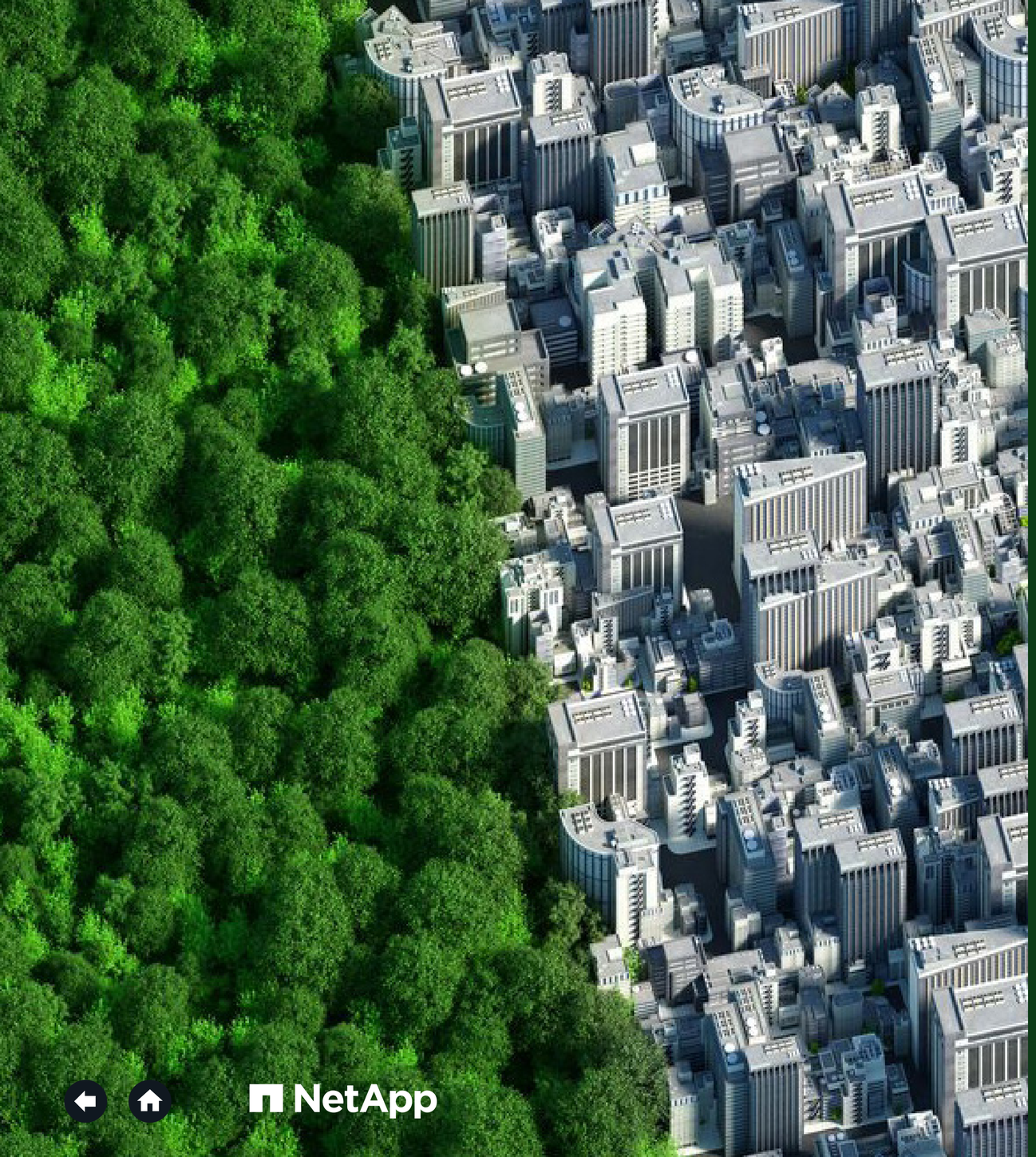
3. Die Aufnahme
Ein Microservice, der die Daten vom Agent empfängt, eine einfache Validierung durchführt und die akzeptierten Datenpunkte in ein festgelegtes Kafka Thema schreibt.

4. Der Message Broker
Ein Kafka-Cluster, der auf mehreren Brokern ausgeführt wird. Für ein Kafka-Thema verantwortlich, das Anfragen speichert, die auf ihre Verarbeitung warten.

5. Der Prozessor
Eine Stream-Processing-Engine. Wir nutzen einen Flink-Job, um die von Kafka übermittelten Daten zu verarbeiten. Dieser Job wird auf einem Flink-Cluster mit mehreren Taskmanagern ausgeführt. Der Job wandelt die Daten um und schreibt die Ergebnisse abschließend auf die Festplatte.

6. Der Storage
Die vom Flink Job genutzte Festplatte, ein NFS-Mount auf Flink-Taskmanagern. Für unseren zugrunde liegenden Storage nutzen wir Cloud Volumes ONTAP.

7. Die Berechnung (Compute)
Wir führen unsere Applikation in Kubernetes („K8s“) aus. Unsere Kubernetes-Nodes werden als AWS EC2 Instanzen bereitgestellt.



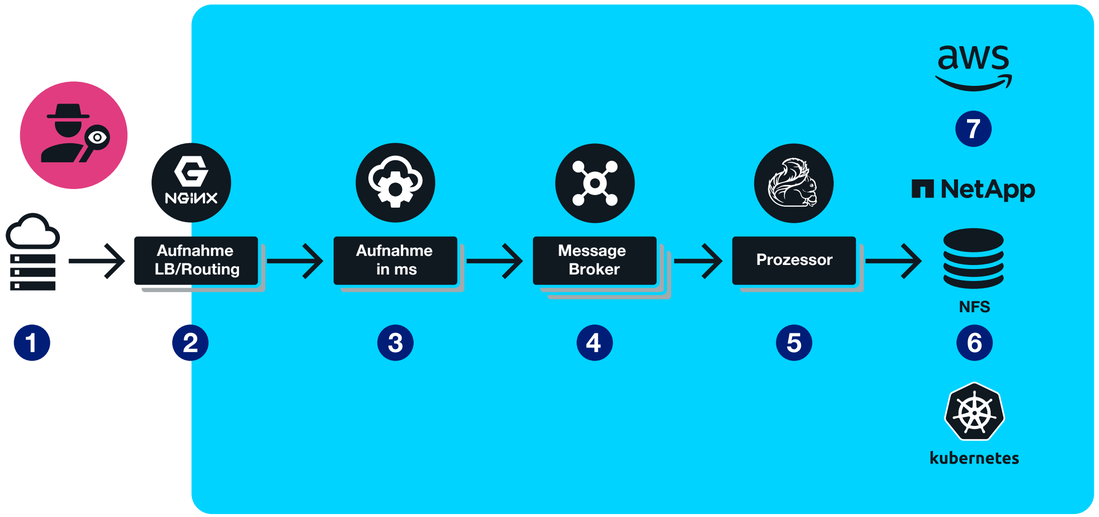
Die Tat

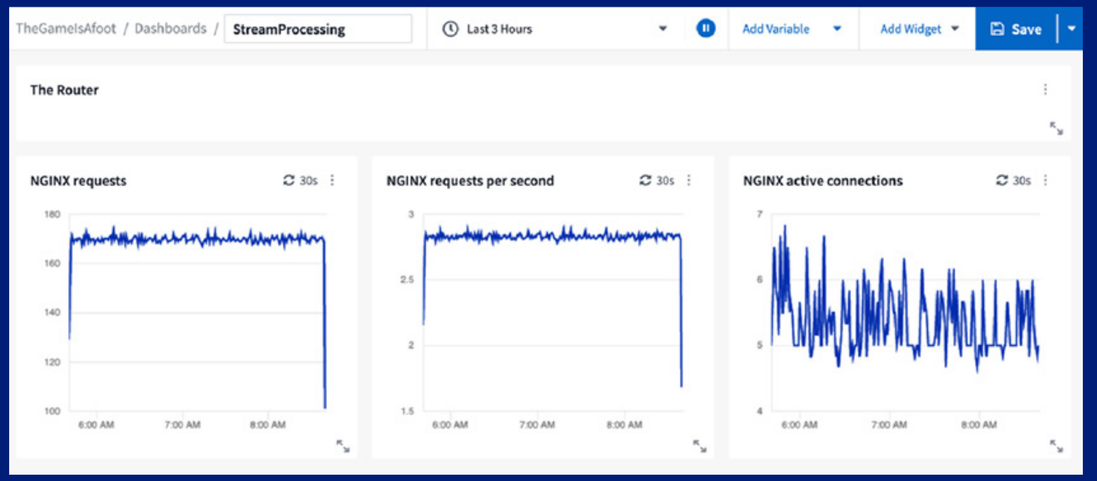
Der Tag begann wie jeder andere:
Die Daten flossen, alles war in bester Ordnung. Doch plötzlich fiel uns etwas Seltsames auf: Unsere neuesten Daten wurden nicht angezeigt. Irgendwann erschienen sie dann doch, aber die Verzögerung war deutlich. Irgendetwas hinderte die Ergebnisse daran, pünktlich in unserem System anzukommen. Doch angesichts all der vielen Komponenten konnten wir die Ursache des Problems nicht ausmachen. Es ging ein paar Tage lang so weiter, und wir stellten fest, dass es etwa um die 40. Minute jeder Stunde etwas länger dauerte, bis die Daten verarbeitet und zur Verfügung gestellt wurden. Das System fing sich stets letztendlich wieder, doch jede Stunde kam es zu genau demselben Zeitpunkt zu dem ungewöhnlichen Verhalten.

Wir beschlossen, uns jede einzelne Komponente unserer Architektur genauer anzusehen. Auf ins Verhör!

Der Router:
Die Ermittlungen beginnen

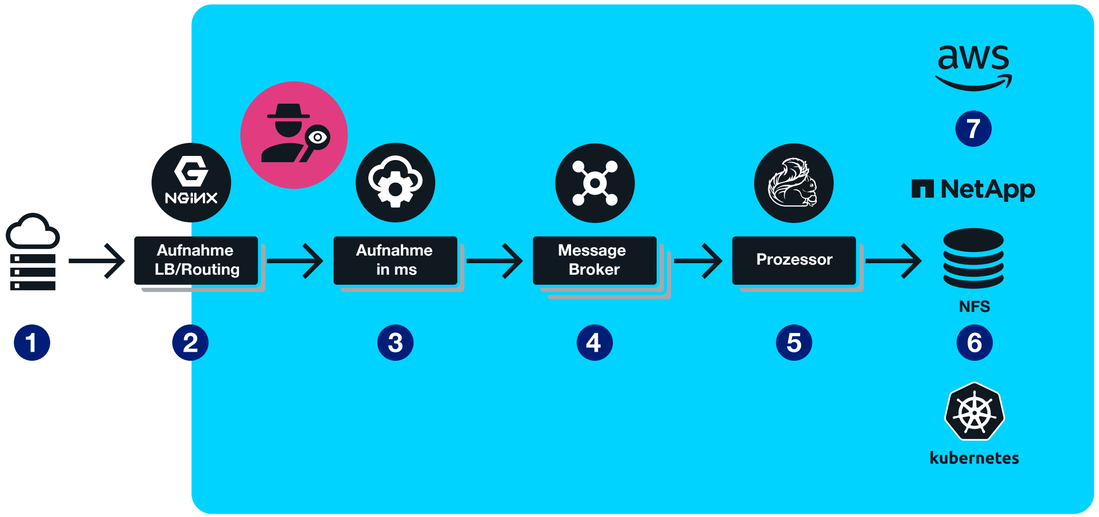
Wir müssen uns ein Gesamtbild von der Umgebung verschaffen und fangen daher ganz am Anfang an. Der erste Einstiegspunkt in unsere Applikation ist unser Ingress Controller (Router/Load Balancer). Wir beginnen also damit, Informationen von unserem NGINX Controller zu erfassen. Cloud Insights ermöglicht es uns, Infrastruktur- und Anwendungsdaten zu erfassen. Um herauszufinden, was mit unserer NGINX Komponente los ist, sehen wir uns die erfassten NGINX Daten an. Vorläufig nehmen wir an, dass vielleicht etwas die Daten daran hindert, zu unserem NGINX Controller zu gelangen.

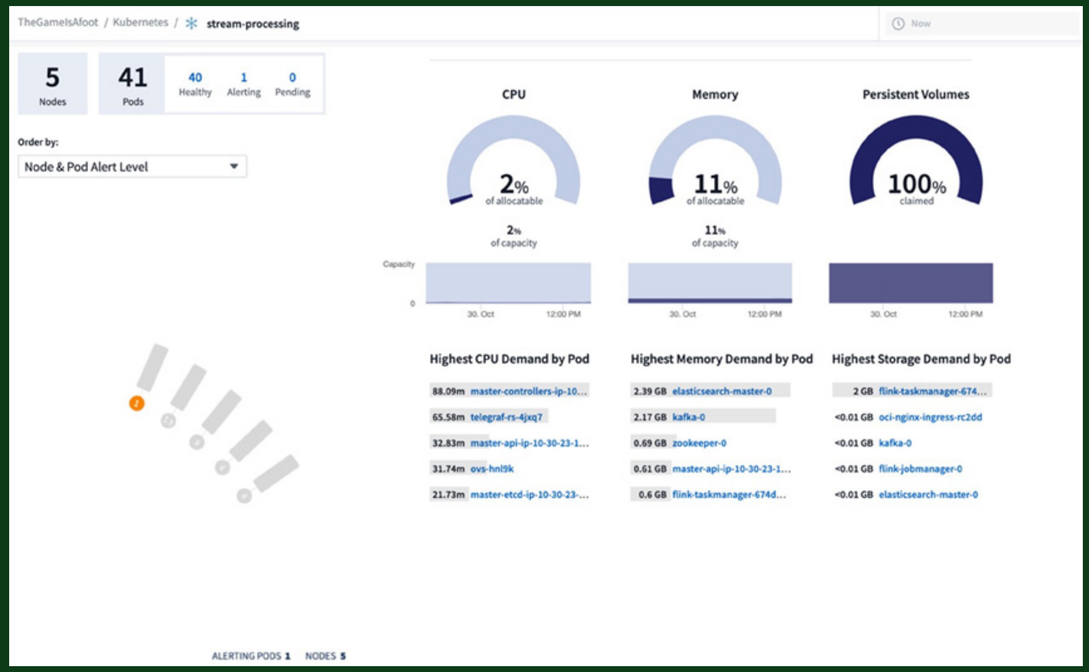




Mhm … Unsere NGINX Zahlen zeigen deutlich, dass wir mit dieser Annahme falsch liegen. Es liegt keine Beeinträchtigung oder wesentliche Veränderung beim Verhalten unseres NGINX Controllers vor. Die Daten werden den ganzen Tag über stetig und mit gleichbleibenden Raten empfangen. NGINX, wir lassen Sie laufen. Wir haben keine weiteren Fragen an Sie. Aber bleiben Sie vorerst in der Stadt.

Die Aufnahme:
Ein neuer Verdächtiger

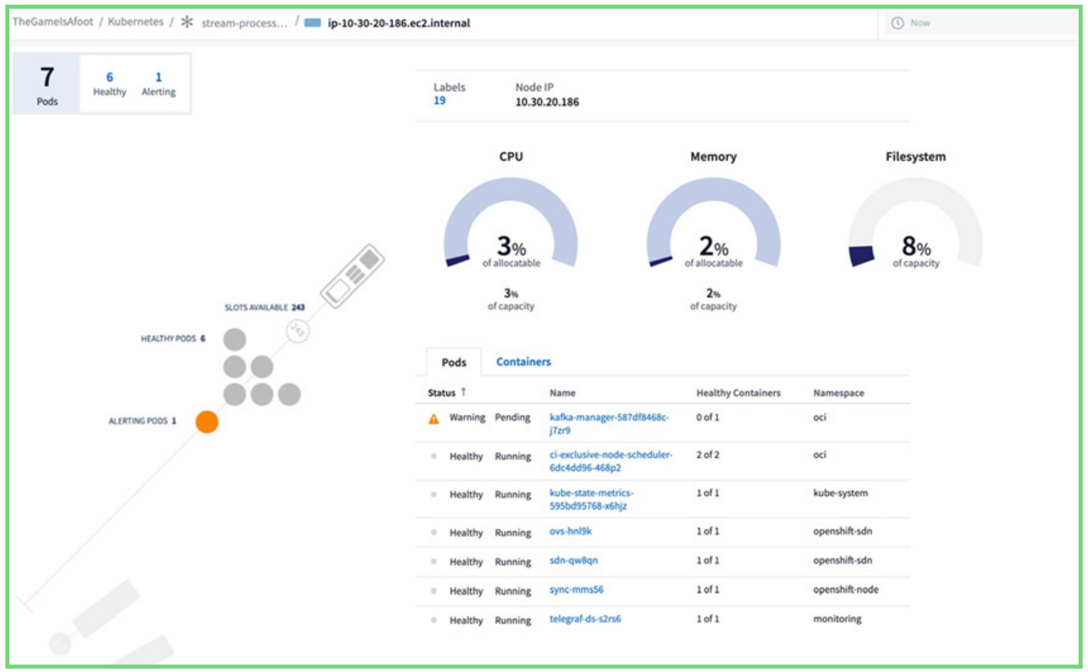




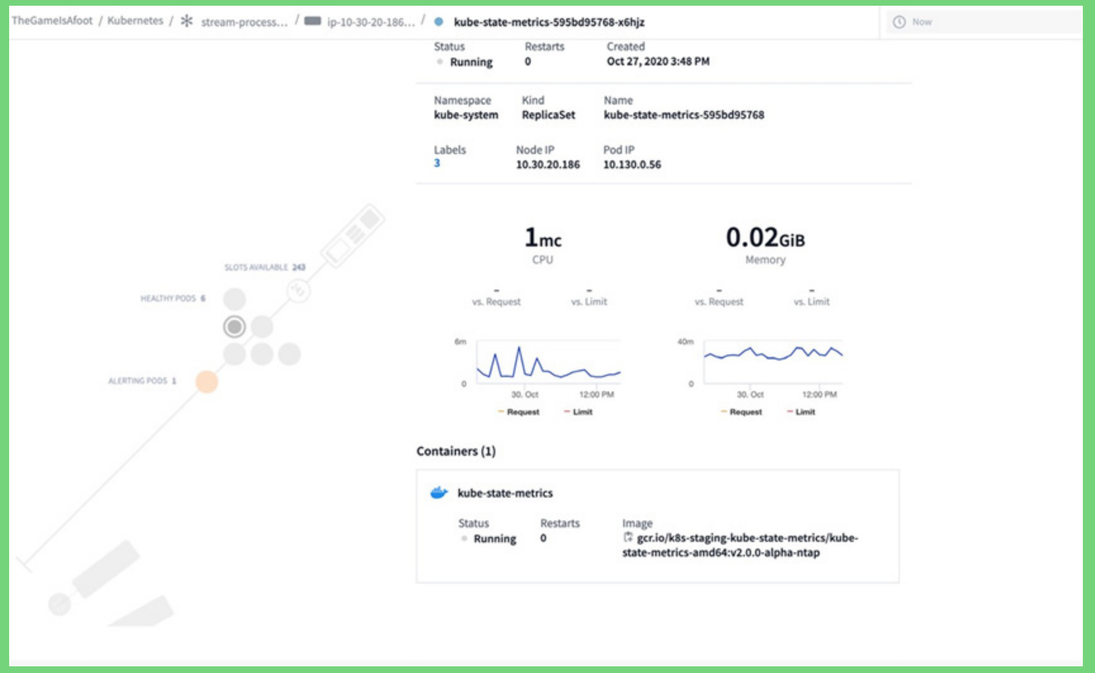
NGINX ist also sauber. Wir machen beim nächsten Verdächtigen weiter – bei unserem Aufnahme-Microservice. Der Aufnahmeservice wird als Kubernetes-Implementierung bereitgestellt und folgt allen Best Practices zu Ausfallsicherheit, Fehlertoleranz und Skalierbarkeit. Doch könnte er an irgendeiner Stelle aus der Reihe tanzen und das Problem verursachen? Wir müssen ihn uns genauer ansehen. Da es sich um einen als Pod implementierten Microservice handelt, entscheiden wir uns für eine Observierung von Kubernetes. Wie viele Daten empfangen und senden die Pods? Geraten die Daten vielleicht dort irgendwo auf Abwege?

Im Kubernetes Cluster Explorer von Cloud Insights klicken wir auf unseren Cluster, um uns die Statistiken anzusehen.

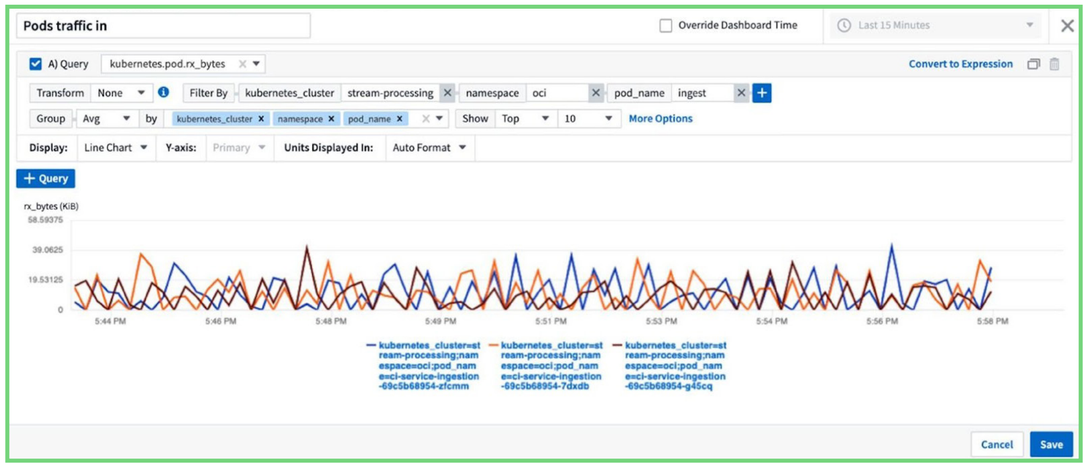
Wir sehen den Zustand unseres Clusters, unter anderem die Top-User des Clusters und seine Gesamtauslastung. Wir können sogar die einzelnen Nodes genau unter die Lupe nehmen.



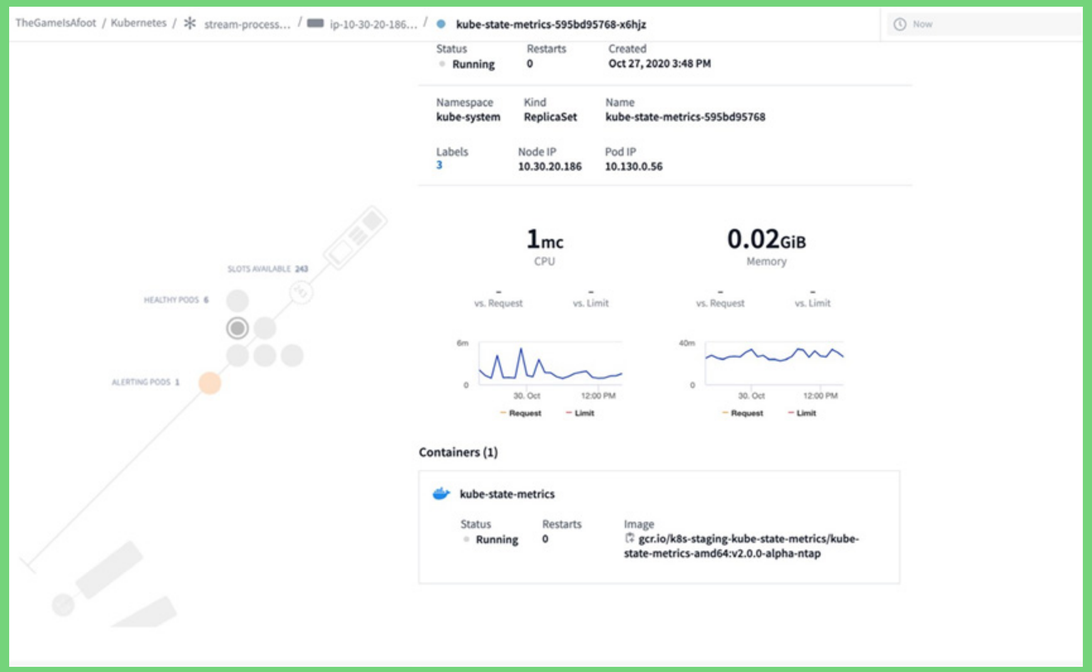
Jetzt können wir alle Pods und Container sehen, die auf unserem Pod ausgeführt werden, und erhalten eine Übersicht über unsere Nutzung des Nodes. Wir können auch einen einzelnen Pod genauer unter die Lupe nehmen und die zu ihm gehörenden Informationen sowie einige einfache CPU- und Arbeitsspeicher-Verlaufsdaten anzeigen lassen.



In diesem Fall möchten wir uns allerdings das Aufnahmeverhalten unserer Pods insgesamt ansehen. Dazu können wir die Kubernetes Performancekennzahlen sowie die Daten zum Rest der Applikation in einer praktischen Ansicht aufrufen.

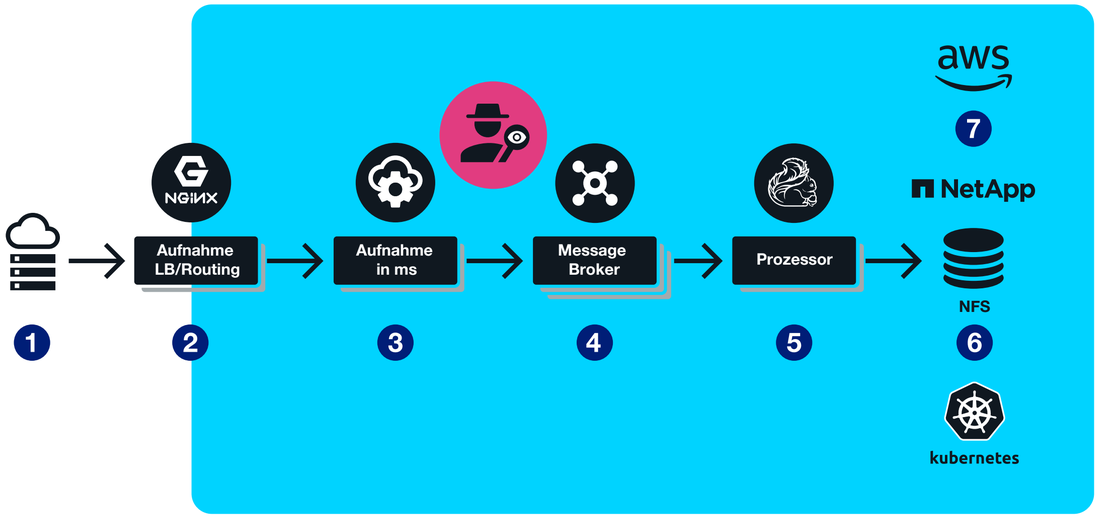


Außerdem möchten wir die CPU und die Speicherauslastung bei den Containern für diese Pods beobachten. In allen diesen Ansichten haben wir unsere Kennzahlen nach `kubernetes_cluster, namespace, pod_name` und container_name gruppiert, um pro Container eine Zeile zu erhalten. So wird es uns sofort auffallen, wenn etwas verdächtig aussieht.

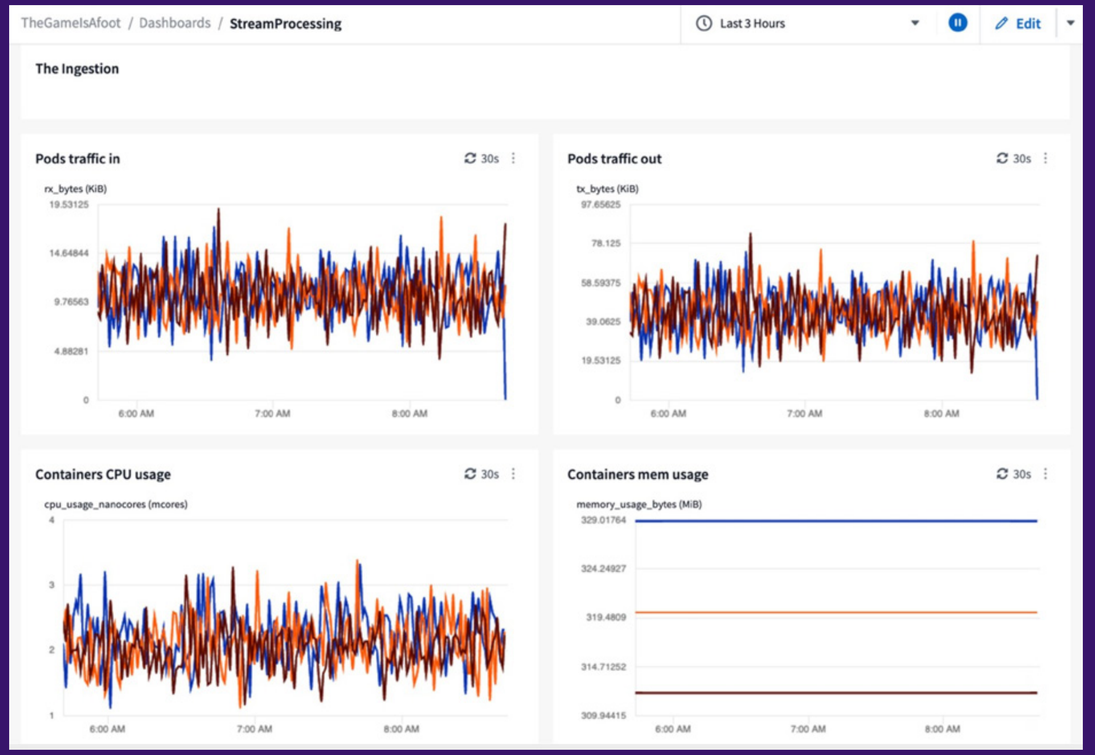


Es sieht so aus, als seien unsere Aufnahmepods frei von jeder Schuld. Es ist zu erkennen, dass die Menge der ein- und ausgehenden Daten den ganzen Tag über konsistent bleibt, und es sind keine wesentlichen Beeinträchtigungen zu sehen. Auch scheint es zu den Zeitpunkten, zu denen die Probleme auftreten, keine Auffälligkeiten bei Arbeitsspeicher oder CPU zu geben.

Der Message Broker: Der Mittelsmann



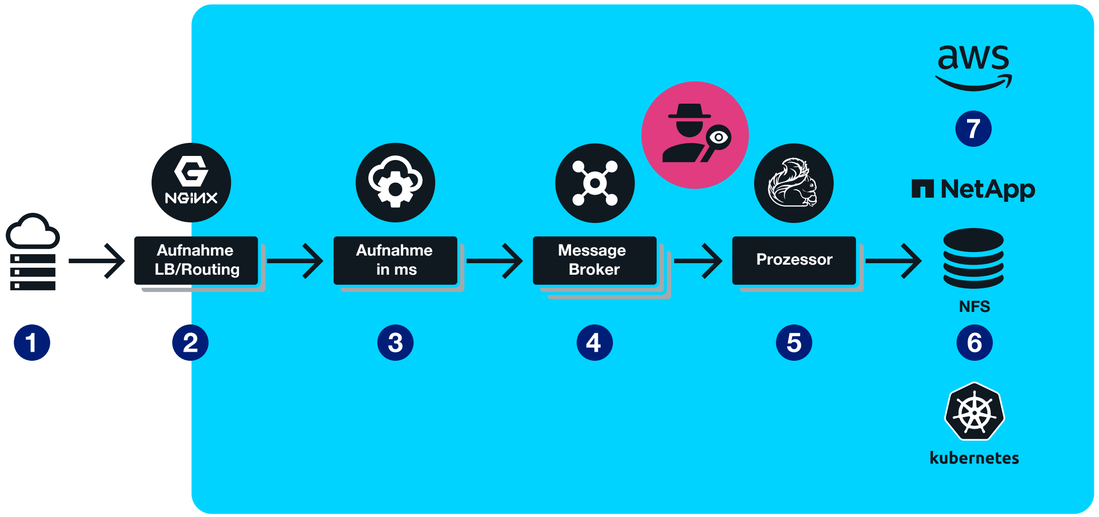
Als Nächstes laden wir unseren Kafka-Broker zum Verhör. Unser Microservice führt eine leichte Verarbeitung eingehender Daten durch und schreibt sie in ein Kafka-Thema. Vielleicht hakt es ja zu den Zeitpunkten, an denen wir eine verlangsamte Verarbeitung verzeichnen, irgendwo bei Kafka? Die entscheidenden Kennzahlen zur Performance von Kafka sehen wir auf dem Dashboard unserer Applikation.



Nun, das ist in der Tat interessant … Es scheint zwar keine wesentlichen Abweichungen bei der Menge der in Kafka eingehenden Daten zu geben, doch zu den Zeitpunkten, an denen sich der Dateneingang verzögert, fällt etwas auf: Es gibt definitiv eine Diskrepanz bei der Menge der Daten, die aus Kafka ausgehen. Wenigstens wissen wir jetzt, dass die bei Kafka eingehenden Daten nicht das Problem sind. Das ist ein anderer Datenpunkt. Die Frage ist also: Warum sehen wir eine Veränderung bei der Menge der Daten, die zu den relevanten Zeiten von Kafka gelesen werden?

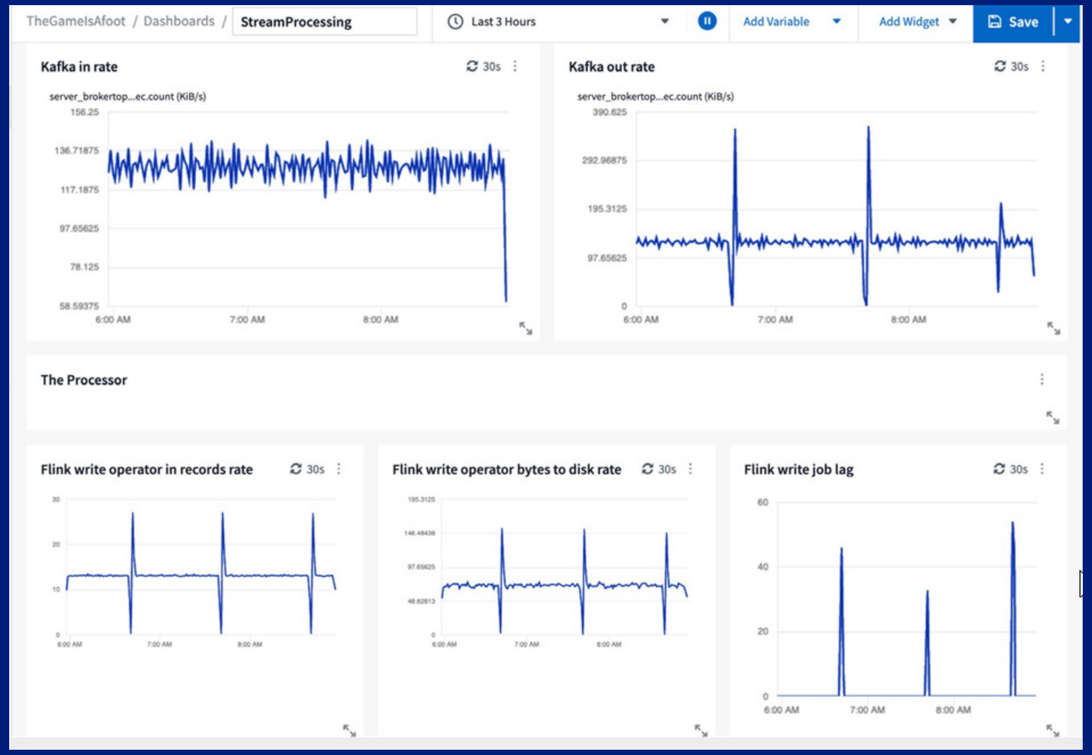
Kafka lässt sich jedenfalls als Verdächtiger ausschließen.

Der Prozessor:
Wir kommen der Sache näher



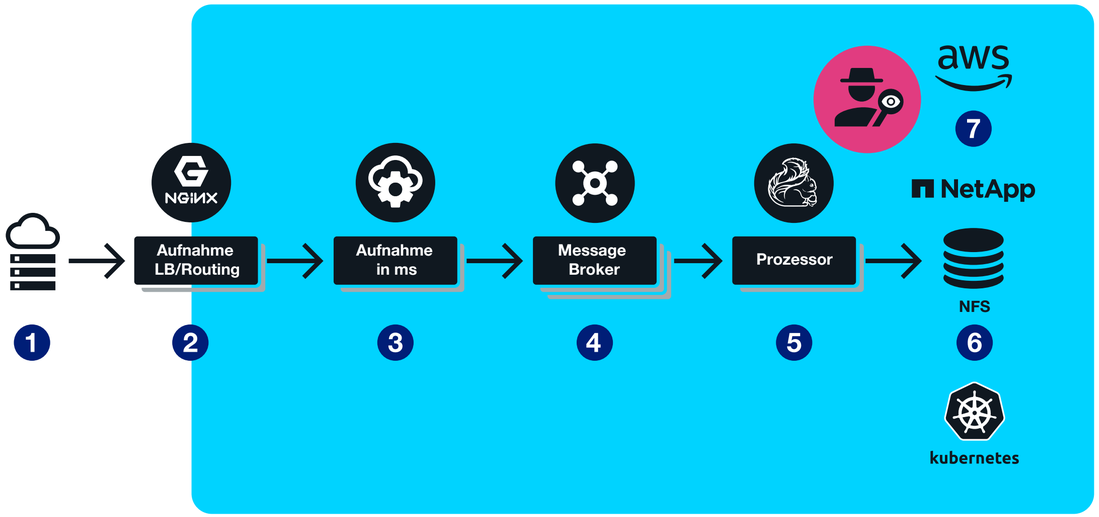
Als Nächstes kommen wir zu unserem Flink-Job. Der Job ist ziemlich gut mit benutzerdefinierten Kennzahlen für das Schreiben auf Festplatte instrumentiert, was uns in Cloud Insights zugutekommt.

Wir haben unserem Dashboard die Anzahl der pro Sekunde aufgenommenen Datensätze für den Flink Operator, der auf Festplatte schreibt, sowie unsere benutzerdefinierten Kennzahlen und die Verzögerung beim Schreibvorgang hinzugefügt. Jetzt wird es interessant.

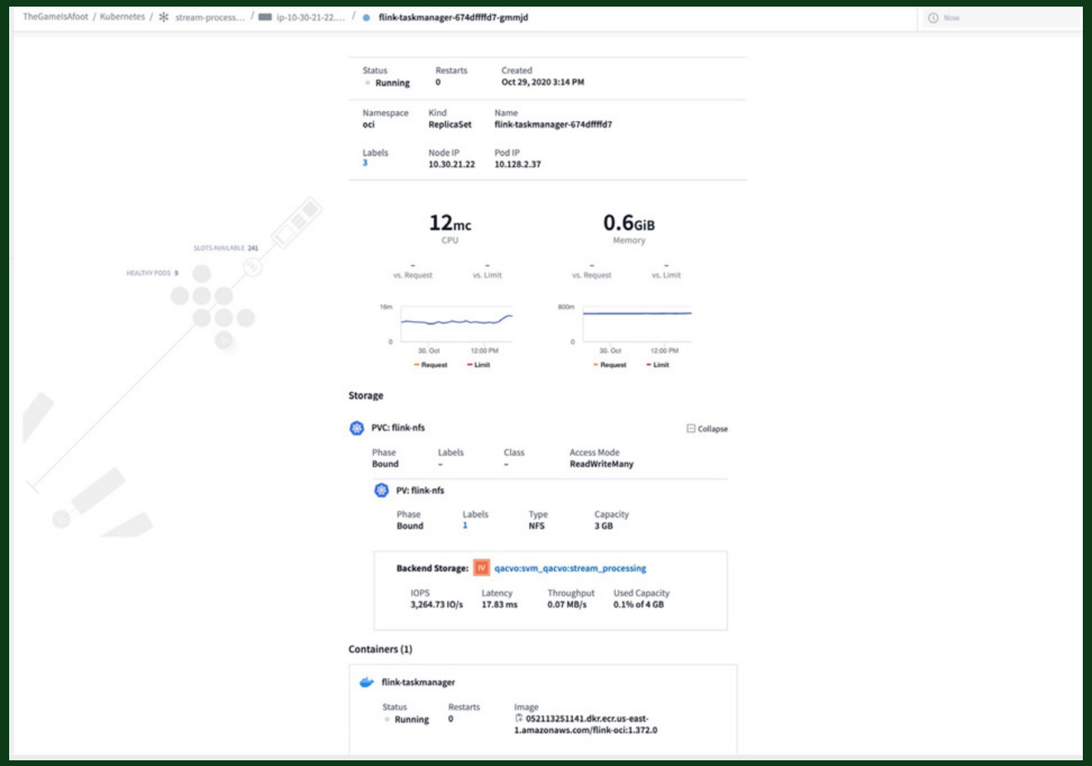


Zum Zeitpunkt der Vorfälle sehen wir einen Einbruch beim Schreibvorgang. Und er stimmt mit dem beim Lesen der Daten von Kafka beobachteten Einbruch überein. Aber wir sehen keine wesentlichen Unterschiede bei den von Kafka aufgenommen Daten. Es scheint also, als würde unser Flink-Operator von irgendetwas aufgehalten werden, wenn er versucht, Daten auf die Festplatte zu schreiben. Könnte das Problem tatsächlich mit dem Schreibvorgang zusammenhängen?

Der Storage:
Es wird spannend



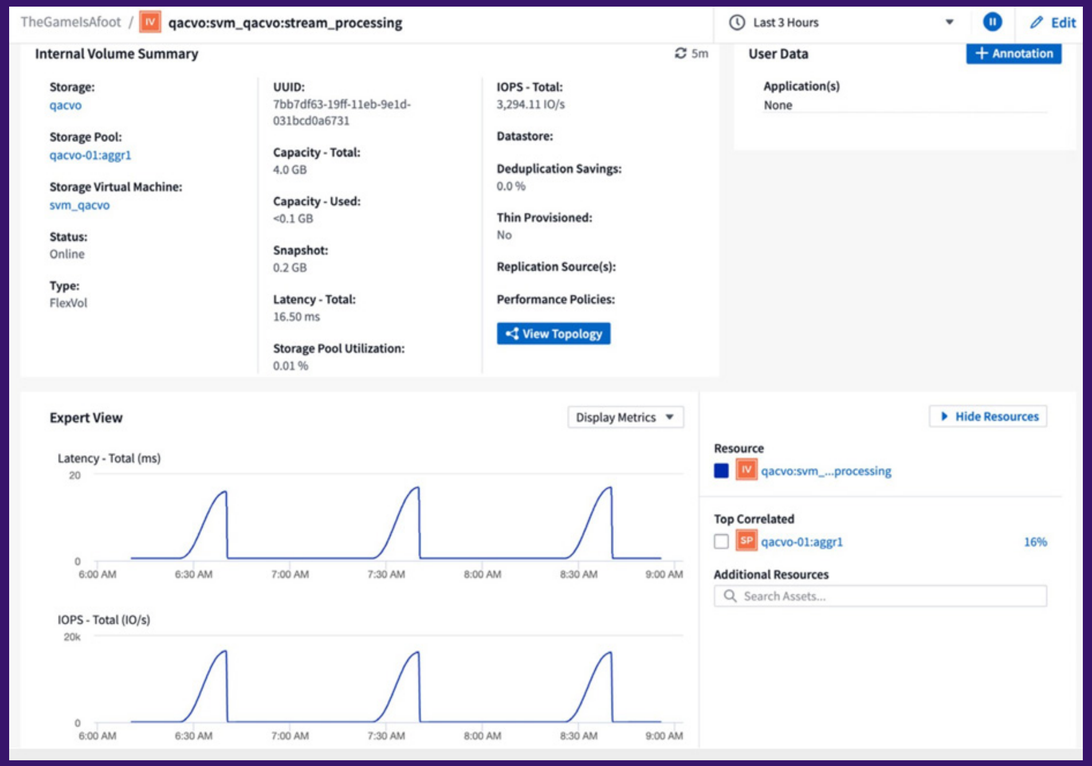
Unser Flink-Job führt das Schreiben in den Storage aus, und die Flink TaskManager-Pods greifen über das übliche Persistent Volume Claim und Persistent Volume Setup auf diesen Storage zu. Wir haben zwar ein paar Schritte gebraucht, bis wir zu dieser Information gelangt sind, doch wie sich herausstellt, waren wir bereits auf der richtigen Spur, als wir uns die Microservice-Pods ansahen und das K8s-Monitoring in Cloud Insights aktivierten. Was uns noch fehlt, ist der tatsächliche Storage hinter den Persistent-Volume-Informationen. Doch das ist in Cloud Insights bekanntlich reine Routine. Werfen wir nun einen Blick auf unseren Storage, NetApp Cloud Volumes ONTAP, der von unserem Flink Verarbeitungs-Pod genutzt wird.



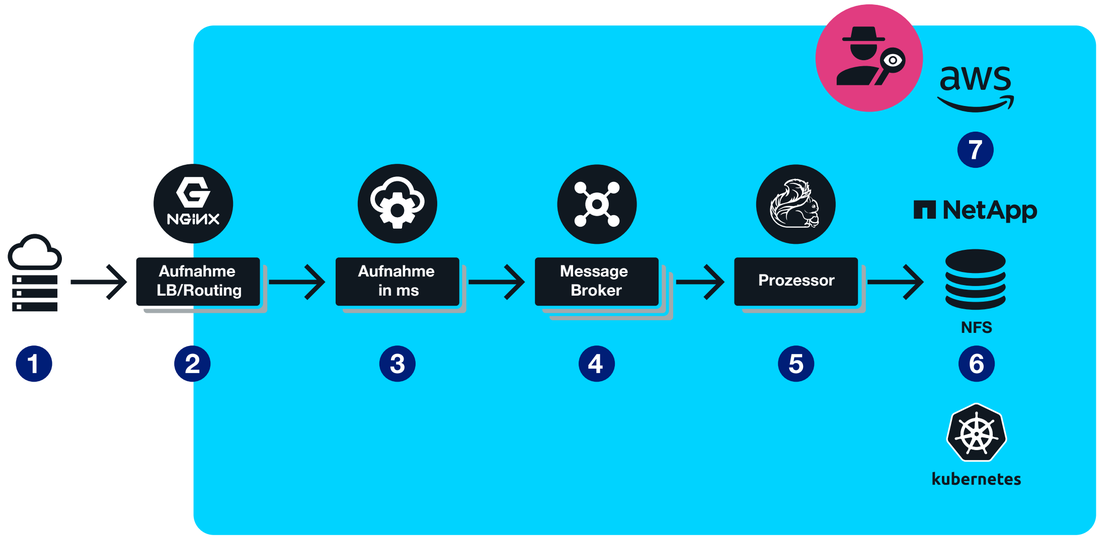
Im Cluster Explorer sehen wir, dass unser persistentes Volume mit dem zugrunde liegenden Storage und einigen damit zusammenhängenden wichtigen Kennzahlen verknüpft ist.

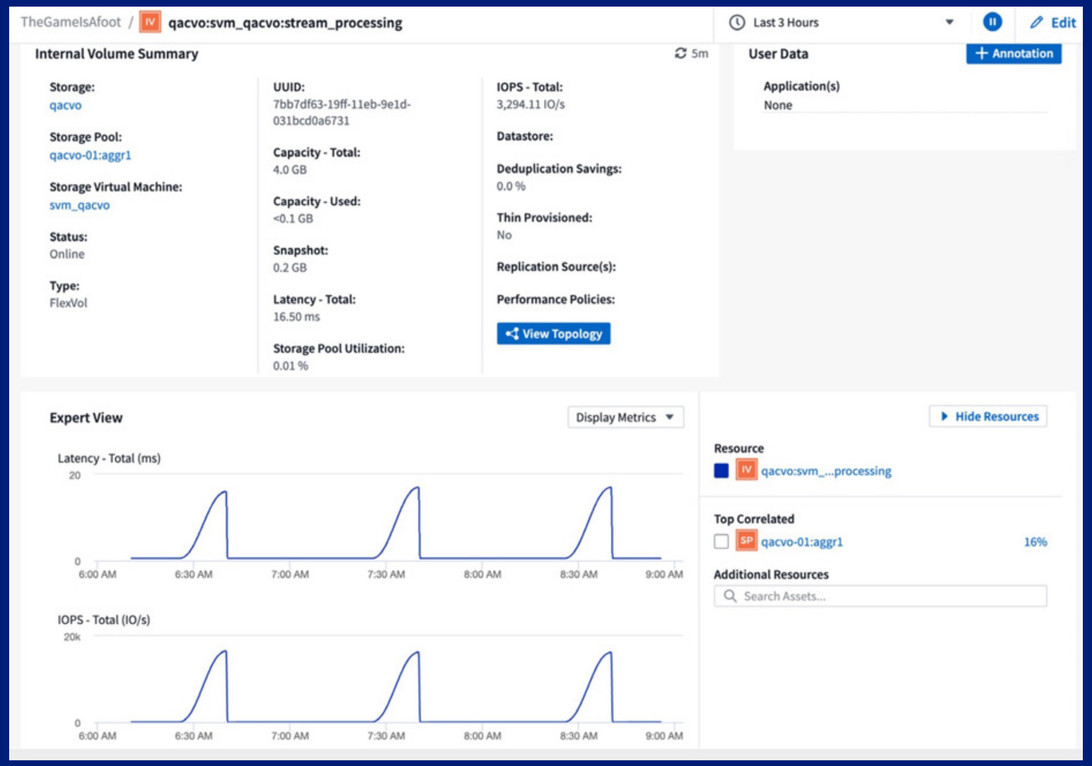
Der Storage:
Es wird spannend

Sind Anhaltspunkte für einen hinreichenden Verdacht zu erkennen? Zum Zeitpunkt eines Vorfalls ist ein I/O-Einbruch und auch ein Latenzeinbruch beim Zugriff auf den Storage zu beobachten. Wir nähern uns des Rätsels Lösung. Doch wo ist die Fehlerquelle? Wer oder was ist für diesen I/O verantwortlich? Unser Pod scheint es nicht zu sein, da wir bereits gesehen haben, dass über den Tag hinweg konsistente Datenmengen eingehen. Es muss etwas anderes ein. Um das Bild noch weiter zu vervollständigen, können wir in derselben konsolidierten Ansicht auch noch die AWS EC2-Informationen prüfen.



Die Berechnung: Was machen unsere AWS EC2 Instanzen?





Wenn wir einen genaueren Blick auf das verknüpfte, von unserem Flink-Job genutzte FlexVol Volume werfen, entdecken wir einen möglichen Schuldigen für den Anstieg beim I/O zum Zeitpunkt des Ereignisses: Es sieht ganz so aus, als würde Virtual Machine frosa-system-backup einen wesentlichen I/O beim Volumen verursachen. Wir klicken weiter und prüfen diese Virtual Machine einmal ganz aus der Nähe.

Wir melden uns also bei dieser EC2-Instanz an und sehen uns dort ein bisschen um. Bei der Analyse der Virtual Machine stoßen wir auf einen Prozess, der jede Stunde etwa in der 40. Minute ausgeführt wird – ein lästiger Cronjob.

```
[root@frosa-system-backup ~]$ crontab -l
40 * * * * /opt/backups/system_backup.sh
[root@frosa-system-backup ~]$
```

Dieser Prozess führt Daten-Dumps aus und generiert so viel I/O, dass die Performance unseres internen Volumes, das auch von unseren Verarbeitungs-Pods genutzt wird, beeinträchtigt wird. Diese Pods sind für geschäftskritische Vorgänge zuständig. Wir müssen diese EC2-Instanz unbedingt fixen, damit sie nicht dasselbe FlexVol Volume hinter dem Kubernetes Persistent Volume unseres Flink-Jobs beschreibt. Wir bitten daher unseren Systemadministrator, diese EC2-Instanz einem anderen FlexVol Volume zuzuweisen.

Die Daten fließen nun reibungslos durch unsere Applikation. Der Fall ist gelöst!

Fazit:
Eigentlich lag es die ganze Zeit auf der Hand

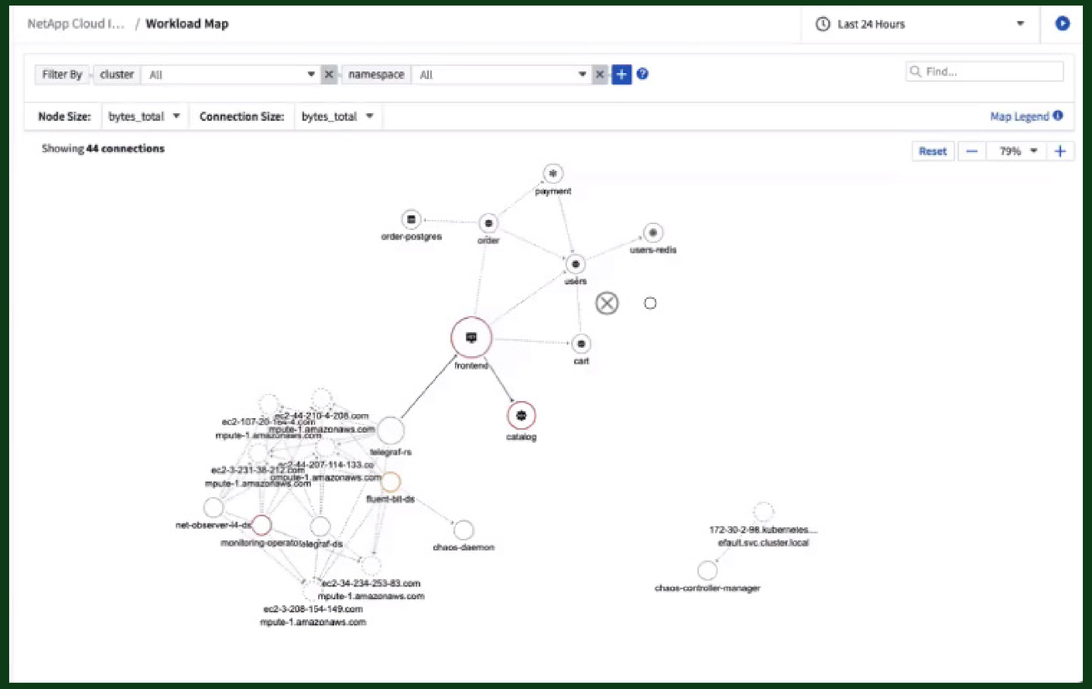
Die typische Kriminalgeschichte präsentiert uns einen überraschenden Täter, den man überhaupt nicht mehr im Blick hatte. In unserer Geschichte ist das nicht anders. Unser Bösewicht wird von Kubernetes-Monitoring-Tools gerne übersehen: Storage. Es stimmt – in der Regel ist es nicht leicht, die Verbindung zwischen Kubernetes-Pods und dem zugrunde liegenden Storage zu erkennen. Wie bei allen Ermittlungen ist es von zentraler Bedeutung, die richtigen Hinweise zu finden und Zusammenhänge herzustellen. Cloud Insights macht diese Hinweise und Zusammenhänge für uns sichtbar. Am Ende unserer Ermittlungen hatten wir uns einen Überblick über die gesamte Infrastruktur verschafft, den wir nutzen konnten, um die Ursache des Problems ausfindig zu machen. Somit verfügen wir nun über eine starke Grundlage, auf der wir weiter aufbauen können, um die Integrität unseres Systems sicherzustellen und Problemquellen zukünftig sehr viel einfacher zu verorten.

Für diese Ermittlung erfassen wir alle Daten für:

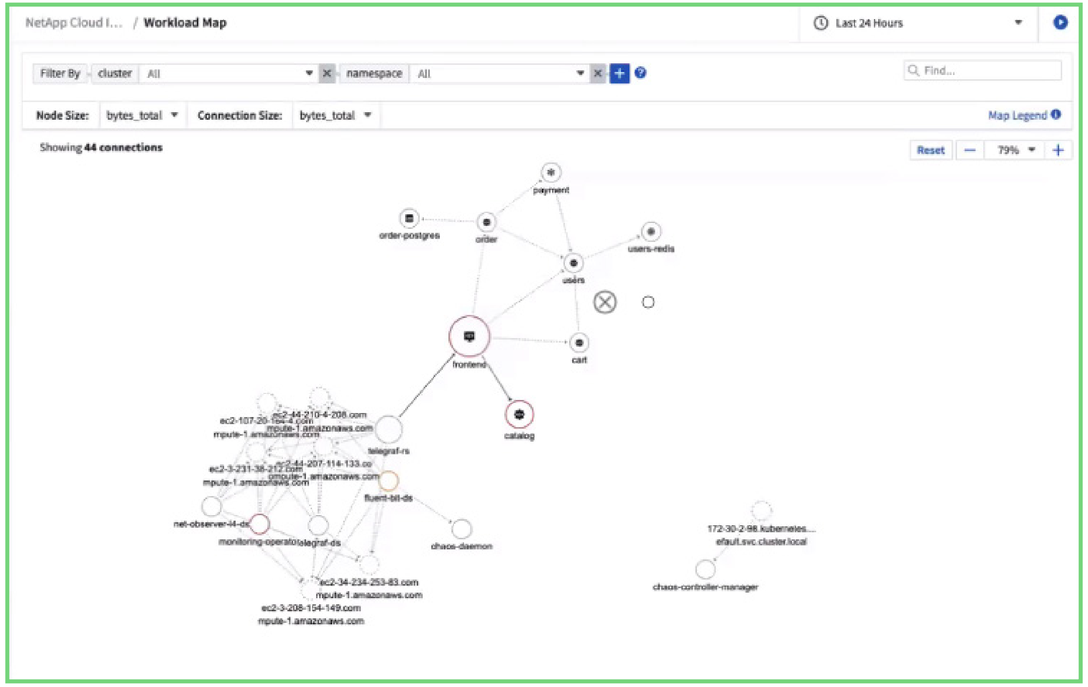
- viele heterogene Infrastruktur- und Servicetypen (NGINX Ingress Controller)
- unsere gesamte Kubernetes Umgebung, einschließlich unseres Aufnahme-Microservices
- Netzwerk-Traffic
- Kafka-Broker und topicsStorage time-to-full
- den Flink-Job und seine Operator
- Persistent Volumes mit zugrunde liegendem NetApp Cloud Volumes ONTAP Storage
- AWS EC2 Virtual Machines

Wir haben alle Hinweise zusammengetragen und konnten so das Rätsel lösen.

Wie in diesem Paper gezeigt, hat Cloud Insights das Monitoring und die Diagnose von Kubernetes-Problemen schon immer erleichtert. Jetzt machen wir es sogar noch einfacher: Unsere neue Netzwerk- und Workload-Übersicht visualisiert Ihre Cluster mitsamt Nodes, Pods, Containern und Storage. Sie sehen alle Abhängigkeiten und können ganz einfach die Quelle von Traffic- oder Latenzproblemen ausfindig machen und betroffene Workloads identifizieren und isolieren.



Ein einzelnes Problem zurückzuverfolgen ist ein Kinderspiel. Aber was ist, wenn Sie plötzlich eine Vielzahl von Alarmen erhalten, die auf eine Fülle von Workload-Problemen deuten? Wo fangen Sie dann an? Genau hier glänzt unsere Workload-Übersicht. In folgendem Beispiel sehen Sie einen einzigen Schuldigen, der von vielen Prozessen genutzt wird und dort zu Fehlern führt. Beheben Sie diesen Fehler, um eine lange Liste von Problemen zu lösen. Und das alles mit nur einem Tool: So geht einfach!



Cloud Insights kann noch viel mehr Informationen liefern – diese Untersuchung kratzt gerade mal an der Oberfläche. Wir können Alarme zu den in Cloud Insights erfassten Komponenten, Kennzahlen und Logs einrichten. Und das ist nur die Spitze dessen, was Sie mit den Informationen machen können.

Überzeugen Sie sich mit einer kostenlosen Testversion. Besuchen Sie die Cloud Insights Registrierungsseite und testen Sie Cloud Insights 30 Tage kostenlos.

Über NetApp

Mehr erfahren Sie unter www.netapp.de.

Über NetApp
In einer Welt voller Generalisten beweist sich NetApp als Spezialist. Wir haben ein Ziel fest im Blick: Ihr Unternehmen darin zu unterstützen, Ihre Daten optimal zu nutzen. NetApp bringt die Datenservices, denen Sie vertrauen, in die Cloud und die Einfachheit und Flexibilität der Cloud in Ihr Datacenter. Selbst bei höchsten Ansprüchen lassen sich die branchenführenden NetApp Lösungen in unterschiedlichsten Kundenumgebungen und den weltweit führenden Public Clouds einsetzen.

Als Cloud- und Daten-orientierter Softwareanbieter stellt nur NetApp alle Technologien bereit, mit denen Sie Ihre eigene maßgeschneiderte Data Fabric aufbauen, Ihre Clouds vereinfachen, Ihre Public Clouds anbinden und so die richtigen Daten, Services und Applikationen sicher bereitstellen können – immer und überall.

NetApp

+49 151 12055761