Inhaltsverzeichnis Seite

Sperrvermerk I

Abbildungsverzeichnis IV

Tabellenverzeichnis V

Abkürzungsverzeichnis VI

1 Einleitung 1

2 Logs und ihre Bestandteile 3

2.1 Log Transport 4

2.2 Log Formate 6

2.3 Log-Syntax 8

2.4 Log Content 9

2.5 Logs Verarbeitung 11

3 Log Files in Salesforce B2C 12

3.1 System sowie Custom Logs 13

3.2 Quota Log 14

3.3 Security Log 15

3.4 Import sowie Export Log 16

3.5 Batch-Processing Log 17

3.6 Andere Arten von Logs in Salesforce B2C 17

4 Log Center 19

4.1 Log Center Dashboard 19

4.2 Log Center Search 19

4.3 Log Streaming 21

5 Grafana Labs 23

5.1 Kern-Grafana-Stack 23

5.2 Grafana Incident Response und Management 24

6 Grafana Loki 25

6.1 Architektur von Grafana Loki 25

6.2 Die Struktur von Logs in Loki 26

6.3 Log-Verarbeitung 27

6.3.1 Log-Stream-Selektor 28

6.3.2 Zeilenfilter 29

6.3.3 Parser 29

6.3.4 Label-Filter 31
6.3.5 Transformation von Logzeilen 32
6.3.6 Erstellung von Metriken aus Logs 33
6.4 Visualisierung von Logdaten 36
7 Aufsetzen des Prototyps 38
8 Vergleich von Log Center und externem Dienst 40
9 Fazit 42
Quellenverzeichnis VIII
Anlagenverzeichnis XI

Abbildungsverzeichnis Seite

Abbildung 1: Formaten von Logs 6
Abbildung 2: System Log 13
Abbildung 3: Custom Log 14
Abbildung 4: Quota Log 15
Abbildung 5: Security Log 16
Abbildung 6: Import Log 16
Abbildung 7: Batch Processing Log 17
Abbildung 8: Der Prozess der Datenübermittlung an Drittanbieter 22
Abbildung 9: High-Level-Architektur von Grafana Loki 25
Abbildung 10: Loki-Logstruktur 27
Abbildung 11: Verwendung des Log-Stream-Selektors 28
Abbildung 12: Suche nach Ausdrücken mithilfe von Zeilenfiltern 29
Abbildung 13: LogQL-Anfrage mit Parser und Label-Filter 32
Abbildung 14: Verwendung von Zeilen- und Label-Format in LogQL-Anfrage 33
Abbildung 15: Verwendung der Logbereichs - Aggregationsfunktion in einer LogQL-Abfrage 34
Abbildung 16: Verwendung des integrierten Aggregationsoperators in einer LogQL-Abfrage 36
Abbildung 17: Generierung eines Zugriffstokens in Grafana Cloud 38
Abbildung 18: Terraform-Setup 38
Abbildung 19: Konfiguration des neuen Log-Streams 39

Tabellenverzeichnis Seite

Tabelle 1: Transportmechanismen von Logs 5
Tabelle 2: Arten von Logs nach Ereignissen 10
Tabelle 3: Andere Arten von Logs in Salesforce B2C 17
Tabelle 4: Zeilenfilter-Operatoren 29
Tabelle 5: Arten von Parsern in Grafana Loki 30
Tabelle 6: Arten von Werten und entsprechende Operatoren von Prädikaten 31
Tabelle 7: Unwrapped-Bereichsaggregationsfunktionen 35
Tabelle 8: Vergleich von Log Center und externem Dienst 40

Abkürzungsverzeichnis

usw.	Und so weiter
z.B.	Zum Beispiel
bzw.	Beziehungsweise
B2C	Business-to-Consumer
SFCC	Salesforce Commerce Cloud
PCI DSS	Payment Card Industry Data Security Standard
HIPAA	Health Insurance Portability and Accountability Act
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
SOAP	Simple Object Access Protocol
HTTP	Hypertext Transfer Protocol
SNMP	Simple Network Management Protocol
FTPS	File Transfer Protocol Secure
TLS	Transport Layer Security
API	Application Programming Interface
SCP	Secure Copy Protocol
FTP	File Transfer Protocol
SSL	Secure Sockets Layer
SCP	Secure Copy Protocol
SDEE	Security Device Event Exchange
IDMEF	Intrusion Detection Message Exchange Format
ELF	W3C Extended Log File Format
CEF	ArcSight Common Event Format
IPS	Intrusion Prevention System
ANSI	American National Standards Institute
RFC	Requests for Comments
WebDAV	Web-based Distributed Authoring and Versioning
PIG	Primary Instance Group
SIG	Secondary Instance Group
AWS	Amazon Web Services
JSON	JavaScript Object Notation
CSV	Comma-separated values

TSDB	Time Series Database
S3	Amazon Simple Storage Service
GCS	Google Cloud Storage
WALs	Write-Ahead Logs
ML	Maschinelles Lernen
LLM	Large Language Model
IRM	Incident Response and Management

1 Einleitung

Fast jede Software, jedes Computersystem und jedes Gerät kann Ereignisse während eines bestimmten Vorgangs aufzeichnen. Diese Einträge werden als Logs bezeichnet und können beispielsweise Fehlermeldungen, Nutzungsdaten oder Systemmessungen umfassen. Sie sind eine wichtige Informationsquelle, deren Nutzung viele Vorteile mit sich bringt, angefangen bei der Minimierung von Problemen in frühen Phasen bis hin zur Aufrechterhaltung eines stabilen Systemzustands[1].

Die Einsatzmöglichkeiten von Logs sind vielfältig: Computerforensik (die Kombination von Informatik und Kriminalistik), Durchführung von Audits, Fehler- sowie Einbruchserkennung usw. E-Commerce ist auch keine Ausnahme. Sicherheitsaspekte und die Aufrechterhaltung der Performance der Website sind sowohl für neue als auch für Stammkunden von entscheidender Bedeutung. Um diese Anforderungen zu erfüllen, ist die Verwendung von Aufzeichnungen unverzichtbar. Die Logs können dabei helfen, langsam ladende Seiten, Datenbankfehler, Serverprobleme, Ausfälle von Warenkörben und Autorisierungen zu erkennen, die Cyberbedrohungen zu verhindern, die suspekter Aktivitäten und Anomalien zu identifizieren sowie die Transaktionssicherheit zu steigern. Als Ergebnis werden Benutzererfahrungen verbessert und die Loyalität und das Vertrauen der Kunden steigen[2].

Salesforce B2C ist einer der Leistungsbereiche von dotSource SE. Deshalb ist die Verwendung von Logs ein wesentlicher Bestandteil der Erstellung und Betreuung eines Online-Shops. Das Problem besteht jedoch darin, dass die Anzahl der generierten Protokolle innerhalb eines Tages mehrere Zehntausend erreichen kann. Einige davon enthalten nur rein informative Daten, andere hingegen relevante Fehler- und Systeminformationen, die unverzüglich behoben werden müssen. Dazu muss es möglich sein, eine große Anzahl von Protokollen effektiv auszuwerten. Aus diesem Grund ist das Ziel der Arbeit, zu untersuchen, wie Logs an einen externen Dienst übermittelt werden können, um dort eine erweiterte Analyse durchzuführen und somit Sicherheit, Systemstabilität sowie Kostensenkung zu gewährleisten.

Im Rahmen dieser Arbeit werden allgemeine Log-Informationen sowie Protokolltypen in Salesforce B2C, das Log Center (eine Anwendung zur Suche nach Logmeldungen), seine Hauptkomponenten sowie die Integration mit Drittanbieterprogrammen untersucht. Besondere Aufmerksamkeit wird dem Dienst gewidmet, in dem die Logs analysiert werden, seiner Architektur

[1] Vgl. [Rüc22], S. 1
[2] Vgl. [JP26]

und den Möglichkeiten der Datenvisualisierung. Als praktisches Ergebnis wird ein Prototyp entwickelt und getestet.

2 Logs und ihre Bestandteile

Der Begriff „Log“ hat in den Bereichen, in welchen er verwendet wird, eine umfangreiche Bedeutung. Für Entwickler hat er eine eigene Definition, die erst nach der Erläuterung anderer Konzepte vollständig verstanden werden kann:

- Event ist ein einzelnes Vorkommnis innerhalb einer Umgebung. Es erläutert den Zeitpunkt des Entstehens sowie Details zum Ereignis oder zur Umgebung, die zur Klärung der Ursachen leisten könnten;
- Ereignisfeld ist eine bestimmte Eigenschaft eines Ereignisses, z.B. Zeitstempel, IP, Benutzeridentifikation usw.;
- Ereignisaufzeichnung ist die Gesamtheit von Ereignisfeldern, die ein Ereignis beschreiben[3].

Letztendlich bedeutet das Log die Gesamtheit der Ereignisaufzeichnungen. Darüber hinaus sollte noch ein wichtiges Konzept berücksichtigt werden: das Logging. Dabei handelt es sich um einen Prozess zur Erfassung von Ereignisaufzeichnungen in Logs, wobei sie entweder in Textdatei oder als Auditdaten in Binärdateien bzw. Datenbanken gespeichert werden[4].

Es existieren insgesamt vier Typen von Logging, die sich nach ihrer Funktionsweise klassifizieren lassen – Sicherheits-, Betriebs-, Compliance- und Anwendungs-Debug-Protokollierung. Fast alle Logquellen unterstützen diese Arten, jedoch sind ihre Analyse sowie Konsums unterschiedlich[5].

Bei dem ersten Typ handelt es sich um eine Sicherheitsprotokollierung, die für die Sicherheit von Systemen verantwortlich ist. Dazu gehören die Erkennung von Cyberangriffen, Virusinfektionen, Datendiebstählen sowie Maßnahmen zu ihrer Neutralisierung. Das einfachste Beispiel ist die Protokollierung der Benutzerauthentifizierung und seine Zugriffsrechte auf bestimmte Ressourcen[6].

Eine umfangreiche Kategorie von Logging ist die Betriebsprotokollierung. Sie umfasst eine große Anzahl von Typen der Logs sowie enthält nützliche Informationen über den Systemzustand. Sie stellen z.B. die Nachrichten über Systemausfälle oder potenziell unsichere

[3] Vgl. [CSP13], S. 29 – 32ff.
[4] Ebenda
[5] Ebenda
[6] Vgl. ebenda

Bedingungen bereit, die dann von Systembetreibern verwendet werden. Dazu kommt noch, dass diese Art von Protokollierung für die Erbringung von Dienstleistungen und finanzielle Entscheidungen genutzt werden kann[7].

Die Compliance-Protokollierung ist bedeutend, da die von Unternehmen festgelegten Anforderungen zur Gewährleistung der Sicherheit heutzutage obligatorisch sind. Es lassen sich zwei Arten von Compliance-Logging unterscheiden. Erstens sind Vorschriften sowie Mandate, die Einfluss auf IT-Operationen haben. Die Beispiele hierfür sind PCI DSS (Payment Card Industry Data Security Standard), HIPAA (Health Insurance Portability and Accountability Act), ISO (International Organization for Standardization), ITIL (Information Technology Infrastructure Library) usw. Zweitens sind Regelungen und Aufträge, die das Systemdesign sowie die Sicherheit festlegen, beispielsweise die Common Criteria[8].

Letztendlich kommt die Debug-Protokollierung der Anwendung. Diese Art von Logging ist besonders brauchbar für System- oder Softwareentwickler und wird fast immer in der Produktionsphase nicht eingesetzt. Um mögliche Fehler oder Schwachstellen mithilfe dieser Art der Protokollierung zu finden, müssen die Architektur der App, ihre Funktionen und die Bestandteile des Codes gut bekannt sein[9].

Es ist zweckmäßig, die unterschiedlichen Arten des Loggings für eine leistungsfähige Auswertung zu wissen. Der nächste, aber nicht weniger wichtige Aspekt sind die Kenntnisse der grundlegenden Mechanismen der Protokollierung. Dazu gehören Transport, Syntax sowie Format, die Event-Taxonomie von Logs und Einstellungen sowie Konfigurationen von Logging[10].

2.1 Log Transport

Der Transport von Logs ist ein Prozess, der die Verfahren zum Verschieben von Protokollmeldungen von einem Ort zu einem anderen beschreibt. Es existieren verschiedene Transportmechanismen von Logs. Die bekanntesten davon sind: syslog UDP, syslog TCP, encrypted syslog, SOAP oder HTTP, SNMP, SCP und FTPS (siehe Tabelle 1)[11].

[7] Vgl. [CSP13], S. 32f.
[8] Ebenda
[9] Vgl. ebenda
[10] Ebenda
[11] Vgl. [CSP13], S. 35f.

Tabelle 1: Transportmechanismen von Logs

Transportmechanismen	Beschreibung
syslog UDP	Das ist ein verbindungsloses und effizientes Protokoll, das eine schnelle Übertragung von Logs gewährleistet und mit großen Datenmengen arbeitet. Es muss berücksichtigt werden, dass Daten verloren gehen können. Deshalb eignet sich dieses Verfahren gut für unkritische Informationen, beispielsweise Metriken zur Netzwerkleistung, Statistiken zur Ressourcennutzung, Statusaktualisierungen, Dashboards zur Echtzeitüberwachung[12].
syslog TCP	Ein Mechanismus, der eine Verbindung zwischen dem syslog-Client und dem Server herstellt. Er gewährleistet die Übertragung wichtiger Daten, die Entfernung von Duplikaten und das erneute Senden verlorener Pakete. Im Gegenteil zu UPD verbraucht er jedoch mehr Ressourcen, sodass der Prozess bei hoher Auslastung langsamer werden kann. Am besten ist er für die Authentifizierungs- und Sicherheitsprotokolle, Finanztransaktionen, Daten zur Compliance, Warnmeldungen bei Systemausfällen geeignet[13].
encrypted syslog (mit TLS)	Ein Verfahren, das mithilfe von Zertifikaten sowie öffentlichen/privaten Schlüssel eine sichere Verbindung zwischen Sender und Empfänger ermöglicht und Angriffe von unbekannten Benutzern verhindert. Das bietet eine Reihe von Sicherheitsvorteilen, bzw. Syslog-Meldungen werden während der Übertragung verschlüsselt. Die Authentifizierung erfolgt auf beiden Seiten: sowohl beim Absender als auch beim Empfänger. Auf diese Weise wird sichergestellt, dass der Receiver kennt, mit wem er kommuniziert. Demgegenüber kann der Sender überprüfen, ob der Empfänger der erwartete ist. Allerdings kann es mittels dieses Verfahrens nicht kontrolliert werden, ob die Nachricht zuverlässig und sicher ist[14].
HTTP	Ein Internetprotokoll, das die Verbindung zwischen Server und Client und Datenaustausch zwischen ihnen ermöglicht. Die Verwendung dieses Verfahrens zum Log-Transport bietet eine Reihe von Vorzügen, z.B. die Effizienzsteigerung. Viele externe Analysedienste wie Elasticsearch, Splunk, Grafana haben eine integrierte HTTP-basierte API zum Sammeln von Logs. Daher können die Protokolle einfach über HTTP an mehrere Anwendungen übertragen und dort analysiert werden[15].
FTPS	Ein standardmäßiges Protokoll, das die Übertragung von Files zwischen Client und Server sicherstellt. Es handelt sich um eine verbesserte Version von FTP, die eine SSL/TSL Schicht unter FTP enthält, um Datenkanäle zu verschlüsseln. FTPS ermöglicht eine starke Authentifizierung, eine hohe Sicherheitsstufe und X.509 Zertifikat. Allerdings kann es die Arbeit von Firewalls behindern[16].
SCP	Ein Protokoll, das Dateien zwischen lokalen Hosts und Remote-Host (oder zwei Remote-Hosts) mittels eines verschlüsselten IP-basierten Datentunnels zu übertragen. Das garantiert Geschwindigkeit sowie Sicherheit[17].

[12] Vgl. [Log26]
[13] Ebenda
[14] Vgl. [Rsy26]
[15] Vgl. [Sys26]
[16] Vgl. [Int26]
[17] Ebenda

Der Transport ist ein wichtiger Teil von Logging. Zu seinen Aufgaben gehören die Bewahrung von Integrität, Nutzbarkeit und Datenschutz der Logdaten. Zudem müssen das Format, die Reihenfolge der Einträge, die Ereignisdetails, das Datum und der Zeitstempel von Logs während der Übertragung nicht geändert werden und genauso bleiben, wie zu Beginn der Übermittlung[18].

2.2 Log Formate

Nach der Übertragung von Logs, soll nun deren Format und Syntax im Fokus stehen. Sie legen fest, wie Aufzeichnungen gebildet, übermittelt, gespeichert, bewertet und analysiert werden[19].

Es existiert keine Unterteilung von Protokollformaten nach bestimmten Eigenschaften. Allerdings ist es möglich, sie in drei Hauptgruppen zu gliedern (siehe Abbildung 1). Zur ersten gehören Binär-, Text-, XML- und relationales Datenbankformat. Die zweite Gruppe enthält Protokolle mit offenem sowie proprietärem Format. Die letzte umfasst W3C Extended Log File Format (ELF), Apache access log, Cisco SDEE/CIDEE, ArcSight Common Event Format (CEF), Syslog, IDMEF – weitere Formate von Logs.

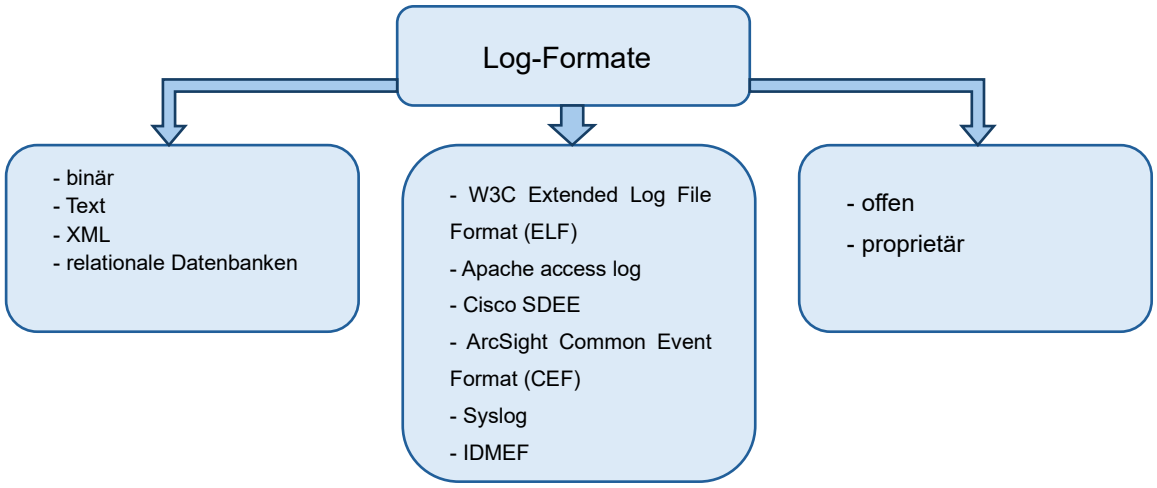


Abbildung 1: Formaten von Logs
Quelle: Eigene Darstellung nach [CSP13], S. 36 - 40

Zunächst wird die erste Kategorie der Protokollformate betrachtet. Binärdateien sind praktisch in der Verwendung, da sie normalerweise kleiner als andere Formate sind und geringeren Ressourcenbedarf für die Verarbeitung durch das System benötigen. Das ermöglicht den

[18] Vgl. [CSP13], S. 35f.
[19] Ebenda

Einsatz von knappem Speicherplatz auf der Festplatte sowie eine reduzierte Verarbeitung von Ein-/Ausgabe. Zusätzlich sind die Felder und Datentypen von binären Logs deutlich definiert, sodass das Log-Parsing weniger aufwendig ist. Deshalb wird die Analyse von Protokollen effizienter. Typische Beispiele für dieses Format sind das Windows Event Log, die Unix wtmp Datei (enthält Aufzeichnungen von Login) sowie pacct (enthält Buchhaltungsdaten)[20].

Als Nächstes kommt das XML-Format der Logs. Hauptsächlich ist es für maschinelles Lesen geeignet und wird verwendet, wenn eine große Anzahl von Strukturinformationen vom Sender zum Empfänger transportiert und dort analysiert werden muss. Allgemein finden sie in der Sicherheitsprotokollierung Einsatz. Beispiel für ein System, das diese Art von Logs zur Aufrechterhaltung der Betriebssicherheit verwendet, ist Cisco IPS serucity appliance. Schließlich zeigen die XML-Logs oft relative geringe Leistungsfähigkeit[21].

Das Textformat wird in der Regel beim Debuggen benutzt. Logs dieses Typs sind ein wichtiges Hilfsmittel für die Fehlersuche in Anwendungen. Sie werden meistens vom Entwickler verwendet und analysiert, wenn es nötig ist (standardmäßig sind Debug-Protokolle ausgeschaltet). Für eine verbesserte Auswertung können sie mit anderen Protokollformaten kombiniert werden[22].

Schließlich wird das relationale Datenbankformat in Betracht gezogen. Die Protokolle dieser Art stellen eine Tabelle mit binären Einträgen dar. Das Format erlaubt Operationen wie Addition („insertion“) sowie Abfrage („selection“). Diese Vorgehensweise ermöglicht eine effektive Arbeit mit Daten. Das gilt sowohl für die Suche nach einem Wert in einer Tabelle als auch nach einer Tabelle in der gesamten Datenbank[23].

Die nächste Gruppe unterscheidet Logs nach ihrem Format hinsichtlich ihrer Zugänglichkeit für einen breiten Nutzerkreis. Das Offene ist für alle Netzwerkbenutzer verfügbar. Es kann in einem Standarddokument (z.B. ISO, ANSI, Internet Standard) oder in der Referenzdokumentation (z.B. RFC) abgelegt werden. Dagegen ist das proprietäre Format in der Regel nicht veröffentlicht, und nur die Besitzer der Entwicklung haben Zugriff auf die Dokumentation[24].

[20] Vgl. [CSP13], S. 38 - 44
[21] Ebenda
[22] Vgl. [CSP13], S. 44f.
[23] Vgl. [CSP13], S. 39 - 40
[24] Ebenda

2.3 Log-Syntax

Eine wesentliche Komponente des Loggings, die alle Protokolle aufweisen, ist die Syntax. Ihre Funktionsweise besteht darin, die Struktur der Aufzeichnungsmeldungen festzulegen. Obwohl die Syntax für Logs unterschiedlicher Typen und Formate variieren kann, können die am häufigsten verwendeten Felder trotzdem identifiziert werden. Darunter fallen Datum und Zeitstempel (inklusive Zeitzone), Typ des Protokolleintrags, System, Anwendung oder Komponente, die Benachrichtigungen generiert haben, Ausfallanzeige, Schweregrad, Priorität oder Wichtigkeit einer Protokollmeldung und Benutzername. Die Syntax ist entscheidend. Je besser sie verstanden wird, desto schneller kann ein Problem gefunden und behoben sowie eine effiziente Datenanalyse durchgeführt werden. Da die Logs von einer ganzen Reihe von Geräten, Systemen, Anwendungen und anderen Mitteln generiert werden, wurde ein Verfahren „Nachrichtenschema“ erstellt. Dies hilft dabei, Protokollmeldungen zu verstehen. Die Grundidee besteht darin, die Nachricht in die oben genannten Felder zu unterteilen und jedes davon schrittweise zu analysieren[25].

Abhängig davon, was mit dem System passiert ist, können die Meldungen der Logs in folgende Kategorien unterteilt werden (nach dem Prinzip des Schweregrads):

- Informational (informativ) – Meldungen, die über harmlose Ereignisse im System berichten, beispielsweise wenn das System neu gestartet wurde;
- Debug (Debuggen) – Nachrichten, die vom System während der Ausführung des Codes generiert werden, um Entwicklern dabei zu helfen, Fehler zu beheben;
- Warning (Warnung) – Mitteilungen, die darüber benachrichtigen, dass etwas fehlt oder für das Programm benötigt wird. Allerdings hat dies keinen Einfluss auf das System, sodass es weiter funktionieren kann;
- Error (Fehler) – Nachrichten, die über Fehler auf verschiedenen Systemebenen berichten, z.B. Fehler beim Synchronisieren des Puffers mit der Festplatte;
- Alert (Alarm) – Benachrichtigungen, die darauf aufmerksam machen, dass etwas Ungewöhnliches passiert. Normalerweise sind sie eng mit Sicherheitsvorrichtungen sowie Sicherheitsaspekte verbunden. Sie können zum Beispiel über eine böswillige Verbindung zum Netzwerk informieren[26].

[25] Vgl. [CSP13], S. 40 - 43
[26] Vgl. [CSP13], S. 3f.

Dazu kommt noch, dass einige Logs nach ihrer Wichtigkeit geordnet werden können. Diese Klassifizierung erleichtert die Priorisierung von Aufgaben oder Problemen. Dazu zählen niedrige, mittlere sowie hohe Priorisierung. Hier liegt der Unterschied zwischen ihnen:

- Low (niedrig): Normalerweise enthalten Logs mit niedriger Priorität informative Mitteilungen. Außerhalb erfordern sie keine Lösung, sondern dienen ausschließlich als die Quelle der Information;
- Medium (mittler): Medium priorisierte Logs melden, dass das Ereignis, das diese Meldung verursacht hat, in naher Zukunft beachtet werden sollte. Allerdings brauchen sie keine sofortige Lösung.
- High (hoch): Die Logs mit hoher Dringlichkeit müssen sofort berücksichtigt und behoben werden[27].

2.4 Log Content

Die Anwendungsmöglichkeiten von Logs sind vielfältig. Sie umfassen Informationen zu den unterschiedlichen Komponenten des Systems. Dazu gehören Daten zur Benutzeraktivität (beispielsweise welche Benutzer sich angemeldet haben, ihre Aktivitäten im System usw.), Informationen zu bestehenden sowie möglichen Fehlern und Beschädigungen (z.B. Festplattenfehler), der Systemstatus, die Ressourcenauslastung und -leistung. Darüber hinaus können sie über Angriffsversuche berichten und angeben, ob er erfolgreich waren oder nicht. Alle diese Ereignisse werden in den Protokollen beschrieben. Aus diesem Grund ist der Kontext ein wichtiger Bestandteil für das Verständnis von Benachrichtigungen[28].

Der Inhalt der Aufzeichnungen ist eng mit dem Begriff der Taxonomie von Logs verbunden. Sie spielt eine bedeutende Rolle bei der Klassifizierung von Protokollen nach ihren Vorkommnissen. Die Taxonomie ist eine Gesamtheit bestimmter Wörter in einer vorgegebenen Reihenfolge, die den Typ der Aktivität kennzeichnet, zu dem die Logs gehören. Die Ausdrücke beschreiben die Art der Tätigkeit, die Teilnehmer, das Ergebnis sowie zusätzliche Daten[29]. Tabelle 2 zeigt und erläutert die grundlegende Klassifizierung von Logs nach Ereignissen.

[27] Vgl. [CSP13], S. 10
[28] Vgl. [CSP13], S. 44-46
[29] Ebenda

Tabelle 2: Arten von Logs nach Ereignissen
Quelle: [CSP13], S. 45 – 46

Typ nach Ereignis	Beschreibung
Change-Management	Die Aufzeichnungen dokumentieren alle Änderungen des Systems, der Komponenten, der Updates, des Kontos usw. Sie können normalerweise nach Operationen wie Hinzufügen, Entfernen, Aktualisieren oder Ändern unterteilt werden. Abgesehen davon finden sie meist Betriebs- und Sicherheitseinsatz.
Authentifizierung und Autorisierung	Protokolle erfassen die Ergebnisse der Authentifizierung sowie Autorisierung von Benutzern und dokumentieren zusätzlich, ob sie erfolgreich oder misslungen waren. Regelmäßig sind sie mit Sicherheitsanwendung verbunden, da die Überwachung solcher Aktivitäten ein wichtiger Aspekt der Absicherung ist. Darüber hinaus könnten sie Einsatzmöglichkeiten in der operativen Tätigkeit finden, z.B. Nachverfolgung von Benutzern, die ein bestimmtes System genutzt haben.
Daten- und Systemzugriff	Die Protokolle umfassen den Zugriff auf Anwendungskomponenten oder Daten (z.B. Dateien sowie Datenbanken). Unter bestimmten Umständen sind diese Meldungen nicht immer eingeschaltet und werden in der Regel in einer unsensiblen Umgebung generiert. Im Allgemeinen werden sie zur Aufrechterhaltung der Sicherheit sowie zur Steigerung der Produktivität und Betriebseffizienz eingesetzt.
Bedrohungsmanagement	Diese Protokolle warnen vor unbefugtem Zugriff oder anderen sicherheitsgefährdenden Aktivitäten und werden von Netzwerkgeräten mit Sicherheitsfunktionen generiert.
Leistungs- und Kapazitätsmanagement	Dies ist eine breite Kategorie von Logs, die Schwellenwerte, Speicher- und Rechenleistungsnutzung sowie Ressourcennutzung umfasst. Im Wesentlichen werden sie zur Überwachung der Leistung und Kapazität eingesetzt und finden Anwendung im Betrieb und in der Sicherheit.
Geschäftskontinuität und Verfügbarkeitsmanagement	Das ist auch eine Kategorie von Logs mit unterschiedlichen Verwendungszwecken. Beispiele hierfür sind das Herunterfahren oder Starten des Systems, Backups, die Redundanz des Systems oder die Nutzung von Funktionen zur Geschäftskontinuität. In der Regel handelt es sich dabei um Logs für den Betriebseinsatz.
Verschiedene Fehler und Ausfälle	Diese Protokolle enthalten Informationen zu anderen Typen von Systemfehlern, die sich von den oben genannten unterscheiden. Sie sind keine kritischen Betriebsmeldungen und können das Einwirken des Geräteadministrators benötigen.
Debugging-Meldungen	Diese Art von Meldungen wird vom Entwickler verwendet und ist regelmäßig in der Produktionsumgebung nicht aktiv.

2.5 Logs Verarbeitung

Logs enthalten eine enorme Menge an Informationen. Aus diesem Grund müssen sie bearbeitet werden, um wertvolle Daten zu extrahieren. Die Verarbeitung von Protokollen umfasst zwei Hauptprozesse: Parsing und Mining. Der erste sorgt dafür, dass das Log so umgewandelt wird, dass es für die weitere Analyse genutzt werden kann. Daraufhin wird das Mining durchgeführt, dessen Aufgaben Textanalyse, Mustererkennung und Trendidentifizierung sind [30].

Das Parsing von Logs ermöglicht die Transformation unstrukturierter oder halbstrukturierter Daten in ein Format (in der Regel in tabellarische Strukturen), das für eine effektive Untersuchung geeignet ist. Es existieren drei Haupttypen des Parsings: regelbasiert, quellcodebasiert und auf Mining der Daten basierend[31].

Das regelbasierte Verfahren verwendet reguläre Ausdrücke, um bestimmte Protokollereignisse zu identifizieren. Die Genauigkeit regulärer Ausdrücke sowie die Stabilität der Strukturen von Protokollmeldungen beeinflussen die Qualität des Ergebnisses erheblich. Da die Anzahl der Log-Meldungen riesig sein kann und die Ausdrücke genau definiert sein müssen, ist dieser Prozess zeitaufwändig und komplex. Das quellcodebasierte Parsing definiert die Protokollvorlagen aus dem Quellcode, da die Anweisungen bereits darin hinterlegt sind. Daher werden reguläre Ausdrücke, die auf Vorlagen basieren, automatisch generiert und die entsprechende Information extrahiert. Der letzte Typ ist das Parsing, das auf Mining von Daten basiert. Es ermöglicht das automatische Extrahieren von festen Daten (Vorlagen) sowie variablen Teilen (Parametern). Zu seinen Technologien gehören die Suche nach Graphen, die Erkennung von Klonen, die evolutionäre Suche und andere[32].

Mit der Entwicklung der künstlichen Intelligenz wird auch das Parsing auf Basis von LLM (Large Language Model) möglich. Dieses Modell kann verschiedene Logformate verarbeiten. Zusätzlich dazu kann es komplexe Protokolleinträge mit mehrstufigen, verschachtelten Strukturen erkennen und analysieren. Dies ist möglich, da diese Maschinen den Kontext gut verstehen können. Ein wichtiger Vorteil ist, dass es neue Protokollvorlagen rasch lernen kann. Dazu sind nur wenige Beispiele erforderlich, sodass es sich schnell an neue Templates und Formate anpassen kann. Darüber hinaus wird es häufig zur Erkennung von Anomalien eingesetzt[33].

---

[30] Vgl. [Rüc22], S. 2, 8f.
[31] Vgl. [Rüc22], S. 19 - 22
[32] Vgl. ebenda
[33] Vgl. [Lee24], S. 3f.

3 Log Files in Salesforce B2C

Die Datenquellen der Logs sind unterschiedlich. Daher ist auch Salesforce B2C keine Ausnahme. Es enthält verschiedene Arten von Protokolldateien. Dazu zählen Systemprotokolle, benutzerdefinierte Protokolle, Import- und Exportprotokolle, Quota-Protokolle, Sicherheitsprotokolle, Batch-Verarbeitungsprotokolle usw. Diese Typen von Logs können im Business Manager gefunden werden[34] – dem Steuerungszentrum, das bei der Verwaltung aller Aspekte des E-Commerce unterschützt[35]. Eine weitere Möglichkeit, auf die Protokolle zuzugreifen, ist die Verwendung von WebDAV. Jedoch ist es nur mit entsprechenden Berechtigungen zulässig[36].

In der Regel erstellen Produktions- und Staging-Instanzen täglich Logs, die bis zu 30 Tage lang gespeichert werden können. Eine Ausnahme bilden die Sicherheitsprotokolle, die bis zu 90 Tage lang aufbewahrt werden. Drei Tage nach ihrer Erstellung werden die Aufzeichnungen in das Verzeichnis „log_archive“ verschoben. Zudem sind die Dateien dort komprimiert. Dies soll Probleme mit der Leistungsfähigkeit, Stabilität und Datenspeicherung verhindern. Eine weitere Gelegenheit, das sicherzustellen, ist das regelmäßige Entfernen unnötiger Dateien. Dies ist in WebDAV möglich, auch wenn ihre maximale Speicherdauer noch nicht abgelaufen ist. Außerdem sollte das Limit von 100.000 Dateien in einem Ordner nicht überschritten werden[37].

Salesforce B2C bietet auch mehrere Ebenen von Log-Meldungen. Außerdem enthalten die Protokolldateien nur Einträge einer bestimmten Stufe oder eines bestimmten Anwendungsservers. Das heißt, die Aufzeichnungen werden nicht vermischt. Deshalb sind eine effektive Suche sowie Analyse möglich. Zu diesen Ebenen gehören:

- Info – Meldungen, die über Skripte, Code, Vorlagen und andere Bereiche von SFCC B2C informieren;
- Error – Meldungen über die Fehler in Vorlagen, Code, Skripten und anderen Bereichen von SFCC B2C;
- Warn – Nachrichten, die als Reaktion auf mögliche Probleme mit dem Slot oder den Servlets generiert werden. Dazu gehören auch die Lock-Statusberichte, die sich besonders gut zur Information über den Job eignen. Normalerweise enthalten diese

[34] Vgl. [Sal26 a]
[35] Vgl. [Med26 a]
[36] Vgl. [Sal26 a]
[37] Ebenda

Meldungen folgende Komponenten: Datum sowie Uhrzeit, WARN, zu berücksichtigender Bereich (z.B. RequestHandlerServlet, Rendering, Name des Servers oder der Anwendung) und deren Details;

- Debug – Meldungen, die Debugging-Informationen von der Website anzeigen. Diese Art von Logs ist auf Produktionsinstanzen immer deaktiviert. Auf Nicht-Produktionsinstanzen können sie jedoch eingeschaltet werden;
- Fatal – Meldungen, die verhängnisvolle Fehler in Vorlagen, Code, Scripts und anderen Bereichen von SFCC B2C dokumentieren[38].

Im Übrigen wird das Schreiben von Warnungen und Fehlermeldungen mit dem Zweck der Minimierung von Redundanz überwacht. Wenn eine Meldung während der ersten drei Minuten mehr als zehn Mal eingetroffen ist, wird ihr Versand angehalten. Dieser Vorgang dauert durchschnittlich 180 Sekunden. Eine entsprechende Nachricht erscheint im Protokoll. Wenn die Mitteilung innerhalb dieser Zeitspanne nach Ablauf der Sperrzeit noch zehn Mal angezeigt wird, wird der Prozess wiederholt[39].

3.1 System sowie Custom Logs

Zunächst werden die Systemprotokolle betrachtet. Sie zeichnen Informationen über das System auf und sind im Standard aktiviert, aber nicht konfigurierbar[40]. Sie können beispielsweise melden, dass Server Core betriebsbereit ist, Demandware Custom Servlet Context gestartet wurde, die Initialisierung abgeschlossen ist usw. Die Abbildung 2 zeigt ein Systemlog-Beispiel.

```
[2026-02-13 07:17:59.920 GMT] User system clears static content cache of Sites-Example -AT-Site
```

Abbildung 2: System Log

Als Nächstes folgen die benutzerdefinierten Protokolle, die vom Entwickler in den Code geschrieben werden. Sie unterstützen alle oben genannten Ebenen der Protokollmeldungen und enthalten Informationen zu benutzerdefinierten Jobs, Importen, Zahlungen oder Fehlern im Skript, die sich auf Benutzer auswirken können. Die Dateien, in denen sie abgelegt sind, beginnen mit dem Präfix „custom“. Jeder Anwendungsserver kann täglich bis zu 10 MB Protokolldaten darin speichern. Darüber hinaus ermöglicht Salesforce B2C, Logdateien mit individuell anpassbaren Namen zu erstellen. Dazu wird die Methode

[38] Vgl. [Sal26 a]
[39] Ebenda
[40] Ebenda

public static Log getLogger(String fileNamePrefix, String category) verwendet. Damit dieses Verfahren im Skript ausgeführt werden kann, muss zuerst die Kategorie der Logs im Business Manager eingetragen werden. Beim ersten Aufruf der Methode werden eine Instanz des Protokolls und eine Datei erstellt, in der die Daten gespeichert werden. Anschließend kann diese Instanz zum Aufzeichnen von Logs in Dateien verwendet werden[41]. Abbildung 3 zeigt ein Beispiel dafür, wie ein benutzerdefiniertes Protokoll gespeichert werden kann:

```
[2026-02-14 15:46:24.782 GMT] INFO CustomJobTh-read|0123456789|ChannelPilotFeed|createCSV_Example -NL
custom.channelPilot_Example -NL []  Job [ChannelPilotFeed] - Product 171819202122 could not be written
because it has no inventory record
```

Abbildung 3: Custom Log

3.2 Quota Log

Salesforce B2C verwendet Limits und Beschränkungen beispielsweise für die Nutzung von Speicher, die Anzahl von API-Aufrufen sowie Business-Objekte, um einen stabilen und effizienten Betrieb der Plattform zu gewährleisten. Quoten kontrollieren ihrerseits die Verwendung von Ressourcen im Benutzercode[42]. Wenn der Warnschwellenwert (60% des Limits) oder das Limit erreicht wird, wird eine entsprechende Meldung generiert. Diese Protokolle können Informationen zum Status und zu den Beschränkungen von Systemobjekt- und API-Quoten, zum Zustand der Sandbox-Datenbank sowie zu den Namen veralteter Methoden und deren Verwendungshäufigkeit in den letzten 24 Stunden enthalten[43].

Die erstellte Meldung wird nicht sofort in die Protokolldateien geschrieben. Stattdessen sammelt die Plattform zunächst Nachrichten über identische Ereignisse und schreibt anschließend eine zusammengefasste Mitteilung mit allen entsprechenden Informationen in eine Aufzeichnungsdatei. Diese Files enthalten Protokolleinträge, deren Format aus drei Hauptkomponenten besteht: Zeitstempel, Name des Threads und aktuelle Meldung. Der Name des Threads umfasst die Art der Anfrage, die Website, den Identifikator der Pipeline usw. Die Protokollmeldung enthält detaillierte Informationen, die dabei helfen, das Problem besser zu verstehen. Dazu zählen Daten darüber, ob die Quote durchgesetzt wurde oder nicht, der Warnschwellenwert (falls vorhanden), der Grenzwert, die Anzahl der Überschreitungen, der höchste gemessene Wert und der aktuelle Standort, an dem das Limit überstiegen wurde. Zusätzlich können

[41] Vgl. [Sal26 a]
[42] Vgl. [Sal26 b]
[43] Vgl. [Sal26 a]

Informationen zu ausgeführten Pipelines, Skripten, Pipeline-Knoten oder Vorlagen angegeben werden[44]. Ein typisches Quota-Log sieht wie folgt aus (Abb. 4):

```
[2026-02-16 05:43:09.322 GMT] [PipelineCallServlet|0123456789|Sites-Example-CH-Site|Cart-AddProduct|
PipelineCall|9r5C9takAW] Quota api.jsJSONStringLength (not enforced, warn 600000, limit 1000000):
warn threshold exceeded 423 time(s), max actual was 658671, current location:
request/site Sites-Example-CH-Site/top pipeline Cart-AddProduct/script mo-dules/server/route.js:101,
further information: n/a
```

Abbildung 4: Quota Log

3.3 Security Log

Die Gewährleistung der Sicherheit ist ein wichtiger Bestandteil jedes Online-Shops. Dabei helfen die Security-Logs. Sie geben einen Überblick über den aktuellen Sicherheitsstatus des Shops und ermöglichen die Analyse schutzrelevanter Ereignisse. Das Sammeln dieser Protokolle kann dabei helfen, Schwachstellen im Security-System aufzudecken und Probleme wie Betrug, Cyberangriffe oder anderes verdächtiges Verhalten besser zu untersuchen[45].

Erstens zeichnen sie die Informationen über Salesforce-Systeme auf, die den Betrieb des B2C-Commerce-Dienstes ermöglichen. Dazu gehören Firewalls, Router, Netzwerkswitches und Betriebssysteme. Ein weiterer Anwendungsfall ist die Erfassung von Daten zu Anmeldung im Business Manager. Diese informiert darüber, wer, wann und wo versucht, Zugriff zu erhalten. Diese Logs enthalten Uhrzeit, Datum, Benutzername, System oder dessen Modul, auf das Zugang erfolgt, IP-Adresse des Users, Aktion und Ergebnis. Sie sind besonders nützlich, um Cyberangriffe, Betrug und verdächtiges Verhalten zu verfolgen. Zusätzlich wird nach dem Einloggen eine einzigartige Sitzungs-ID generiert, die in Logs angezeigt werden kann. Bei einer erneuten Anmeldung aufgrund von Inaktivität werden außerdem die neue und die alte Identifikationsnummer im Protokoll eingetragen[46].

Ein weiteres Beispiel für die Generierung eines Sicherheitsprotokolls ist der Zugriff auf WebDAV und die Interaktion mit den darauf befindlichen Dateien. Diese Meldungen haben alle die gleiche Struktur. Sie enthalten den Namen der WebDAV-Methode und den Pfad zur aufgerufenen oder abgefragten Datei bzw. zum Ordner. Zu den entsprechenden Verfahren gehören das Erstellen eines Verzeichnisses (MKCOL), das Hochladen (PUT), das Herunterladen (GET)

[44] Vgl. [Sal26 a]
[45] Vgl. [Sal26 c]
[46] Vgl. ebenda

und das Verschieben von Dateien an einen anderen Speicherort (MOVE). Das Durchsuchen von Ordnern wird allerdings nicht in den Sicherheitsprotokollen gespeichert[47].

Ein weiterer wichtiger Aspekt, den Salesforce B2C bietet, ist die Aufzeichnung von Mitarbeiteraktivitäten in sensiblen Bereichen. Dazu gehören Sicherheitseinstellungen, der Zugriff auf Kunden- oder Bestelldaten, auf Kampagnen oder Gutscheine sowie die Änderung von Benutzerrollen und Zugangsberechtigungen. Jeder Vorgang (Lesen oder Schreiben) wird protokolliert. Der Log-Eintrag enthält in der Regel den in Salesforce registrierten Benutzernamen (in der Regel die E-Mail-Adresse), den Bereich und die Aktivität. Das Sicherheitsprotokoll kann wie folgt aussehen (siehe Abbildung 5)[48].

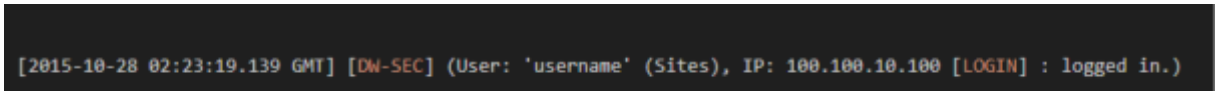


Abbildung 5: Security Log
Quelle: [Sal26 c]

3.4 Import sowie Export Log

Eine weitere Funktion von Salesforce B2C ist der Export und Import verschiedener Datentypen. Diese Prozesse können über den Business Manager oder über Pipelet im Code ausgeführt werden. Detaillierte Informationen hierzu können in den Export- und Import-Protokolldateien gefunden werden, die im Impex-Verzeichnis abgelegt sind. Diese Art von Logs gibt einen Überblick über Fehler bei Prozessen wie dem Hochladen von Katalogen, Preislisten, Benutzerkanälen usw., bei der Ausführung von Jobs oder bei Datenproblemen. Die Struktur des Verzeichnisses, in dem diese Protokolle gespeichert sind, ist für alle Instanzen (Sandbox, Staging, Development und Production) identisch. Es besteht aus den Ordnern „log“ und „src“. Das Verzeichnis „log“ enthält Aufzeichnungen, in denen Informationen über die Batch-Verarbeitung von Katalogen, Kunden und Produkten sowie die Übernahme und Ausgabe von Katalogen, Inventar und Preislisten abgelegt sind. Der andere Ordner („src“) umfasst Import- und Exportquelldateien mit Details zu entsprechenden Prozessen und benutzerdefinierten Jobs[49]. Die Darstellung des Logs zum Datenimport ist in Abbildung 6 zu sehen.

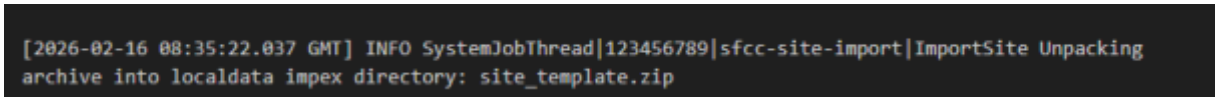


Abbildung 6: Import Log

[47] Vgl. [Sal26 c]
[48] Ebenda
[49] Vgl. [Sal26 a]

3.5 Batch-Processing Log

Salesforce B2C bietet eine Batch-Verarbeitung von Business Manager-Aufgaben in Bezug auf Kunden, Kataloge und Produkte an. Mit diesen Objekten können die folgenden Vorgänge durchgeführt werden: Aktualisieren, Kopieren und Löschen. Darüber hinaus können sie einer Kategorie hinzugefügt oder aus dieser entnommen werden und ihre Attribute können geändert oder entfernt werden[50]. Entsprechende Meldungen im Protokoll werden nur bei Fehlern im Prozess oder fehlgeschlagenen Vorgängen generiert. Daher kann die Ebene der Batch-Verarbeitungsprotokolle nur WARN, ERROR oder FATAL sein. Sie werden im Import-/Exportprotokollkatalog gespeichert und können zusätzlich im Business Manager auf der Seite „Batch Processing“ eingesehen werden[51]. Hier ist ein Beispiel für das Protokoll (siehe Abb. 7), wenn ein Kunde während der Batch-Verarbeitung nicht gelöscht werden kann.

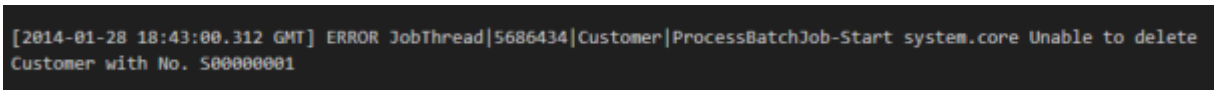


Abbildung 7: Batch Processing Log
Quelle: [Sal26 d]

3.6 Andere Arten von Logs in Salesforce B2C

Salesforce B2C bietet auch andere Arten von Logs, um die laufenden Prozesse im Online-Shop effektiver zu überwachen. Weitere Informationen dazu sind in Tabelle 3 aufgeführt.

Tabelle 3: Andere Arten von Logs in Salesforce B2C
Quelle: [Sal26 a]

Log-Typ	Verwendungszweck
deprecation	Sie zeigen an, ob veraltete APIs verwendet wurden. Die Syntax umfasst Datum und Uhrzeit, API-Name und Nutzungshäufigkeit.
analytics	Sie tracken die Aktivitäten mit B2C Commerce Analytics.
api	Sie berichten über Probleme mit der API. Der Protokolleintrag enthält den Bereich (Pipeline), das Datum und die Uhrzeit, die Art des Problems (Fehler, Information, Vorlage, Warnung), den Pfad und Details zum Problem.
jobs	Sie zeigen den Status der Ausführung von Jobs sowie zusätzliche Informationen dazu an, z.B. Datum, Uhrzeit, Bereich, Status (beispielsweise „Job Manager Stopped“).
migration	Sie zeigen Informationen zu internen Migrationsdaten.

[50] Vgl. [Sal26 d]
[51] Vgl. [Sal26 a]

performance	Sie zeigen Informationen zu internen Leistungsdaten.
sql	Sie können dabei helfen, Probleme bei der Replikation zu finden.
staging	Sie zeigen den Ausführungsstatus von Jobs und zusätzliche Daten.
sysevent	Sie zeigen die Anmeldung des Anwendungsservers und Informationen zur Cartridge an. Jeder Eintrag enthält das Datum, die Uhrzeit und das Ereignis.
syslog	Sie zeigen Informationen zur API-Verarbeitung, Datenaufbereitung, Fehlerbehandlung, zum Import und Export sowie zum Host und zur Version des aktivierten Codes an.
console	Dies ist ein internes Konsolenprotokoll, das Informationen über den AppServer anzeigt. Der Dateiname dieses Typs hat folgende Struktur: console-hostname-appserver_id-server_name-timestamp.log. Jeder Eintrag enthält das Datum, die Uhrzeit, den Pfad, den Befehl (init, load, start oder log) und den Meldungstyp (Error, Info, Severe oder Warning).

4 Log Center

Der Business Manager ist nicht nur ein einziger Bereich, in dem die generierten Aufzeichnungen eingesehen werden können. Eine weitere Gelegenheit ist das Log Center – eine Anwendung, die vom System und Benutzercode erzeugten Logs sowohl für PIGs (Primary Instance Groups: Produktions-, Staging- und Development-Instanz) als auch für SIGs (Secondary Instance Groups: Sandbox-Instanzen - die Umgebung, in der Entwickler den Code testen) enthält. Darüber hinaus ermöglicht es nicht nur die Einsicht in Protokolle. Weitere Funktionen umfassen die Erstellung von Dashboards, die Suche nach Aufzeichnungen, die Konfiguration von Log Stream für die erweiterte Analyse von Einträgen in einem externen Dienst, die Einstellung von Log-Ebenen und deren Volumen (jedoch nur mit entsprechenden Berechtigungen), die Überwachung des Zustands der B2C Commerce-Instanz mit Control Center Audit und die Überprüfung der Leistung, Sicherheit und Ressourcennutzung des Shops mit Metrics Dashboard[52].

4.1 Log Center Dashboard

Das Dashboard ist ein Element des Log Centers, das die Trace-Stacks von Salesforce-Instanzen in Form eines Histogramms anzeigt. Die dargestellten Daten hängen von den Berechtigungseinstellungen im Account Manager ab. Beispielsweise können Benutzer mit Administrator- oder Supportrechten die Daten für jedes Realm[53] (eine Einheit der Salesforce-Architektur, die aus Instanzen besteht, auf denen der Online-Shop entwickelt, getestet und implementiert wird[54]) im Unternehmen einsehen. Standardmäßig werden die zehn häufigsten Stack-Traces der letzten zehn Tage angezeigt. Mit Hilfe von Filtern können sie allerdings für einen konkreten Realm (nur für Administratoren), eine Instanz oder einen Dienst gefunden werden. Zusätzlich können die Stack-Traces für einen bestimmten Zeitraum oder eine Region, ihre genaue Anzahl pro Dienst sowie die entsprechenden Meldungen angezeigt werden[55].

4.2 Log Center Search

Die Suche nach Protokollmeldungen anhand bestimmter Kriterien ermöglicht die Log Center Search. Die Anwendung bietet die Gelegenheit, Filter nach folgenden Kategorien einzustellen: Zeitraum, Schweregrad, Fehlertyp, Schlüsselwörter, spezifischer Text usw. Die Protokolle in

[52] Vgl. [Sal26 e]
[53] Vgl. [Sal26 g]
[54] Vgl. [Sal26 f]
[55] Vgl. [Sal26 g]

der Log Center Search können aber nur 14 Tage lang gespeichert werden. Nach Ablauf dieser Frist ist die Anzeige dieser Einträge nicht mehr möglich. Obwohl das Log Center große Datenmengen verarbeiten kann, gibt es jedoch Einschränkungen hinsichtlich der Anzahl der Log-Dateien, die gespeichert werden können. Sie sind für SIG und PIG unterschiedlich. Beispielsweise kann SIG täglich zwischen einer und zwei Millionen Protokolle enthalten, während die Menge der Logs in PIG vom Instanztyp (Produktion, Staging usw.) abhängt. Darüber hinaus kann eine Suchregel erstellt und gespeichert werden. Dies ist besonders sinnvoll, wenn bestimmte Aufzeichnungen mehrfach gesucht werden müssen[56].

Mithilfe von Schlüsselwörtern (z. B. Text oder ID) können die Protokolle mit bestimmten Begriffen gefunden werden. Die Suche wird in verschiedenen Bereichen durchgeführt. Dazu gehören der Text der Meldung, die ID der Aktion, der Name der Kategorie, der Name des Threads, die Meldung des Trace-Stacks usw. Zudem wird in der Log Center Search auf der linken Seite die Anzahl der Protokolle auf jeder Ebene nach Schweregrad angezeigt, die den Suchbedingungen entsprechen. Die Anwendung unterstützt auch die AND-Logik. Das bedeutet, dass die Suche nach mehreren Begriffen gleichzeitig durchgeführt werden kann. Zu beachten ist, dass die Filterkriterien nicht groß-/kleinschreibungssensitiv sind und ihre Reihenfolge keinen Einfluss hat. Außerdem gibt es eine Reihe von Wörtern, die bei der Suche ignoriert werden, z.B. a, an, be, if, in, into, was, will usw[57].

Es gibt zwei Arten der Suche: die grundlegende und die erweiterte Suche. Die erste Option führt die Abfrage gleichzeitig in allen Bereichen durch. Als Ergebnis wird eine Liste von Aufzeichnungen dargestellt, die mindestens in einem Gebiet den genauen Textabschnitt enthalten. Andernfalls werden keine Einträge gefunden. Die Verwendung von Operatoren oder Wildcards ist nicht erlaubt. Während der erweiterten Suche ist die gleichzeitige Durchführung von Searchparametern in gewünschten Bereichen möglich. Beispielsweise enthält das Protokoll den Text „Pipeline“ und seine Kategorie ist „Checkout“. Für die erweiterte Suche können die folgenden Verfahren verwendet werden:

- Spezielle Zeichen wie +, -, =, &&, ||, <, > usw., die vom System als Operatoren interpretiert werden;
- Boolesche Operatoren wie AND, OR, NOT, - , die groß-/kleinschreibungssensitiv sind;
- Wildcards wie das Sternchen (*), das das Fehlen oder Vorhandensein mehrerer Zeichen bezeichnet, oder das Fragezeichen (?), das genau ein Zeichen darstellt;
- die Gruppierung von Suchbegriffen mit Klammern;

[56] Vgl. [Sal26 h]
[57] Vgl. [Sal26 i]

- die Verwendung regulärer Ausdrücke;
- eine phasenbezogene Suche (Phrase Query), bei der die Reihenfolge wichtig ist;
- eine ungefähre Suche (Fuzzy Query), die nach ähnlichen, aber nicht genau übereinstimmenden Begriffen sucht. Am Ende eines Ausdrucks oder einer Phrase wird ein Tilde-Zeichen (~) verwendet[58].

4.3 Log Streaming

Eine weitere Möglichkeit, die das Log Center für eine erweiterte Auswertung von Protokollen bietet, ist deren Übertragung an einen externen Analysedienst. Die Auswahl an verfügbaren Anwendungen ist umfangreich. Dazu gehören Splunk, Datadoc, New Relic, Grafana Loki, Sumo Logic, Dynatrace und AWS Elastic Cloud. Eine weitere wichtige Funktion ist die Verwendung von generischem HTTP. Damit können Logs nicht nur an verfügbare Dienste, sondern auch an den eigenen Endpunkt, andere Dienste usw. übertragen werden. Die Einführung der Übermittlung von Protokollen bietet folgende Vorteile: Automatisierungsmöglichkeiten, skalierbare Überwachung für große Unternehmen mit komplexer Struktur und erweiterte Analyse[59].

Für die Übertragung von Protokollen verwendet das Log Streaming Amazon Kinesis Data Firehose. Dabei handelt es sich um einen Server, der Streaming-Daten an verschiedene AWS- sowie Drittanbieter-Ziele überträgt. Nach dem Erhalt der Salesforce-Protokolle und vor deren Weiterleitung an einen Drittanbieter werden sie in ein passendes Format umgewandelt. Dazu wird AWS-Lambda verwendet. Die Datentransformation umfasst die Konvertierung des Formats (z. B. von JSON in CSV), das Hinzufügen zusätzlicher Metadaten zu den Firehose HTTP-Header X-Amz-Firehose-Common-Attributes[60], die während der Konfiguration von Log Streaming als Parameterattribute eingegeben wurden[61] sowie die Komprimierung von Informationseinheiten mit GZIP, Snappy oder ZIP, um die Speicherkosten zu reduzieren. Eine weitere wichtige Funktion, die Firehose bietet, ist das Backup von Daten. Auf diese Weise können fehlgeschlagene Protokolle zur weiteren Verarbeitung im S3-Bucket gespeichert werden. Darüber hinaus puffert der Server Daten, um eine effiziente Übermittlung zu gewährleisten[62]. Die Logs werden in Batches mit einem Intervall von 20 Sekunden bis zwei Minuten gesendet. Dies hängt von der Geschwindigkeit der Sammlung und Übermittlung der Protokolle ab[63]. Der gesamte Prozess der Protokollübertragung ist in Abbildung 8 dargestellt.

[58] Vgl. [Sal26 i]
[59] Vgl. [Sal26 j]
[60] Vgl. [Med26 b]
[61] Vgl. [Sal26 j]
[62] Vgl. [Med26 b]
[63] Vgl. [Sal26 j]

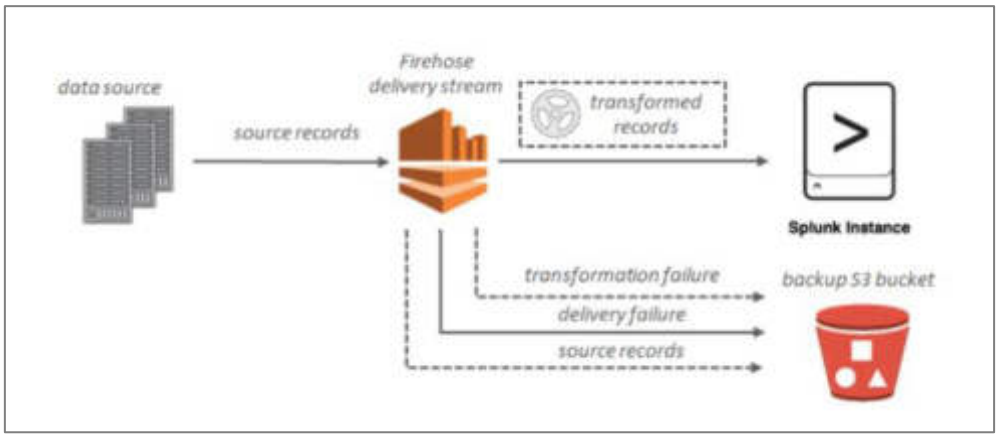


Abbildung 8: Der Prozess der Datenübermittlung an Drittanbieter
Quelle: [Med26 b]

Die Übermittlung von Protokollen mit dem Log Streaming ist nur in der Sandbox möglich und findet ausschließlich in Echtzeit statt. Deshalb kann es nicht für den periodischen Betrieb konfiguriert werden[64].

Bei der Einrichtung der Protokollübertragung können auch zusätzliche Optionen verwendet werden. Beispielsweise kann der primäre Besitzer, der den Log Stream konfiguriert, neue Benutzer hinzufügen oder deren Berechtigungsliste ändern. Eine weitere Möglichkeit ist die Anpassung von Filtern. SFCC B2C bietet zwei Arten von Abfrageoptionen: normale und negative. Bei Verwendung der ersten Art werden Logs übertragen, die den Auswahlbedingungen entsprechen. Die Negativen dagegen geben an, welche Aufzeichnungen nicht gesendet werden sollen. Die Filterkategorien umfassen: Art des Logs (system, custom, syslog, api usw.), Anfrage (JOB, BUSINESSMGR, STOREFRONT, TEMPORARY, REST), Dienst (ecom, JWA, MRT), Schweregrad (info, error, warn usw.), Name des Tenants und dessen Typ (sbx, prd, stg, dev). Es ist allerdings zu beachten, dass die gleichzeitige Benutzung einer Kategorie für normale und negative Filter nicht erlaubt ist[65].

Eine weitere Möglichkeit, die das Log Streaming bietet, ist die Weiterleitung von Sicherheitsprotokollen. Dies ist jedoch nur mit entsprechenden Rechten möglich, nämlich mit „Log Center External Admin“, da diese Art von Aufzeichnungen vertrauliche Daten wie Anmelde- und Kundeninformationen enthält. Wenn sich die Rolle des Benutzers ändert, z. B. zu User, wird das Log Streaming automatisch deaktiviert[66].

[64] Vgl. [Sal26 j]
[65] Vgl. [Sal26 k]
[66] Vgl. [Sal26 l]

5 Grafana Labs

Die drei miteinander verbundenen Begriffe, die derzeit in verschiedenen Bereichen eine wichtige Rolle spielen, sind Observability, Monitoring und Telemetrie. Der erste Ausdruck beschreibt die Fähigkeit, die Prozesse im System zu verstehen und zu erkennen, ob sie wie geplant funktionieren. Wenn der Ablauf nicht wie vorgesehen erfolgt, hilft Observability dabei, die Ursachen des Problems zu ermitteln. Der zweite Begriff bezeichnet die Eigenschaft, Fehler zu erkennen. Der letzte Term schließlich ist ein wichtiger Bestandteil der Überwachung. Er bezeichnet die Erfassung wichtiger Daten, deren Übertragung sowie die Erstellung von Visualisierungen, Warnmeldungen und Berichten. Um das System vollständig zu verstehen und eine effektive Kontrolle durchzuführen, sollten diese drei Konzepte gemeinsam angewendet werden. Ein Beispiel für einen Anbieter, der diese Voraussetzungen erfüllt, ist Grafana Labs. Sie bietet eine Reihe von Anwendungen an und unterstützt Open-Source-Projekte. Zu den Tools gehören der Kern-Grafana-Stack, Enterprise-Plugins, Grafana Incident Response und Management sowie weitere Applikationen wie Faro, k6 und Pyroscope[67].

5.1 Kern-Grafana-Stack

Grafana Labs unterstützt drei Haupttypen von Telemetrie: Metriken, Logs und verteilte Traces. Aus diesem Grund ist die Struktur des Kern-Grafana-Stacks eng mit diesen verbunden. Dieser besteht aus LGTM und dem Grafana-Agenten. Die Abkürzung LGTM steht für Mimir, Loki, Tempo und Grafana. Dies sind die Hauptkomponenten, die die Speicherung und Visualisierung von Telemetrie gewährleisten[68].

Mimir ist eine Zeitreihendatenbank (TSDB), die für die Speicherung von Metriken zuständig ist. Metriken sind numerische Daten, die zu einem bestimmten Zeitpunkt aufgezeichnet wurden und zusätzliche Markierungen und Messungen zur Analyse enthalten. Ihr Einsatz löst das Problem der Skalierbarkeit, wenn große Datenmengen abgelegt und anschließend durchsucht werden müssen. Darüber hinaus nutzt sie kostengünstige Speicherobjekte wie S3, GCS oder Azure Blob Storage. Im Gegenzug ist Loki eine Reihe von Komponenten, die eine preiswerte Speicherung und effektive Überwachung von Protokollen ermöglichen. Es verwendet auch sparsame Aufbewahrungsmodelle wie S3 und GCS. Außerdem indexiert Loki nicht die vollständigen Einträge, sondern nur Label-Metadaten. Die Menge der erfassten Daten ist deshalb gering, was es von anderen unterscheidet[69].

[67] Vgl. [CH24]
[68] Vgl. ebenda
[69] Ebenda

Die nächste Komponente des Grafana-Stacks ist Tempo, ein System zur Speicherung großer Mengen verteilter Traces (Indikatoren, die den gesamten Prozess der Durchführung einer Aktion abbilden). Es verarbeitet jedes Element dieses Telemetrietyps vollständig, von Anfang bis Ende. Somit wird das Risiko verringert, dass bestimmte Informationen unberücksichtigt bleiben. Darüber hinaus bietet Tempo die Möglichkeit, Spans (Teile verteilter Traces) in Metriken umzuwandeln und diese dann an andere Systeme zu senden, die die Remote-Aufzeichnung von Prometheus unterstützen. Außerdem ist der Storage-Typ identisch mit dem von Loki und Mimir[70].

Das Element, dessen Aufgabe die Datenvisualisierung umfasst, ist Grafana. Es ermöglicht die Verwendung verschiedener Arten von Grafiken und Diagrammen. Auf diese Weise können die unterschiedlichsten Datentypen optimal visualisiert werden. Zusätzlich unterstützt Grafana zahlreiche Plugins für viele Arten von Daten. Daher kann es mit diesen oder auch anderen Überwachungstools verbunden werden. Ein weiterer Bestandteil des Grafana-Stacks ist der Grafana-Agent. Dabei handelt es sich um eine Reihe von Tools, deren Aufgabe in der Erfassung von Protokollen, Metriken und verteilten Traces besteht[71].

5.2 Grafana Incident Response und Management

Die rechtzeitige Erkennung von Problemen ist entscheidend für die Aufrechterhaltung eines stabilen Betriebs von Programmen oder Systemen. Aus diesem Grund ermöglicht Grafana Labs die Verwendung von „Incident Response und Management“ (IRM). Es besteht aus drei Komponenten. Die erste sind Alarmregeln. Sie informieren über Probleme mithilfe von Benachrichtigungen in der Anwendung, per E-Mail oder über Grafana OnCall. Die zweite Komponente ist das bereits erwähnte „Grafana OnCall“. Dabei handelt es sich um ein Tool, das sich ständig im Bereitschaftsmodus befindet, um Fehler zu berichten. Darüber hinaus ermöglicht es die Zentralisierung und Gruppierung von Alarmmeldungen. Eine weitere wichtige Funktion ist die vordefinierte Eskalationssequenz. Die letzte Komponente ist „Grafana Incident“. Dies ist ein weiteres Tool zur Bearbeitung von Vorfällen. Zudem ist die Verwendung eines Chatbots möglich, der viele Aufgaben ausführen kann. Beispielsweise kann er bei einem Unfall Kanäle für Vorfälle und Bereiche für die Zusammenarbeit erstellen. Er kann auch Zeitstempel sammeln, was für die Analyse eines Ereignisses nach dessen Eintritt besonders wichtig ist. Eine weitere Funktion ist die automatische Ausführung von Anfragen von Entwicklern, z.B. die Behebung eines Vorfalls[72].

[70] Vgl. [CH24]
[71] Ebenda
[72] Vgl. ebenda

6 Grafana Loki

Grafana Loki ist eine Reihe leistungsstarker Komponenten, die eine erweiterte Verarbeitung von Protokollen und kostengünstige Lösungen für deren Speicherung bieten. In folgendem Abschnitt werden daher dieses Tool und die Möglichkeiten, die es für die fortgeschrittene Bearbeitung von Aufzeichnungen bereitstellt, vorgestellt.

6.1 Architektur von Grafana Loki

Zunächst wird die Architektur von Grafana Loki im Fokus stehen. Das gibt ein grundlegendes Verständnis über die Prozesse, die in dieser Anwendung ausgeführt werden.

Grafana Loki hat eine Microservices-Architektur. Es wurde als verteiltes System konzipiert, das horizontal skalierbar ist. Die Komponenten der Microservices ermöglichen die Ausführung folgender Aufgaben: Lesen, Schreiben und Speichern von Protokollen (siehe Abbildung 9)[73]. Loki kann in zwei Bereitstellungsmodi betrieben werden: monolithisch und mikroservicebasiert. Im ersten Fall werden alle Elemente des Microservices als Teil eines einzigen Prozesses in Form einer Binärdatei oder eines Docker-Images ausgeführt (siehe Anlage 1). Dies ist effektiv, vorausgesetzt, dass das Volumen der Lese- oder Schreibvorgänge nicht zu groß ist und maximal etwa 20 GB pro Tag beträgt. Andererseits werden die Komponenten bei der Verwendung eines mikroservicebasierten Modus als separate Prozesse betrieben (siehe Anlage 2). Dieses Verfahren ermöglicht Granularität sowie die separate Konfiguration der einzelnen Komponenten entsprechend den Anforderungen des Dienstes[74].

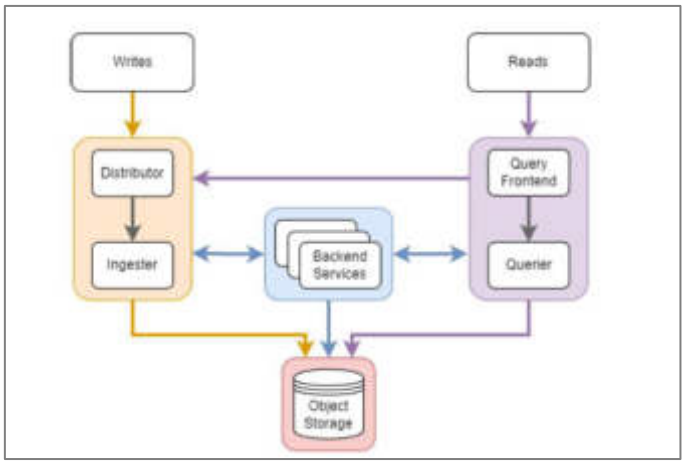


Abbildung 9: High-Level-Architektur von Grafana Loki
Quelle: [CH24]

[73] Vgl. [CH24]
[74] Vgl. [Gra26 a]

Zu den Hauptkomponenten von Loki, die Lese- und Schreibfunktionen ausführen, gehören: Distributor, Ingester, Query-Frontend, Querier und Backend-Dienste (Ruler, Compactor und Query Scheduler).

Der Schreibvorgang beginnt beim Distributor. Seine Aufgaben umfassen: den Empfang von Logs, die Partitionierung der bekommenen Daten und deren Weiterleitung an die Ingester. Nach dem Erhalt einer Reihe von Streams überprüfen die Distributoren sie. Dieser Prozess beinhaltet die Validierung von Labels, Zeitstempeln und der Größe der Protokollzeilen. Anschließend fasst er sie zu Batches zusammen und sendet die Chunks an verschiedene Ingester. Nach diesem Vorgang nehmen die Aufnahmekomponenten die Blöcke entgegen und leiten sie an das Storage-Objekt weiter. Zunächst speichern sie die Chunks jedoch in WALs (Write-Ahead Logs), um Datenverluste zu vermeiden[75].

Beim Lesen wird zunächst das Query-Frontend verwendet. Zu seinen Aufgaben gehören die Beschleunigung der Abfrageausführung, die Verteilung großer Queries auf verschiedene Queriers sowie die Wiederholung der Operation im Fehlerfall. Die Queriers analysieren ihrerseits LogQL (eine für Loki entwickelte Abfragesprache) und interagieren mit dem Ingester und dem Storage-Objekt. Als Antwort auf die Anfrage an den ersteren werden die aktuellsten Daten bereitgestellt. Zusätzlich ermöglicht das Speicherobjekt den Zugriff auf alte Protokolle[76].

Zur Erfüllung der Hauptaufgaben von Grafana Loki werden auch Backend-Dienste aktiv genutzt. Compactor ist beispielsweise für die Verwaltung der in Batches verarbeiteten Logs im Speicherobjekt zuständig. Darüber hinaus kontrolliert es das Storage-Objekt, dedupliziert Daten und löscht alte Protokolle. Eine weitere Komponente des Serverdienstes ist Ruler, der Queries auswertet und basierend auf den Ergebnissen bestimmte Aktionen ausführt. Eine mögliche Maßnahme kann beispielsweise das Senden von Meldungen zu Systemereignissen sein[77].

6.2 Die Struktur von Logs in Loki

Nach der Auseinandersetzung mit der Architektur von Grafana Loki wird die Struktur der dort bereits gespeicherten Logs betrachtet. Auf dieser Grundlage wird nachvollziehbar, warum Loki ein Dienst ist, der eine kostengünstige Speicherung und eine effiziente Suche nach Protokollen gewährleistet.

[75] Vgl. [CH24]
[76] Ebenda
[77] Ebenda

Grafana Loki unterstützt zwei Dateitypen: „Index“ und „Chunks“. Der erste ist eine Tabelle mit Verweisen auf bestimmte Log-Einträge. Chunks hingegen sind Container, in denen Protokollaufzeichnungen gespeichert werden[78]. Die Logs werden im Objektspeicher in Form von Log-Streams abgelegt, wobei jeder Eintrag aus folgenden Komponenten besteht: Zeitstempel, Labels und Inhalt (siehe Abbildung 10). Die ersten beiden Elemente sind indexiert, das letzte jedoch nicht. Die Besonderheit, dass nicht alle Daten indiziert werden, unterscheidet Loki von anderen Anbietern und reduziert die Speicherkosten[79].

Der Zeitstempel wird für eine höhere Genauigkeit in Nanosekunden angezeigt. Deshalb ist es möglich, die Aufzeichnungen für bestimmte Zeiträume anzusehen. Der Inhalt des Logs seinerseits umfasst alle erforderlichen Informationen zum Ereignis. Er besteht aus unbearbeiteten Logzeilen, die gebündelt, komprimiert und als Chunks gespeichert sind. Labels sind Schlüssel-Wert-Paare, deren Aufgabe darin besteht, gespeicherte Daten zu identifizieren. In der Regel werden Metadaten, die zusammen mit Logs gesendet werden, in Grafana Labs als Bezeichner definiert. Außerdem bilden Tags den Loki-Index und haben folgende Eigenschaften:

- Jeder Log-Stream wird aus einer einzigartigen Kombination von Labels und Werten erstellt;
- Labels sind ein Index für den Loki-Log-Stream;
- Labels werden für die Suche nach Logs verwendet[80].

Abbildung 10: Loki-Logstruktur
Quelle: [CH24]

6.3 Log-Verarbeitung

Im nächsten Abschnitt wird die Verarbeitung von Protokollen behandelt. Dazu wird die Query-Sprache LogQL verwendet, die speziell für die Anforderungen dieses Systems entwickelt wurde. Da der Inhalt der Protokolle nicht indexiert wird und für die Suche nach bestimmten

[78] Vgl. [Gra26 b]
[79] Vgl. [CH24]
[80] Vgl. ebenda

Aufzeichnungen Labels verwendet werden, unterstützt Loki eine verteilte Filterung. Bei dieser Art der Filtration werden die Aufzeichnungen zunächst nach Tags und anschließend nach den in den Chunks gespeicherten Log-Daten selektiert[81].

Eine grundlegende LogQL-Abfrage besteht aus einem oder mehreren Log-Stream-Selektoren zum Abrufen von rohen Log-Zeilen und optional aus einer Log-Pipeline (z.B. Zeilenfilter, Parser, Aggregationsfunktionen usw.), die zum Filtern von diesen Einträgen sowie zu deren Analyse verwendet werden kann[82].

6.3.1 Log-Stream-Selektor

Zunächst wird die Filterung von Protokollen nach Labels betrachtet. Auf diese Weise können bestimmte Log-Streams in Grafana Loki mit einfachen Operatoren gesucht werden. Zu diesen gehören:

- = – entspricht genau;
- != – entspricht nicht;
- =~ – entspricht einem regulären Ausdruck (Regex);
- !~ – entspricht nicht einem regulären Ausdruck[83].

Zum Durchsuchen von Aufzeichnungen in Loki muss mindestens ein Stream-Selektor verwendet werden. Dieser muss am Anfang der Anfrage in geschweiften Klammern stehen und so geschrieben sein, dass er nicht mit einer leeren Zeile übereinstimmt. Zudem besteht die Möglichkeit, mehrere Filterkriterien zu nutzen. Je besser diese formuliert sind, desto effizienter ist das Ergebnis der Anfrage[84]. Ein Beispiel für die Verwendung des Log-Stream-Selektors ist in Abbildung 11 dargestellt. Die Abfrage wurde so formuliert, dass Einträge aus einem bestimmten Realm gefunden werden.

```
{realm="zzgl_02"}
```

Abbildung 11: Verwendung des Log-Stream-Selektors

[81] Vgl. [CH24]
[82] Ebenda
[83] Ebenda
[84] Vgl. ebenda

6.3.2 Zeilenfilter

Nachdem die entsprechenden Log-Streams gefunden wurden, können diese weiterverarbeitet werden. Die erste Gelegenheit besteht darin, die rohen Log-Daten mithilfe von Zeilenfiltern zu bearbeiten. Dabei handelt es sich um eine groß-/kleinschreibungsabhängige Suche im Protokollinhalt, bei der nicht passende Ergebnisse nicht angezeigt werden. Die Auswahlbedingung wird mithilfe eines Filteroperators sowie von Text oder regulären Ausdrücken festgelegt. Die Tabelle 4 zeigt die verfügbaren Operatorzeichen[85].

Tabelle 4: Zeilenfilter-Operatoren
Quelle: [CH24]

Operator	Beschreibung
|=	Er wird verwendet, um eine Log-Zeile zu suchen, die einen bestimmten String enthält.
!=	Er findet Anwendung, wenn eine Log-Zeile gesucht werden muss, die einen bestimmten String nicht enthält.
|~	Er wird verwendet, wenn die Log-Zeile einem regulären Ausdruck entsprechen muss.
!~	Er wird benutzt, wenn die Log-Zeile nicht einem regulären Ausdruck entsprechen muss.

Abbildung 12 stellt die Suche nach Einträgen mithilfe von Zeilenfiltern, deren Daten einem regulären Ausdruck entsprechen dar.

```
{realm="zzgl_02"} |~ 'Feature\w+'
```

Abbildung 12: Suche nach Ausdrücken mithilfe von Zeilenfiltern

Darüber hinaus bietet Grafana Labs die Möglichkeit, die passende IP-Adresse, ohne die Verwendung komplexer regulärer Ausdrücke zu finden. Dies gewährleistet die Funktion ip("<pattern>"), die IPv4- sowie IPv6-Adressen unterstützt. Sie kann sowohl zum Filtern von Log-Zeilen als auch von Labels eingesetzt werden. Außerdem dürfen nur die Operatoren |= (= für Labels) und != verwendet werden[86].

6.3.3 Parser

Mithilfe von Zeilenfiltern können Aufzeichnungen nach einem bestimmten Text oder einem regulären Ausdruck im gesamten Log-Inhalt gesucht werden. Manchmal ist es jedoch

[85] Vgl. [CH24]
[86] Ebenda

erforderlich, den Zugriff auf bestimmte Felder in der Protokollzeile zu bekommen. Zu diesem Zweck bietet Grafana Labs die Möglichkeit, Parser zu verwenden.

Grafana Loki kann Logs aus verschiedenen Quellen empfangen. Folglich sind die Formate der erhaltenen Aufzeichnungen unterschiedlich. Einige enthalten beispielsweise JSON-Daten, andere wurden in Promtail erstellt usw. Aus diesem Grund bietet das System verschiedene Arten von Parsern an[87]. Diese und ihre Anwendungsfälle sind in Tabelle 5 aufgeführt.

Tabelle 5: Arten von Parsern in Grafana Loki
Quelle: [CH24]

Parser-Typ	Beschreibung
json	Dieser Parsertyp wird für Protokolle verwendet, wenn deren Inhalt strukturierte oder halbstrukturierte JSON-Daten sind oder die eingebetteten JSON-Daten enthalten. Um alle Eigenschaften als Labels zu extrahieren, muss „| json“ verwendet werden. Der Zugriff auf verschachtelte Schlüssel ist mit „_“ möglich. Jedes Tag wird damit mit diesem Zeichen getrennt werden, bis ein bestimmtes erreicht ist (beispielsweise message_category). Dazu kommt noch, dass nur bestimmte Eigenschaften extrahiert werden können. Zu diesem Zweck muss die Prozedur | json label1="expression", label2="expression" verwendet werden. Unter „expression“ sind hier Schlüssel oder verschachtelte Schlüssel zu verstehen. Außerdem werden Arrays bei diesem Parsen übersprungen. Es ist jedoch möglich, einzelne Elemente mit | json label1="expression" abzurufen.
logfmt	Für strukturierte Log-Inhalte, die als einstufige Schlüssel-Wert-Paare dargestellt werden, kann der Parser „logfmt“ genutzt werden. Dieser extrahiert alle entsprechenden Kombinationen. Dazu muss „| logfmt” in die Abfragebedingung eingesetzt werden. Außerdem gibt es die Möglichkeit, nur bestimmte Paare herauszuziehen. In diesem Fall sollte eine Prozedur ähnlich wie bei json-Parser verwendet werden: | logfmt label1="expression", label2="expression".
pattern	Für unstrukturierte Log-Inhalte sollte der Pattern-Parser verwendet werden. Dieser ermöglicht es, Felder in einer Protokollzeile mit Hilfe von | pattern "<expression>" zu extrahieren. Dabei entspricht „expression“ der Struktur der Log-Zeile. Wenn die Aufzeichnung beispielsweise das Datum, die Meldung und den Host-Server enthält, können diese wie folgt herausgezogen werden: | pattern „<Datum> <Host> <Meldung>“.
regexp	Eine weitere Möglichkeit, unstrukturierte Daten zu extrahieren, ist die Verwendung eines regulären Ausdrucks. Dazu muss | regexp <“expression“> aufgerufen werden, wobei „expression“ ein Regex ist, der mindestens eine Übereinstimmung haben muss.
unpack	Dieser Parser-Typ dient zum Extrahieren eingebetteter Labels, die mit der Funktion von Promtail erstellt wurden.

Als Ergebnis des durchgeführten Parsings werden die extrahierten Eigenschaften als Labels und ihre Werte betrachtet. Dies ermöglicht eine weitere Auswahl relevanter Einträge.

[87] Vgl. [CH24]

6.3.4 Label-Filter

In den vorherigen Abschnitten wurde bereits erwähnt, dass Labels auf zwei Arten erstellt werden können. Die erste besteht darin, dass Tags beim Sammeln von Einträgen aus verschiedenen Quellen definiert werden. In der Regel werden sie aus Metadaten gebildet und dann als Stream-Selektor verwendet. Die zweite Methode erfolgt nach erfolgreichem Parsing. Die aus der rohen Log-Zeile extrahierten Parameter werden als Tags festgelegt und für die weitere Verarbeitung in der Log-Pipeline verwendet. Daher liegt der Schwerpunkt nun auf den Label-Filtern.

Der Aussageteil des Filters wird als Prädikat bezeichnet. Er besteht aus folgenden Komponenten: Identifikator der Labels, Operator und Wert. Der Wertetyp kann unterschiedlich sein und hängt von den Eingabedaten der Anfrage ab. Die verwendbaren Operatoren hängen ebenfalls vom jeweiligen Datentyp ab. Die Tabelle 6 enthält eine Übersicht über die verschiedenen Wertetypen und die entsprechenden Vergleichsoperatoren. Zudem können Prädikate miteinander verbunden werden. Mit Hilfe folgender Elemente ist dies möglich: „and“, „|“, „,“ oder einfach ein Leerzeichen[88].

Tabelle 6: Arten von Werten und entsprechende Operatoren von Prädikaten
Quelle: [CH24]

Wertetyp	Beschreibung	Operatoren
String	Der Wert dieses Typs kann in doppelten Anführungszeichen oder umgekehrten Klammern geschrieben werden, z.B. „host“ oder `host`. Tritt häufig beim Filtern des Labels „_error_“ auf. Dies ist ein integriertes Tag, das Log-Einträge mit Parsing- oder Formatierungsfehlern enthält.	=, !=, =~, !~
Dauer	Dieser Typ wird als Folge von Dezimalzahlen dargestellt. Optional können Bruchwerte und Einheiten als Suffix verwendet werden. Verfügbare Zeiteinheiten: „ns“, „us“, „ms“, „s“, „m“ und „h“. Es können auch mehrere Maßeinheiten benutzt werden. Beispiele: 280 ms, 1.4 h oder 3h 45m.	= oder == , != , > oder >= , < oder <=
Zahl	Dies sind Standard-Fließkommazahlen, beispielsweise 345 oder 52.68.	Ähnlich wie die Operatoren, die für den Wertetyp „Dauer“ verwendet werden.
Bytes	Dieser Typ wird in Form von Dezimalzahlen dargestellt. Optional können Bruchwerte und Einheiten als Suffix verwendet werden. Verfügbare Maßeinheiten in Byte: „b“, „kib“, „kb“, „mib“, „mb“, „gib“, „gb“, „tib“, „tb“, „pib“, „pb“, „eib“ und „eb“. Beispiele: 45mb oder 20b usw.	Ähnlich wie die Operatoren, die für den Wertetyp „Dauer“ verwendet werden.

[88] Vgl. [CH24]

Nachfolgend wird ein Beispiel für die Eingabe einer LogQL-Abfrage gezeigt, die den JSON-Parser und die zusätzliche Verwendung eines Label-Filters umfasst (siehe Abb. 13).

```
{realm="zzgl_02"} | json | message_category=~"custom.*"
```

Abbildung 13: LogQL-Anfrage mit Parser und Label-Filter

6.3.5 Transformation von Logzeilen

Nach der erfolgreichen Auswahl von Protokollen nach Selektoren sowie dem Parsing und der Verarbeitung von Einträgen mithilfe von Label-Filtern liegt der Fokus nun auf der Transformation der Logzeilen. Dies ist ein effektives Instrument zur Steigerung der Analyseeffizienz, da nur gewünschte und erforderliche Daten aus den Logs angezeigt werden können.

In der Regel gibt es zwei Möglichkeiten zur Transformation: das Zeilen- und das Label-Format. Die Funktionsweise dieser beiden Konzepte basiert auf einer Template-Engine (das Text-/Template-Format von Go). Diese kann beispielsweise auf Protokollzeilen und ihre Daten zugreifen oder Labels als Variablen verarbeiten. Des Weiteren kann sie Template-Funktionen ausführen. Dazu gehören String-, Math-, JSON-, Kodierungs-, Dekodierungs-, Datum-, Zeitfunktionen usw[89].

Zunächst wird das Zeilenformat betrachtet. Mit diesem Ansatz kann die Vorlage für die Anzeige der Protokollzeile festgelegt werden. Zu diesem Zweck wird der Ausdruck | line_format „{{.label}}“ eingesetzt werden. Außerdem interpretiert LogQL die Labels in diesem Fall als Variablen, sodass sie in die Vorlage eingefügt, dort verwendet und mittels Template-Funktionen geändert werden können[90].

Eine weitere Option – das Label-Format – ermöglicht das Umbenennen, Ändern und Erstellen neuer Tags. Dazu muss der Befehl | label_format new_label="{{.label}}" ausgeführt werden. Außerdem können mehrere Aufträge gleichzeitig bearbeitet werden, da dieser Transformationstyp die Erstellung einer durch Kommas getrennten Liste von Aufgaben erlaubt. Um Labels umzubenennen, müssen auf beiden Seiten des Gleichheitszeichens Identifikatoren stehen. Der erste ist das Ziel, der zweite die Quelle: target=source. Nach Durchführung dieser Operation erhält das Target den neuen Wert der Source. Die Quelle wird aber gelöscht. Es besteht auch die Möglichkeit, den Wert der Source nicht zu entfernen. Dazu muss jedoch eine andere

[89] Vgl. [CH24]
[90] Ebenda

Methode target="{{.source}}" verwendet werden. Wenn das Ziel-Tag noch nicht existiert, wird automatisch ein neues Label erstellt. Die Verwendung von Template-Funktionen ist auch möglich[91]. Ein Beispiel für den Einsatz von Zeilen- und Label-Format ist in Abbildung 14 zu sehen.

```
{realm="zzgl_02"} | json | label_format server="{{.process_host}}" | line_format "{{.server}}"
```

Abbildung 14: Verwendung von Zeilen- und Label-Format in LogQL-Anfrage

6.3.6 Erstellung von Metriken aus Logs

Nach dem Parsing, der Filterung und Formatierung der Logs kann ein weiteres leistungsstarkes Konzept von Loki genutzt werden, nämlich die Umwandlung von Einträgen in Metriken. Mit dieser Fähigkeit können beispielsweise die Quellen mit dem größten Logvolumen für einen bestimmten Zeitraum oder die häufigsten Fehler ermittelt werden. Eine weitere wichtige Funktion ist die Verwendung dieses Telemetrietyps zur Visualisierung. Protokolle können nicht in ihrer ursprünglichen Form dargestellt werden, sondern müssen zur weiteren Verarbeitung in Metriken transformiert werden[92]. Aus diesem Grund wird im nächsten Abschnitt der Schwerpunkt auf dem Konzept der Umwandlung liegen.

Um die Umwandlung durchzuführen, müssen zunächst die numerischen Werte aus den Protokollen extrahiert und anschließend zusammengefasst werden. Der erste Schritt kann mit Hilfe eines Parsers ausgeführt werden. Für den zweiten werden Aggregationsfunktionen verwendet. Generell gibt es zwei Arten der Aggregation: Bereichsvektoraggregation und eingebaute Aggregationsoperatoren[93].

Zunächst werden die Bereichsvektoraggregationen berücksichtigt, bei denen der Bereich als eine Sammlung von Logs oder Labels für einen bestimmten Zeitraum dargestellt wird. Das Zeitintervall kann in den folgenden Einheiten angegeben werden: ms (Millisekunden), s (Sekunden), m (Minuten), h (Stunden), d (Tage), w (Wochen) und y (Jahre). Darüber hinaus gibt es bei diesem Aggregationstyp zwei Unterarten: die Logbereichs- und Unwrapped-Bereichsaggregationen. Die erste gewährleistet die Zusammenfassung der Protokolleinträge für eine bestimmte Zeitspanne, die in einer LogQL-Abfrage nach dem Stream-Selektor oder am Ende der Log-Pipeline stehen kann. Die Funktionen dieses Typs sind nachfolgend aufgeführt:

[91] Vgl. [CH24]
[92] Ebenda
[93] Vgl. ebenda

- rate (range) - berechnet die Anzahl der Logs pro Sekunde;
- count_over_time (range) - rechnet die Anzahl der Einträge eines bestimmten Log-Streams für einen definierten Zeitraum aus;
- bytes_rate (range) - berechnet die Anzahl der Bytes pro Sekunde für jeden Log-Stream. Es kann verwendet werden, um Änderungen im Protokollvolumen zu erkennen;
- bytes_over_time (range) – kalkuliert die Anzahl der Bytes eines Log-Streams für eine bestimmte Zeitspanne. Es kann zur Berechnung des Log-Volumens verwendet werden;
- absent_over_time (range) - überprüft, ob Logs vorhanden sind. Es sendet einen leeren Vektor, wenn mindestens ein Protokoll gefunden wurde. Andernfalls schickt es einen Vektor mit einem Element, dessen Wert eins ist. Dies ist besonders nützlich für Warnmeldungen[94].

Die unten gezeigte LogQL-Abfrage berechnet die Anzahl der Logs pro Minute, die bestimmten Auswahlkriterien entsprechen (siehe Abb. 15).

```
count_over_time(
    {realm="zzgl_02"}
    | json
    | message_category=~"custom.*"
    [1m]
)
```

Abbildung 15: Verwendung der Logbereichs - Aggregationsfunktion in einer LogQL-Abfrage

Bei der Unwrapped-Bereichsaggregation wird zunächst die LogQL-Funktion unwrap() verwendet, um Werte herauszuziehen. Anschließend werden diese mit Aggregationsoperatoren verarbeitet. Zusätzlich können Aufzeichnungen mit den Befehlen „by“ oder „without“ nach ihren Labels gruppiert werden. Die erste Funktion fasst nur Protokolle mit passenden Tags zusammen. Die zweite hingegen sammelt alle Logs außer denen, deren Labels der Bedingung entsprechen[95]. Die einigen Funktionen, die Unwrapped-Bereichsaggregationen ermöglichen, sind in der Tabelle 7 aufgeführt.

[94] Vgl. [CH24]
[95] Ebenda

Tabelle 7: Unwrapped-Bereichsaggregationsfunktionen
Quelle: [CH24]

Aggregationsfunktion	Beschreibung
sum_over_time (unwraped-range)	Summiert alle Werte im Intervall.
avg_over_time (unwraped-range)	Berechnet den Durchschnittswert aller Elemente im Intervall.
max_over_time (range)	Gibt den Maximalwert im Intervall zurück.
min_over_time (unwraped-range)	Gibt den Minimalwert im Intervall zurück.
first_over_time (unwraped-range)	Gibt den ersten Wert im Intervall zurück.
last_over_time (unwraped-range)	Gibt den letzten Wert im Intervall zurück.
stdvar_over_time (unwraped-range)	Gibt die Varianz aller Werte im Intervall zurück.
stddev_over_time (unwraped-range)	Gibt die Standardabweichung aller Werte im Intervall zurück.
quantile_over_time (scalar, unwraped-range)	Gibt einen bestimmten Quantil zurück.
absent_over_time (unwraped-range)	Überprüft, ob Werten vorhanden sind. Es sendet einen leeren Vektor, wenn mindestens ein Wert gefunden wurde. Andernfalls schickt es einen Vektor mit einem Element, dessen Wert eins ist. Besonders nützlich für Warnmeldungen.

Als Nächstes werden die integrierten Aggregationsoperatoren betrachtet. Dabei handelt es sich um Operationen, die die Zusammenfassung von Protokollen und deren Daten gewährleisten. Als Ergebnis wird ein neuer Vektor erstellt, der die aggregierten Daten enthält. Deshalb wird die Anzahl der Elemente darin reduziert. Zu diesen Operatoren gehören die folgenden:

- sum - summiert alle Elemente des Vektors mit bestimmten Labels;
- avg - berechnet den Mittelwert aus den Elementen des Vektors mit bestimmten Labels;
- min - wählt das Minimum aus den Elementen des Vektors mit bestimmten Labels aus;
- max - wählt das Maximum aus den Elementen des Vektors mit bestimmten Labels aus;
- count - berechnet die Anzahl der Elemente im Vektor;
- topk - wählt die k größten Elemente aus der Stichprobe aus;
- bottomk - wählt die k kleinsten Elemente aus der Stichprobe aus;
- sort - gebt einen in aufsteigender Reihenfolge sortierten Vektor aus der Stichprobe zurück;
- sort_desc – gebt einen nach absteigender Reihenfolge sortierten Vektor aus der Stichprobe zurück[96].

[96] Vgl. [CH24]

Ein Beispiel für die Verwendung des integrierten Aggregationsoperators, der die zehn häufigsten Protokolleinträge nach Schweregrad „error“ anzeigt, ist in Abbildung 16 dargestellt.

```
topk(10,
  count_over_time(
        {realm="zzgl_02"}
        | json
        | message_severity="error"
        [5m]
    )
)
```

Abbildung 16: Verwendung des integrierten Aggregationsoperators in einer LogQL-Abfrage

6.4 Visualisierung von Logdaten

Nach dem Filtern, Analysieren und Aggregieren der Protokolle kann eine weitere leistungsstarke Komponente zum Einsatz kommen. Dabei handelt es sich um die Visualisierung der Aufzeichnungen. Zusammengefasste Logs in Form von Diagrammen oder Histogrammen können die Effizienz der Analyse erheblich steigern. Deshalb steht dieses Thema im Mittelpunkt des folgenden Abschnitts.

Datenvisualisierung ist der Prozess der Umwandlung von Daten in Grafiken, Diagramme, Karten, Infografiken usw. Die Hauptaufgabe besteht darin, Daten in einer verständlichen Form zu organisieren. Darüber hinaus bietet sie die Möglichkeit, wichtige Informationen einem breiten Publikum (z.B. Kunden, Kollegen usw.) ohne technische Kenntnisse nachvollziehbar zu machen. Die Datenvisualisierung kann auch dabei helfen, bestimmte Muster, Trends und Abweichungen besser zu verstehen. Zudem kann dies für die Entscheidungsfindung sehr nützlich sein[97].

Die Verwendung von Visualisierungen ist vielfältig: Bestandsüberwachung, Marketing, Finanzen, Verwaltung von Kundensupport-Tickets[98]. Sie spielt auch eine wichtige Rolle bei der Analyse von Protokollen. In der Regel kann die Anzahl der Logs Tausende pro Tag erreichen. Daher ist es schwierig, alle Meldungen zu bearbeiten. Die Visualisierung hilft dabei. Sie ermöglicht einen einfachen Überblick über die Leistung und den Zustand des Systems, die Identifizierung von Trends und Anomalien, das rechtzeitige Erkennen von Fehlern usw. Außerdem ist die Verwendung von Datenvisualisierung und Alarmmanagement eine leistungsstarke Kombination für die Überwachung der Plattformstabilität. Wenn ein Entwickler beispielsweise eine

[97] Vgl. [Cou26]
[98] Ebenda

Meldung erhält, dass die Anzahl der Fehler das Limit überschritten hat, kann er anhand eines Diagramms erkennen, welches Problem wie oft und wann zum ersten Mal aufgetreten ist[99].

Wie bereits erwähnt, müssen Logs zur Visualisierung zunächst in Metriken umgewandelt werden. Standardmäßig erstellt Grafana nach der Konvertierung automatisch ein Diagramm in Form einer Zeitreihe (Spalten, Punkte oder Linien). Die Visualisierung kann jedoch verbessert werden und andere grafische Darstellungen können verwendet werden. Dazu gibt es zwei Möglichkeiten. Die erste besteht darin, das bereits erstellte Diagramm im Menü „Dashboard" hinzuzufügen und dort die weiteren Schritte auszuführen. Die zweite ist, ein neues Panel im bereits erwähnten Fenster zu erstellen. Die Vorgänge wie Parsing, Filtern usw. können dort ausgeführt werden.

Grafana Labs bietet zahlreiche Visualisierungsmöglichkeiten. Darüber hinaus können zusätzliche Plugins heruntergeladen werden[100]. Der Grund dafür ist, dass verschiedene Datentypen unterschiedliche grafische Darstellungen erfordern. Nur somit können die Daten am effektivsten analysiert werden.

Für kategoriale Daten eignet sich beispielsweise ein „Bar Chart" am besten. Um zu zeigen, wie sich die Daten vom Maximal- oder Minimalwert unterscheiden, kann eine radiale „Gauge" oder eine horizontale bzw. vertikale „Bar Gauge" verwendet werden. Ein "Pie Chart" kann zum Einsatz kommen, wenn aggregierte Werte vorliegen und deren Zusammenhang dargestellt werden soll (z. B. in Prozent). Eine "Heatmap" kann zeigen, wie sich die Daten im Laufe der Zeit verteilt haben. Eine „State Timeline" stellt dar, wie sich diskrete Daten im Zeitverlauf verändern[101].

[99] Vgl. [Bet26]
[100] Vgl. [Gra26 c]
[101] Vgl. ebenda

7 Aufsetzen des Prototyps

Um die bisher erörterten Konzepte im Praxisansatz zu betrachten, wurde ein Prototyp zur Anbindung des SFCC Log Centers an Grafana Loki erstellt. Dieser Prozess besteht aus zwei Hauptschritten. Der erste umfasst die Konfiguration in Grafana, der zweite – in Log Center. Im nachfolgenden Abschnitt werden diese Etappen ausführlich erläutert.

Log Streaming vom Log Center nutzt AWS Kinesis Firehose als Basisdienst für die Datenübertragung. Seine Einrichtung ist in Grafana Cloud möglich. Dazu wurde Terraform verwendet, das automatisch einen Stream erstellt. Außerdem wurde ein Zugriffstoken mit einer Gültigkeitsdauer von 30 Tagen generiert (siehe Abb. 17).

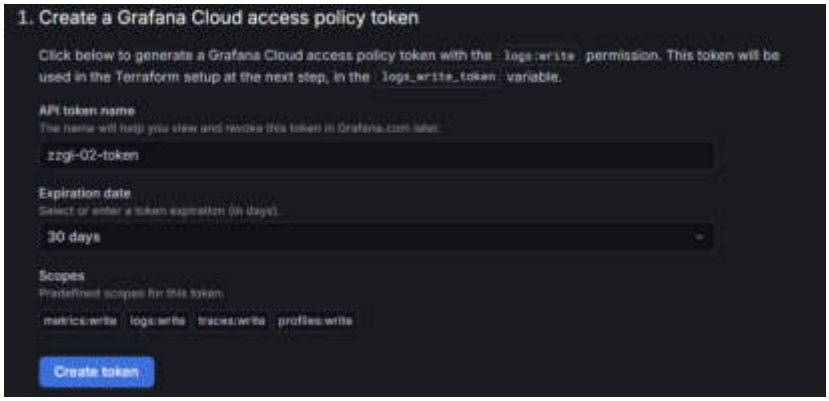


Abbildung 17: Generierung eines Zugriffstokens in Grafana Cloud

Zudem generiert Grafana Cloud die folgenden Parameter: Zielendpunkt und Loki-Instanz-ID (siehe Abbildung 18). Der erste gibt den Speicherort an, an den die Logs übertragen werden. Der zweite Parameter und der generierte API-Token werden für die Authentifizierung verwendet.

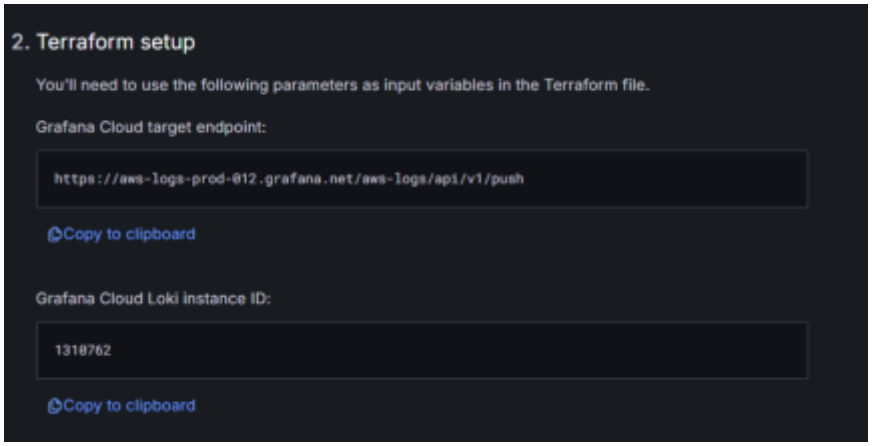


Abbildung 18: Terraform-Setup

Als Nächstes wurde ein neuer Log-Stream im Log-Center erstellt (siehe Abbildung 19). Für dessen Konfiguration wurde ein externer Dienst ausgewählt (Grafana Loki) und folgende Felder ausgefüllt:

- Name des Streams;
- URL-Adresse des Grafana Cloud-Endpunkts;
- Benutzer-ID, die der Loki-Instanz-ID entspricht;
- API-Token, das im Grafana generiert wurde.

Außerdem wurden „Realm“ und „Alert Time Interval“ konfiguriert. Schließlich wurde ein Parameter mit dem Präfix „lbl_“ eingegeben, der bei der Übertragung von Protokollen zu den Metadaten hinzugefügt wird. Diesmal wurden die Filterparameter ignoriert, um die Auswahlmöglichkeiten in Grafana Loki zu testen.

Abbildung 19: Konfiguration des neuen Log-Streams

Die Ergebnisse der Nutzung der Filter- und Darstellungsfunktionen von Grafana Loki sind in den Anlagen 3 – 8 zu finden.

8 Vergleich von Log Center und externem Dienst

Die Unterschiede zwischen den Eigenschaften von Log Center und Grafana Loki sind in Tabelle 8 dargestellt.

Tabelle 8: Vergleich von Log Center und externem Dienst

Vergleichskriterien	Log Center	Grafana Loki
Aufbewahrungsfrist	Die Logs werden nur 14 Tage lang gespeichert.	Die Aufbewahrungsfrist ist standardmäßig deaktiviert, sodass Logs unbegrenzt gespeichert werden können. Außerdem besteht die Möglichkeit, diesen Zeitraum anzupassen. Dies kann mit Hilfe des Compactors erfolgen. Es ist zu beachten, dass die Mindestaufbewahrungsfrist 24 Stunden betragen muss[102].
Visualisierungsmöglichkeiten	Das Dashboard zeigt nur die Trace-Stacks von Salesforce-Instanzen in Form eines Histogramms an.	Standardmäßig werden Zeitreihendiagramme angezeigt. Es besteht jedoch die Möglichkeit, Daten in einem anderen Format anzuzeigen. Grafana bietet zahlreiche Diagrammtypen, Histogramme usw. Darüber hinaus können zusätzliche Plugins heruntergeladen werden. Vor der Visualisierung müssen die Logs jedoch eine Reihe von Filter- und Konvertierungsschritten durchlaufen.
Filtermöglichkeiten	Mit der Log Center Search können Logs effektiv gefiltert werden. Zunächst können Einträge nach bestimmten Parametern sortiert werden, beispielsweise nach Schweregrad, Bereichstyp, Protokolltyp usw. Die Aufzeichnungen können auch für eine bestimmte Zeitspanne ausgewählt werden. Eine weitere Gelegenheit besteht darin, eine Suche anhand von Schlüsselwörtern durchzuführen. Es werden sowohl die grundlegende als auch die erweiterte Suche unterstützt.	Grafana Loki bietet verschiedene Möglichkeiten zur Verarbeitung von Logs, die effektive Analyse gewährleisten. Dazu gehören: die Auswahl nach Stream-Selektor, die Verwendung von Parsern, die Suche nach bestimmten Texten oder Übereinstimmungen mit regulären Ausdrücken in der Log-Pipeline, die Transformation von Zeilenformaten und die Umwandlung von Einträgen in Metriken mithilfe von Aggregationsoperatoren.

[102] Vgl. [Gra26 d]

Benachrichtigungsmöglichkeiten bei Alarm	Bei kritischen Fehlern („fatal“ Schwergrad) können Benachrichtigungen an die angegebene E-Mail-Adresse gesendet werden.	Grafana Labs ermöglicht die Verwendung von IRM zur Aufrechterhaltung der Stabilität von Anwendungen, Systemen usw. Mit seiner Hilfe können Benachrichtigungsregeln erstellt werden, die per E-Mail oder über Grafana OnCall über Probleme informieren. Zusätzlich kann der integrierte Chatbot bei einem Unfall Kanäle für Vorfälle und Bereiche für die Zusammenarbeit erstellen, was für die Arbeit eines großen Teams praktisch ist.
Erweiterte Analysefunktionen	Nicht vorhanden.	Grafana Labs bietet maschinelles Lernen-Tools (ML) zur Vorhersage von Metriken an. Diese können prognostizieren, wie sich Metriken in Zukunft entwickeln werden. Dies ist für die Kapazitätsplanung sehr nützlich. Ein weiterer Vorteil des Einsatzes von maschinellem Lernen ist die Erkennung von Anomalien.
Kosten	Für die Nutzung der Funktionen des SFCC Log Centers sind keine zusätzlichen Kosten erforderlich.	Die Enterprise-Lizenz kostet 25.000 Dollar pro Jahr.

9 Fazit

Für den sicheren und stabilen Betrieb eines Online-Shops ist die Verwendung von Logs unverzichtbar. Aufgrund der großen Anzahl generierter Protokolle ist deren Analyse jedoch schwierig und es besteht die Wahrscheinlichkeit, wichtige Einträge zu übersehen. Daher war es das Ziel der Arbeit, zu untersuchen, wie Logs aus Salesforce B2C an einen externen Dienst zur weiteren Analyse übertragen werden können.

Log Center bietet die Möglichkeit, Log Steam mit Analysetools zu erstellen und Aufzeichnungen auf diese Weise zu übertragen. Es gibt eine Reihe der verfügbaren Anbieter, aber für diese Arbeit wurde Grafana Loki ausgewählt. Dort bestehen die Möglichkeiten, Logs zu filtern, zu parsen und im gewünschten Zeilenformat anzuzeigen. Darüber hinaus ist eine Umwandlung anhand von Metriken zur Visualisierung möglich. Außerdem wurde auch die Architektur von Grafana Loki erläutert, um Verständnis dafür zu haben, wie Logs dort abgelegt werden. Von diesem Punkt ist die Verarbeitung von Aufzeichnungen abhängig.

Schließlich wurde eine Tabelle erstellt, in der die Aspekte von Log Center und Grafana Loki verglichen werden. Diese Übersicht wurde zusammengestellt, um zu verdeutlichen, wann die Verwendung eines der beiden Verfahren sinnvoller ist.

Abschließend ist anzumerken, dass dieses Wissen für dotSource wichtig ist. Gründe dafür sind die Gewährleistung der Sicherheit und Stabilität des Systems und damit eine Kostensenkung. Diese Aspekte sind sowohl für neue als auch für bestehende Kunden von entscheidender Bedeutung.

Quellenverzeichnis

[Bet26] Better Stack: "What is Log Visualization? Getting Started and Best Practices". Abgerufen am 06.03.2026 von: https://betterstack.com/community/guides/logging/log-visualization/

[CH24] Chapman, P., Holmes, P.: "Observability with Grafana: Monitor, control, and visualize your Kubernetes and cloud platforms using the LGTM stack", Packt Publishing, Birmingham , 2024

[Cou26] Coursera: "What is Data Visualization? Definition, Tools, & Examples". Abgerufen am 06.03.2026 von: https://www.coursera.org/articles/data-visualization

[CSP13] Chuvakin, A., Schmidt, K., Phillips, C.: "Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management", Syngress, Waltham, 2013

[Gra26 a] Grafana Labs: "Loki deployment modes". Abgerufen am 27.02.2026 von: https://grafana.com/docs/loki/latest/get-started/deployment-modes/#loki-deployment-modes

[Gra26 b] Grafana Labs: "Loki architecture". Abgerufen am 02.03.2026 von: https://grafana.com/docs/loki/latest/get-started/architecture/

[Gra26 c] Grafana Labs: "Visualizations". Abgerufen am 06.03.2026 von: https://grafana.com/docs/grafana/latest/visualizations/panels-visualizations/visualizations/

[Gra26 d] Grafana Labs: "Log retention". Abgerufen am 08.03.2026 von: https://grafana.com/docs/loki/latest/operations/storage/retention/#log-retention

[Int26] Integrate.io: "The Complete Guide to FTP, FTPS, SFTP, and SCP". Abgerufen am 20.01.2026 von: https://www.integrate.io/blog/the-complete-guide-to-ftp-ftps-sftp-and-scp/

[JP26] Jem Products: "Log Management Is Crucial for E-Commerce Success, and Here’s Why". Abgerufen am 16.01.2026 von: https://jem-products.com/log-management-is-crucial-for-e-commerce-success-and-heres-why/

[Lee24] Lee, C.-C.: "Leveraging Large Language Models and BERT for Log Parsing", MDPI, Basel, Schweiz, 2024

[Log26] LogCentral: "Should you use TCP or UDP to transmit your syslogs?". Abgerufen am 19.01.2026 von: https://logcentral.io/blog/should-you-use-tcp-or-udp-to-transmit-your-syslogs

[Med26 a] Medium: "What Is Business Manager In SFCC (Salesforce Commerce Cloud)". Abgerufen am 26.01.2026 von: https://medium.com/@princedev007/what-is-business-manager-in-sfcc-salesforce-commerce-cloud-c6dc0db90512

[Med26 b] Medium: "Seamless Data Delivery with Amazon Kinesis Data Firehose". Abgerufen am 17.02.2026 von: https://aws.plainenglish.io/seamless-data-delivery-with-amazon-kinesis-data-firehose-be7d644a7cad

[Rsy26] Rsyslog documntation: "Encrypting Syslog Traffic with TLS (SSL)". Abgerufen am 20.01.2026 von: https://www.rsyslog.com/doc/tutorials/tls_cert_summary.html

[Rüc22] Rücker, N.: "Logdatenverarbeitung - Veranderungen, Herausforderungen und Moglichkeiten", Dissertation, der Technischen Fakultät der Friedrich-Alexander-Universität Erlangen-Nürnberg, o.O, 2022

[Sal26 a] Salesforce: "Log Files". Abgerufen am 26.01.2026 von: https://developer.salesforce.com/docs/commerce/b2c-commerce/guide/b2c-log-files-overview.html

[Sal26 b] Salesforce: "Governance and Quotas". Abgerufen am 11.02.2026 von: https://developer.salesforce.com/docs/commerce/b2c-commerce/guide/b2c-governance-and-quotas.html

[Sal26 c] Salesforce Help: "Security Event Auditing in B2C Commerce". Abgerufen am 11.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_security_event_auditing.htm&type=5

[Sal26 d] Salesforce Help: "Batch Processing in B2C Commerce". Abgerufen am 11.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_batch_processing.htm&type=5

[Sal26 e] Salesforce Help: "Log Center". Abgerufen am 16.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_log_center.htm&type=5

[Sal26 f] Salesforce B2C Commerce: "Concepts and Terminology". Abgerufen am 16.02.2026 von: https://sfcclearning.com/infocenter/content/b2c_commerce/topics/getting_started/b2c_platform_overview.php

[Sal26 g] Salesforce Help: "Log Center Dashboard". Abgerufen am 16.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_log_center_dashboard.htm&type=5

[Sal26 h] Salesforce Help: "Log Center Search". Abgerufen am 16.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_log_center_search.htm&type=5

[Sal26 i] Salesforce Help: "Keyword Search in Log Center". Abgerufen am 16.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_keyword_search_in_log_center.htm&type=5

[Sal26 j] Salesforce, Help: "Log Streaming for Enhanced Monitoring". Abgerufen am 17.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_log_streaming_faq.htm&type=5

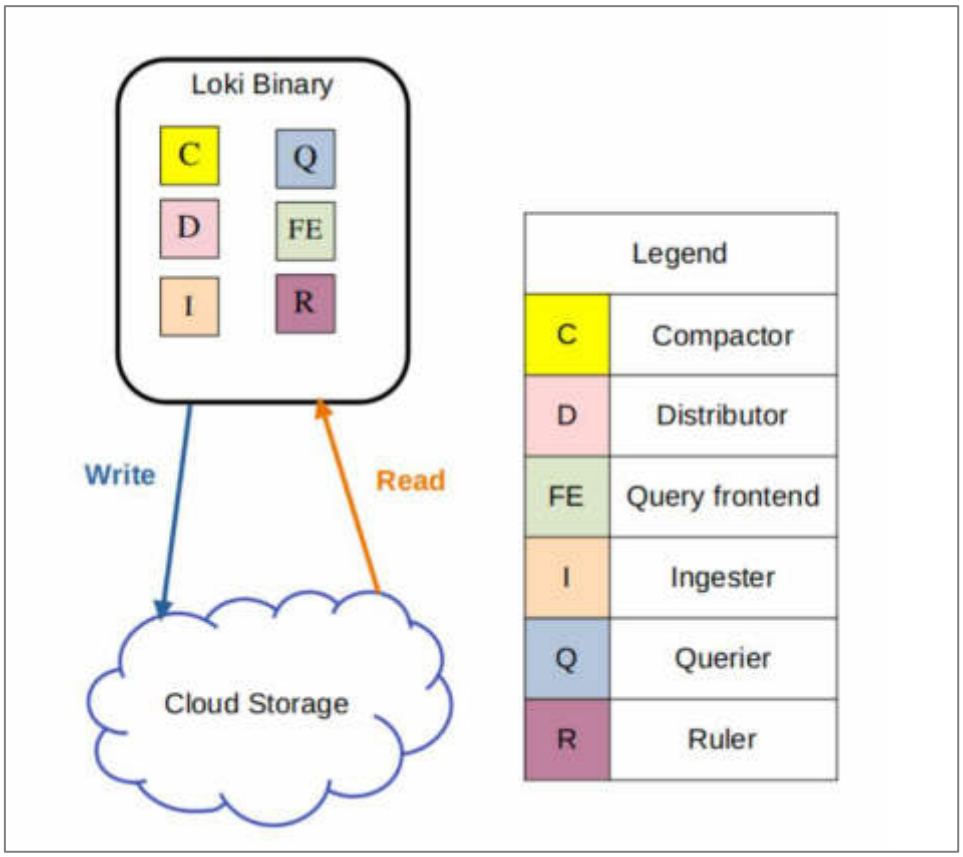
[Sal26 k] Salesforce Help: "Set Up Log Streaming to Third-Party Monitoring Tools". Abgerufen am 18.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_set_up_log_streaming.htm&type=5

[Sal26 l] Salesforce Help: "Security Logs Streaming". Abgerufen am 18.02.2026 von: https://help.salesforce.com/s/articleView?id=cc.b2c_security_logs_streaming.htm&type=5

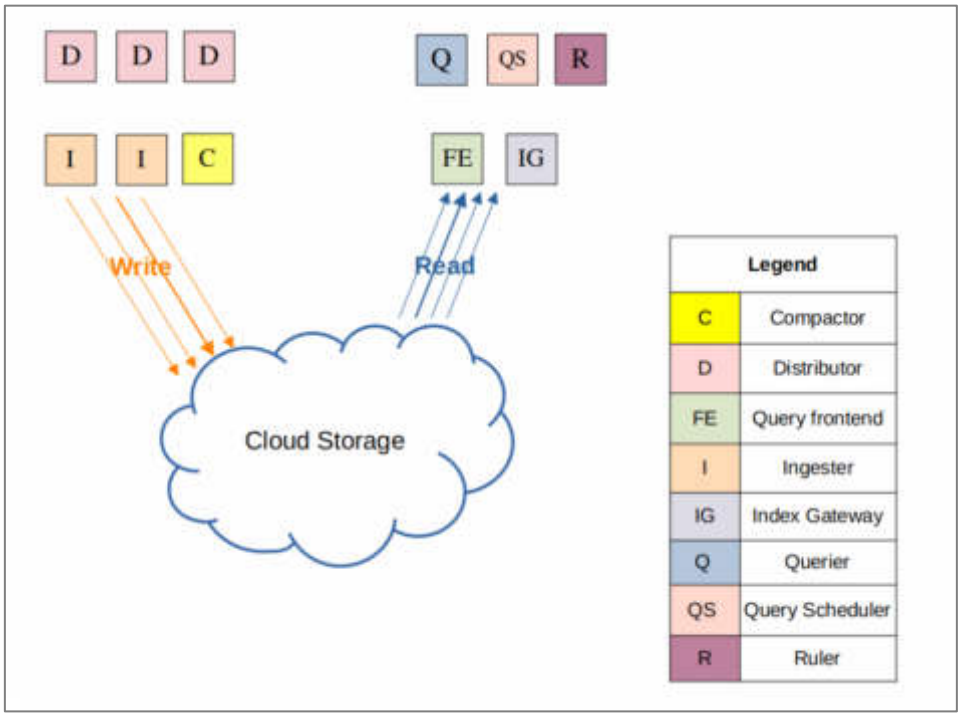
[Sys26] Syslog-ng: "Why use a http()-based destination in syslog-ng?". Abgerufen am 20.01.2026 von: https://www.syslog-ng.com/community/b/blog/posts/why-use-a-http--based-destination-in-syslog-ng

Anlagenverzeichnis

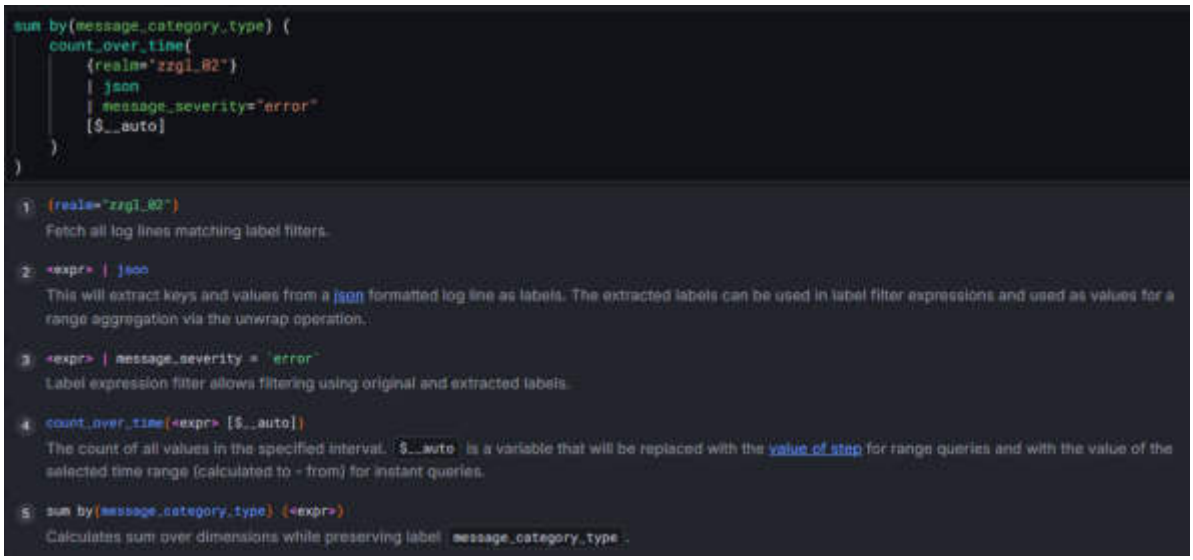
Anlage 1: Monolithischer Modus in Grafana Loki XII
Anlage 2: Mikroservicebasierter Modus in Grafana Loki XII
Anlage 3: LogQL-Abfrage zum Erstellen von Metriken, die Anzahl verschiedener Fehlerkategorien anzeigen XIII
Anlage 4: Automatisch erstellter Zeitreihendiagramm XIII
Anlage 5: Gefundenes Log mit der Fehlerkategorie „job“ XIV
Anlage 6: Gefundenes Log mit der Fehlerkategorie „system“..... XIV
Anlage 7: Gefundenes Log mit der Fehlerkategorie „custom“ XV
Anlage 8: Optimiertes Diagramm mit entsprechendem Namen und Beschreibung XV



Anlage 1: Monolithischer Modus in Grafana Loki
Quelle: [Gra26 a]

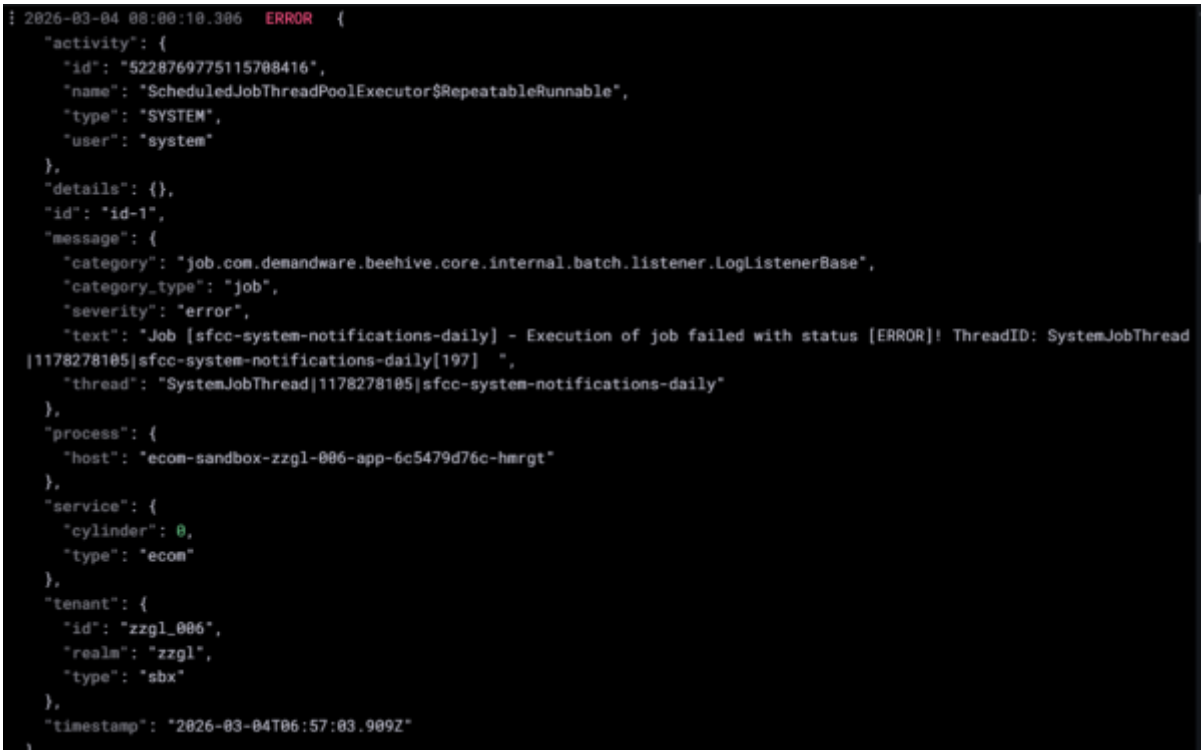


Anlage 2: Mikroservicebasierter Modus in Grafana Loki
Quelle: [Gra26 a]

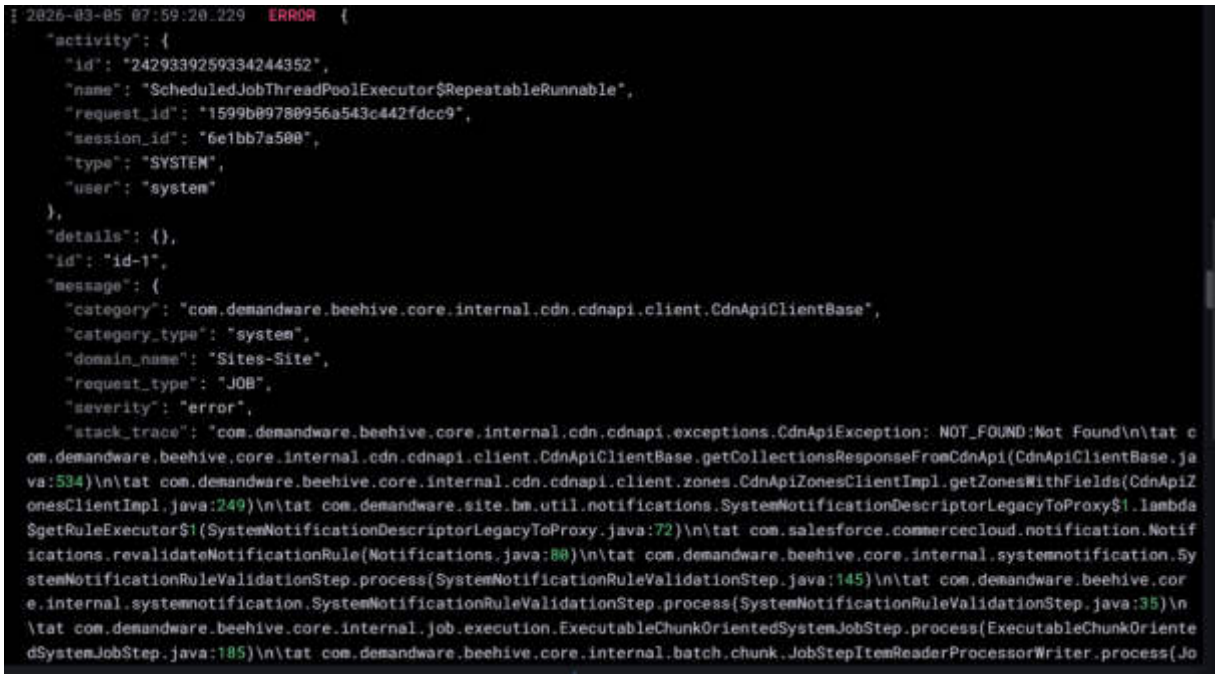


Anlage 3: LogQL-Abfrage zum Erstellen von Metriken, die Anzahl verschiedener Fehlerkategorien anzeigen

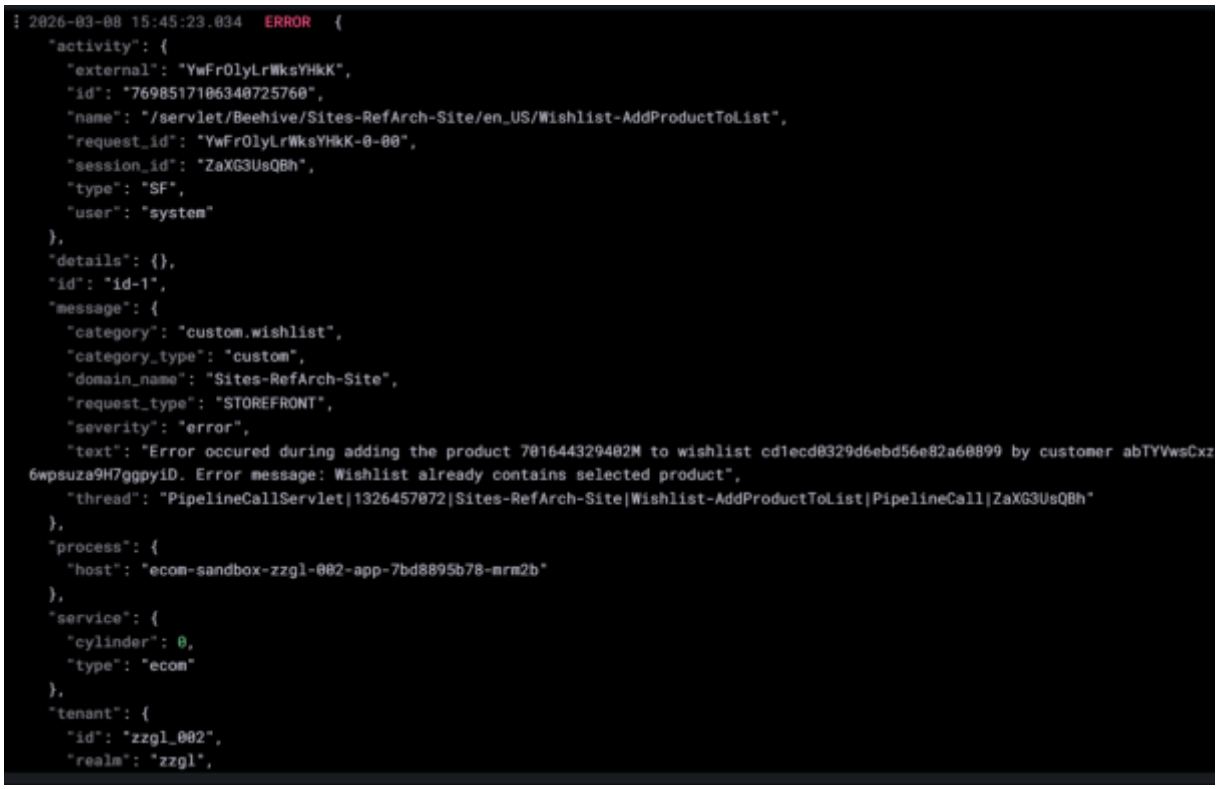
Anlage 4: Automatisch erstellter Zeitreihendiagramm



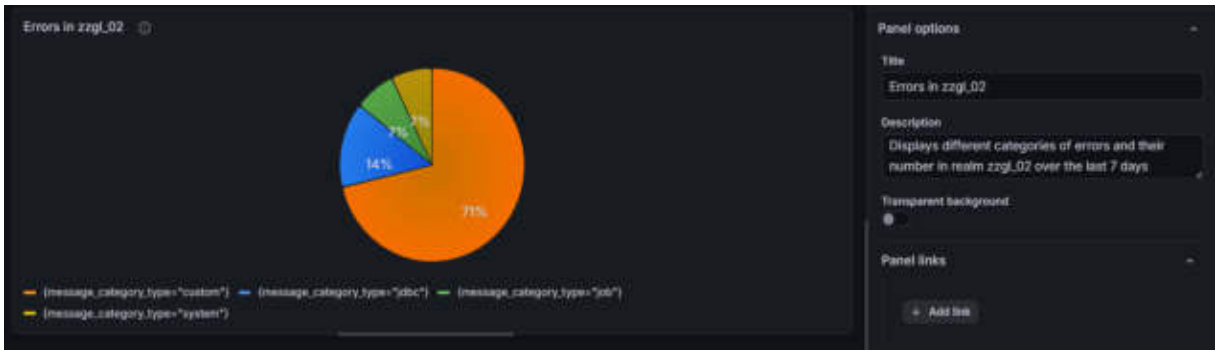
Anlage 5: Gefundenes Log mit der Fehlerkategorie „job“



Anlage 6: Gefundenes Log mit der Fehlerkategorie „system“



Anlage 7: Gefundenes Log mit der Fehlerkategorie „custom“



Anlage 8: Optimiertes Diagramm mit entsprechendem Namen und Beschreibung