Cluster Logging mit Loki

Die kontinuierliche Überwachung eines Kubernetes-Clusters erfordert ein effizientes Logmanagement, welches das zentrale Sammeln, Speichern und Analysieren von Protokolldaten ermöglicht. In Kubernetes-Umgebungen wird hierfür traditionell ein Stack aus Logstash oder Fluentd zur Datenerfassung, Elasticsearch als Speichersystem sowie Kibana oder Graylog zur Analyse und Visualisierung eingesetzt. Neben diesem etablierten Ansatz gewinnt der Loki-Grafana-Stack zunehmend an Bedeutung, da er eine ressourcenschonendere und leichter implementierbare Alternative darstellt.

Loki ist ein Log-Aggregationssystem, das speziell für die Speicherung und Abfrage von Logdaten in Cloud-nativen Architekturen konzipiert wurde. Es verfolgt einen multidimensionalen, labelbasierten Ansatz zur Indizierung und nutzt ein binäres, von externen Abhängigkeiten weitgehend freies Datenformat. Vom Hersteller Grafana Labs wird Loki als „Prometheus-ähnlich, jedoch für Logdaten“ charakterisiert. Im Gegensatz zu Prometheus, das auf die Erhebung von Metriken fokussiert ist, verarbeitet Loki ausschließlich Protokolldaten und setzt hierbei auf ein Push-basiertes Übertragungsverfahren.

Während Kibana und Graylog ein umfangreicheres Funktionsspektrum und fortgeschrittene Analysemöglichkeiten bereitstellen, zeichnet sich Loki durch eine geringere Komplexität und eine beschleunigte Bereitstellung aus, was insbesondere für Entwicklungs- und Testumgebungen vorteilhaft ist.

Ein wesentliches Merkmal von Loki ist die ausschließliche Indizierung von Metadaten, vergleichbar mit dem Vorgehen in Prometheus. Dies reduziert den Ressourcen- und Verwaltungsaufwand im Vergleich zu einer Volltextindizierung erheblich. In Kubernetes-Umgebungen werden primär Labels indiziert, sodass sich Logdaten konsistent mit den Metadaten der jeweiligen Anwendungen verwalten lassen.

Index und Chunks - Loki Datenformat

Loki speichert seine Daten bevorzugt in einem zentralisierten Objektspeicher-Backend, wie beispielsweise Amazon Simple Storage Service (S3), Google Cloud Storage (GCS), Azure Blob Storage oder vergleichbaren Systemen. Für diesen Betriebsmodus wird der sogenannte Index Shipper (kurz: Shipper) als Adapter eingesetzt. Diese Betriebsart wurde mit der Version Loki 2.0 allgemein verfügbar und zeichnet sich durch hohe Effizienz, geringe Kosten und einfache Handhabung aus. Sie stellt den aktuellen Standard dar und bildet zugleich die Grundlage für die zukünftige Weiterentwicklung.

Falls kein Zugriff auf einen Objektspeicher gegeben ist, kann in Test- oder Entwicklungsumgebungen (beispielsweise in Trainings-Labs) ersatzweise lokaler Speicher genutzt werden.

Loki verwaltet zwei grundlegende Datentypen: **Index** und **Chunks**.

Index

Der Index dient als Inhaltsverzeichnis und enthält Verweise auf die Speicherorte der Protokolldaten für eine definierte Gruppe von Labels.

Indexformate Zur Speicherung des Index können unterschiedliche Datenbanken als Backend verwendet werden. Für lokale Setups unterstützt Loki zwei Indexformate im Single Store-Modus:

- **TSDB (Time Series Database):**
  Die TSDB wurde ursprünglich für Prometheus zur Verwaltung von Zeitreihendaten entwickelt und stellt das empfohlene Indexformat für Loki dar. Sie ist erweiterbar, leistungsfähig und bietet zahlreiche Vorteile gegenüber dem veralteten BoltDB-Index. Neue Speicherfunktionen in Loki stehen ausschließlich in Verbindung mit der TSDB zur Verfügung.
- **BoltDB:**
  Bolt ist ein in Go implementierter, transaktionaler Low-Level-Key-Value-Store. Aufgrund fehlender Replikationsunterstützung und eingeschränkter Funktionalität gilt BoltDB als veraltet (deprecated) und ist primär für Entwicklungs- und Testumgebungen geeignet.

Um Indexdateien (unabhängig vom Format: TSDB oder BoltDB) konsistent wie Chunk-Dateien im Objektspeicher zu verwalten, wird ebenfalls der Index Shipper eingesetzt. Darüber hinaus unterstützt Loki auch Cloud-Datenbanken wie BigTable und DynamoDB als Index-Backends.

Chunks

Die eigentlichen Logdaten werden in komprimierter und segmentierter Form im sogenannten Chunk Storage gespeichert. Ein Chunk stellt dabei einen Container für die Log-Einträge einer definierten Label-Gruppe dar und umfasst die Logzeilen eines spezifischen Streams innerhalb eines begrenzten Zeitraums.

Chunk-Formate Wie beim Index können auch für die Chunks verschiedene Speicher-Backends verwendet werden. Aufgrund des typischerweise größeren Datenvolumens empfiehlt sich der Einsatz eines Objektspeichers (z. B. Ceph Object Store, Amazon S3, Azure Blob Storage, Google Cloud Storage oder Swift). Der Objektspeicher übernimmt in der Regel auch die Replikation sowie die automatische Löschung abgelaufener Chunks. Für kleinere Projekte oder Entwicklungsumgebungen ist der Einsatz eines lokalen Dateisystems jedoch ebenfalls möglich.

Warum Loki

Die herkömmliche Methode für das Sammeln und Verwalten von Protokolldaten in Kubernetes-Umgebungen basiert auf einem zentralisierten Logmanagement, das in der Regel durch einen sogenannten Logging-Stack realisiert wird. Dieser besteht typischerweise aus drei Kernkomponenten:

- **Datenerfassung:** Fluentd oder Logstash
- **Datenspeicherung:** Elasticsearch
- **Datenvisualisierung:** Graylog oder Kibana

Neben dieser etablierten Architektur hat sich mit der Kombination aus Loki und Grafana Alloy ein alternativer, ressourcenschonenderer OpenTelemetry Collector etabliert.

Loki bietet im Vergleich zu traditionellen Log-Aggregationssystemen eine Reihe spezifischer Vorteile:

- Verzicht auf Volltextindizierung: Statt vollständiger Textindizes speichert Loki komprimierte, unstrukturierte Protokolldaten und indiziert ausschließlich Metadaten. Dies vereinfacht die Handhabung und reduziert die Betriebskosten erheblich. Die Indexierung beschränkt sich auf Metadaten, ähnlich dem Ansatz von Prometheus, wodurch der Verwaltungsaufwand signifikant reduziert wird.
- Integration in bestehende Label-Systeme: Log-Streams werden anhand derselben Labels gruppiert und indiziert, die auch in Prometheus verwendet werden. Dies ermöglicht einen nahtlosen Wechsel zwischen Metriken und Protokolldaten.
- Optimierte Unterstützung für Kubernetes: Loki eignet sich besonders für die Speicherung von Protokolldaten aus Kubernetes-Pods, da relevante Metadaten wie Pod-Labels automatisch erfasst und indexiert werden.
- Native Integration in Grafana: Seit Grafana Version 6.0 ist eine direkte Unterstützung von Loki gegeben.
- Einfache Architektur: Loki ist ein binäres, von externen Abhängigkeiten weitgehend freies System, das eine unkomplizierte Bereitstellung ermöglicht.
- Kosteneffiziente Speicherung: Durch die Nutzung gängiger und kostengünstiger Objektspeicher entfällt die aufwendige Administration eines Elasticsearch-Clusters.

Obwohl Kibana und Graylog ein umfangreicheres Funktionsspektrum bereitstellen, zeichnet sich Loki durch seine geringere Komplexität und den reduzierten Ressourcenbedarf aus, was ihn insbesondere für Entwicklungs- und Testumgebungen prädestiniert.