# Bewährte Methoden für die Leistungsoptimierung AWS Glue für Apache Spark-Jobs

# : Bewährte Methoden für die Leistungsoptimierung AWS Glue für Apache Spark-Jobs

# Table of Contents

# Bewährte Methoden für die Leistungsoptimierung AWS Glue für Apache Spark-Jobs

Roman Myers, Takashi Onikura und Noritaka Sekiyama, Amazon Web Services ()AWS

Dezember 2023 (Dokumentverlauf)

AWS Glue bietet verschiedene Optionen zur Leistungsoptimierung. Dieses Handbuch definiert wichtige Themen für die Optimierung AWS Glue von Apache Spark. Anschließend bietet es eine grundlegende Strategie, die Sie bei der Optimierung dieser Tools AWS Glue für Apache Spark-Jobs befolgen können. In diesem Handbuch erfahren Sie, wie Sie Leistungsprobleme anhand der Interpretation der unter verfügbaren Kennzahlen identifizieren können AWS Glue. Integrieren Sie anschließend Strategien zur Lösung dieser Probleme, zur Maximierung der Leistung und zur Minimierung der Kosten.

In diesem Handbuch werden die folgenden Optimierungsmethoden behandelt:

- Skalieren Sie die Clusterkapazität
- Verwenden Sie die neueste AWS Glue Version
- Reduzieren Sie den Umfang der gescannten Daten
- Aufgaben parallelisieren
- Minimieren Sie den Planungsaufwand
- Optimieren Sie Shuffles
- Optimiere benutzerdefinierte Funktionen

# Wichtige Themen in Apache Spark

In diesem Abschnitt werden die grundlegenden Konzepte und Schlüsselthemen von Apache Spark für die Optimierung AWS Glue der Apache Spark-Leistung erläutert. Es ist wichtig, dass Sie sich mit diesen Konzepten und Themen vertraut machen, bevor Sie über Optimierungsstrategien aus der Praxis sprechen.

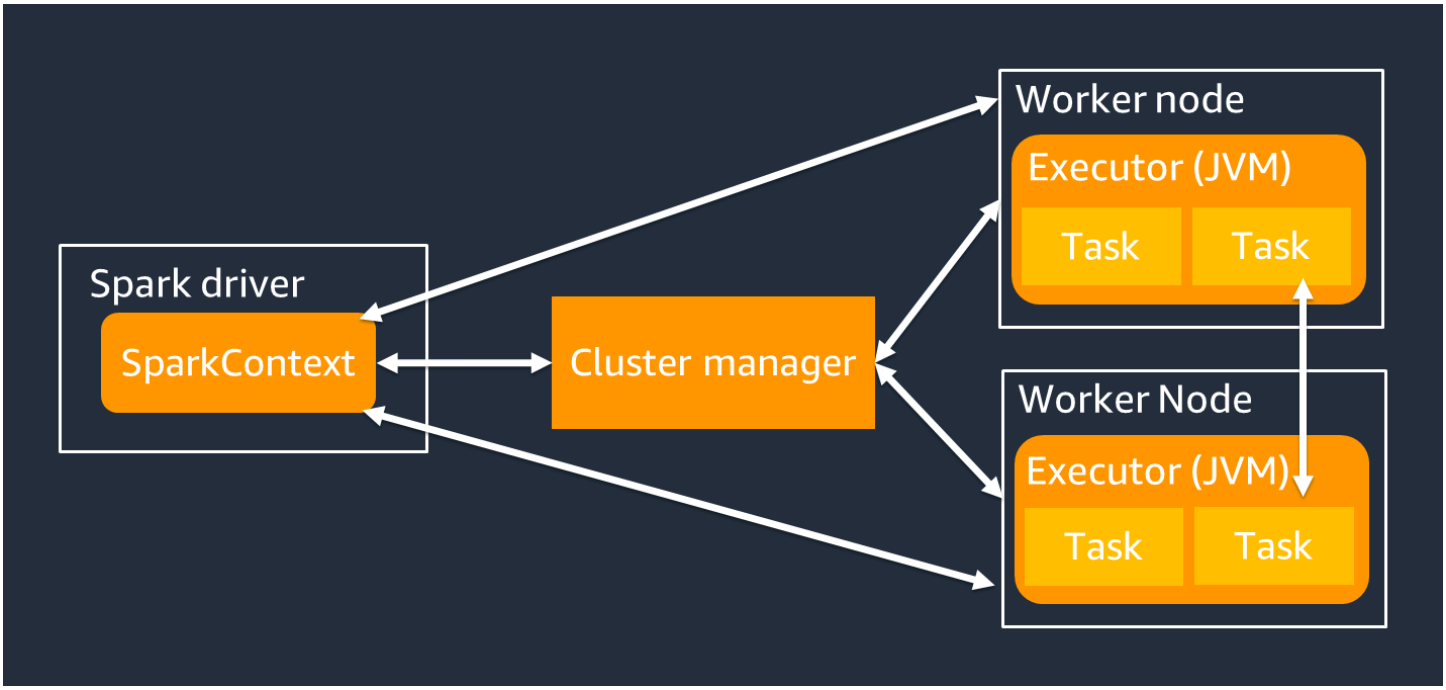
## Architektur

Der Spark-Treiber ist hauptsächlich dafür verantwortlich, Ihre Spark-Anwendung in Aufgaben aufzuteilen, die von einzelnen Workern ausgeführt werden können. Der Spark-Treiber hat die folgenden Aufgaben:

- Wird `main()` in Ihrem Code ausgeführt
- Generierung von Ausführungsplänen
- Bereitstellung von Spark-Executoren in Verbindung mit dem Clustermanager, der die Ressourcen auf dem Cluster verwaltet
- Planung von Aufgaben und Anforderung von Aufgaben für Spark-Executoren
- Verwaltung des Aufgabenfortschritts und der Wiederherstellung

Sie verwenden ein `SparkContext` Objekt, um mit dem Spark-Treiber für Ihre Jobausführung zu interagieren.

Ein Spark-Executor ist ein Worker, der Daten speichert und Aufgaben ausführt, die vom Spark-Treiber übergeben werden. Die Anzahl der Spark-Executoren steigt und fällt mit der Größe Ihres Clusters.

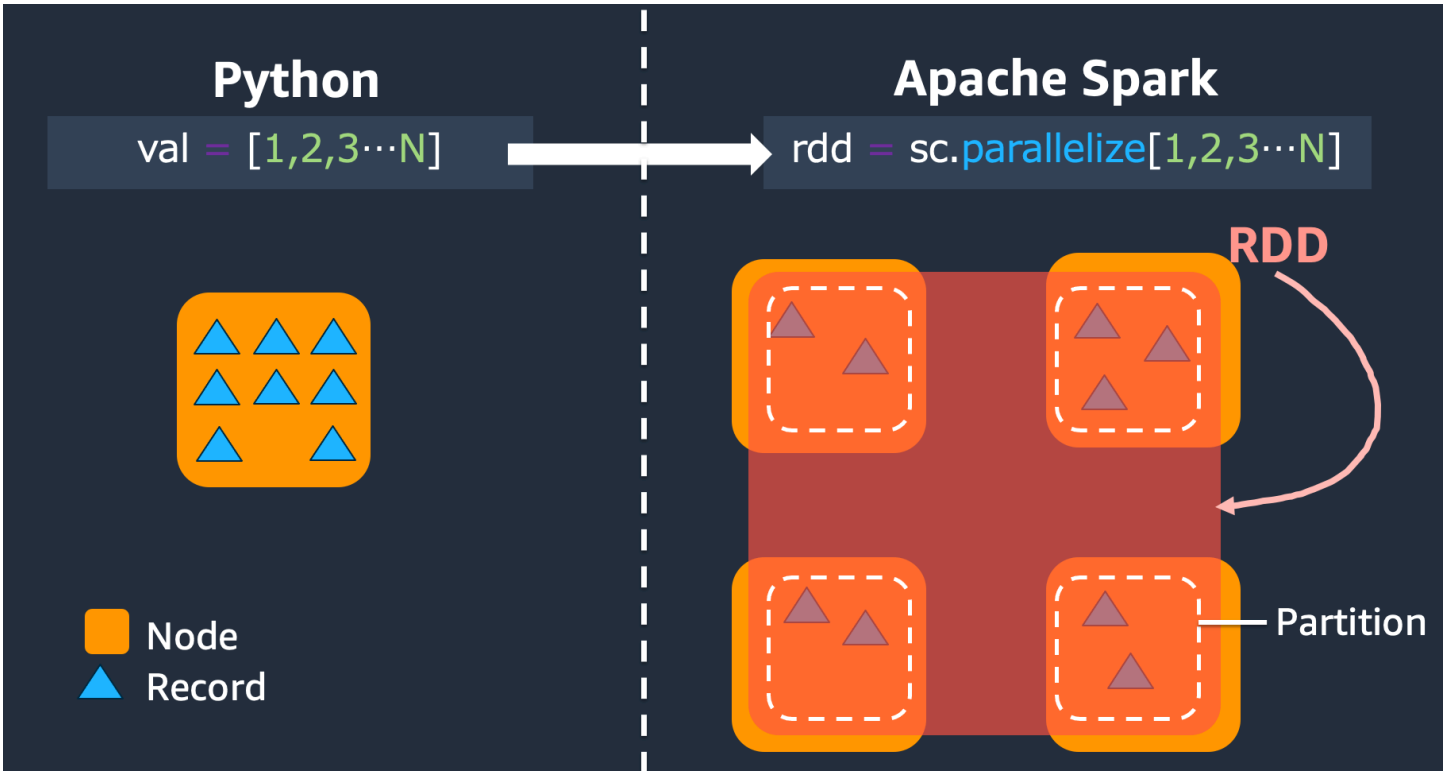



> **Note**
>
> Ein Spark-Executor hat mehrere Steckplätze, sodass mehrere Aufgaben parallel verarbeitet werden können. Spark unterstützt standardmäßig eine Aufgabe für jeden virtuellen CPU-Kern (vCPU). Wenn ein Executor beispielsweise über vier CPU-Kerne verfügt, kann er vier Aufgaben gleichzeitig ausführen.

## Belastbarer verteilter Datensatz

Spark erledigt die komplexe Aufgabe, große Datenmengen über Spark-Executoren hinweg zu speichern und zu verfolgen. Wenn Sie Code für Spark-Jobs schreiben, müssen Sie sich keine Gedanken über die Speicherdetails machen. Spark bietet die Resilient Distributed Dataset (RDD) - Abstraktion, bei der es sich um eine Sammlung von Elementen handelt, die parallel bearbeitet und auf die Spark-Executoren des Clusters aufgeteilt werden können.

Die folgende Abbildung zeigt den Unterschied darin, wie Daten im Speicher gespeichert werden, wenn ein Python-Skript in seiner typischen Umgebung ausgeführt wird und wenn es im Spark-Framework (PySpark) ausgeführt wird.




- Python — Beim Schreiben `val = [1,2,3...N]` in ein Python-Skript bleiben die Daten im Speicher des einzelnen Computers, auf dem der Code ausgeführt wird.
- PySpark— Spark bietet die RDD-Datenstruktur zum Laden und Verarbeiten von Daten, die über den Speicher verteilt sind, auf mehreren Spark-Executoren. Sie können ein RDD mit Code wie dem generieren`rdd = sc.parallelize[1,2,3...N]`, und Spark kann Daten automatisch auf mehrere Spark-Executoren verteilen und im Speicher speichern.

  In vielen AWS Glue Jobs verwenden Sie RDDs über und Spark. AWS Glue DynamicFramesDataFrames Dies sind Abstraktionen, mit denen Sie das Schema der Daten in einem RDD definieren und mit diesen zusätzlichen Informationen Aufgaben auf höherer Ebene ausführen können. Da sie RDDs intern verwenden, werden Daten im folgenden Code transparent verteilt und auf mehrere Knoten geladen:

  - DynamicFrame

```
dyf= glueContext.create_dynamic_frame.from_options(
    's3', {"paths": [ "s3://<YourBucket>/<Prefix>/"]},
    format="parquet",
    transformation_ctx="dyf"
)
```

- DataFrame

```
df = spark.read.format("parquet")
    .load("s3://<YourBucket>/<Prefix>")
```

Ein RDD hat die folgenden Funktionen:

- RDDs bestehen aus Daten, die in mehrere Teile unterteilt sind, die als Partitionen bezeichnet werden. Jeder Spark-Executor speichert eine oder mehrere Partitionen im Speicher, und die Daten werden auf mehrere Executoren verteilt.
- RDDs sind unveränderlich, was bedeutet, dass sie nach ihrer Erstellung nicht mehr geändert werden können. Um a zu ändern DataFrame, können Sie Transformationen verwenden, die im folgenden Abschnitt definiert werden.
- RDDs replizieren Daten auf allen verfügbaren Knoten, sodass sie nach einem Ausfall eines Knotens automatisch wiederhergestellt werden können.

## Faule Bewertung

RDDs unterstützen zwei Arten von Operationen: Transformationen, bei denen aus einem vorhandenen Datensatz ein neuer Datensatz erstellt wird, und Aktionen, bei denen ein Wert an das Treiberprogramm zurückgegeben wird, nachdem eine Berechnung für den Datensatz ausgeführt wurde.

- Transformationen — Da RDDs unveränderlich sind, können Sie sie nur mithilfe einer Transformation ändern.

  `map`Ist beispielsweise eine Transformation, bei der jedes Datensatzelement eine Funktion durchläuft und eine neue RDD zurückgibt, die die Ergebnisse darstellt. Beachten Sie, dass die `map` Methode keine Ausgabe zurückgibt. Spark speichert die abstrakte Transformation für die future, anstatt Sie mit dem Ergebnis interagieren zu lassen. Spark reagiert erst auf Transformationen, wenn Sie eine Aktion aufrufen.
- Aktionen — Mithilfe von Transformationen erstellen Sie Ihren logischen Transformationsplan. Um die Berechnung zu starten, führen Sie eine Aktion wie`write`, `countshow`, oder aus. `collect`

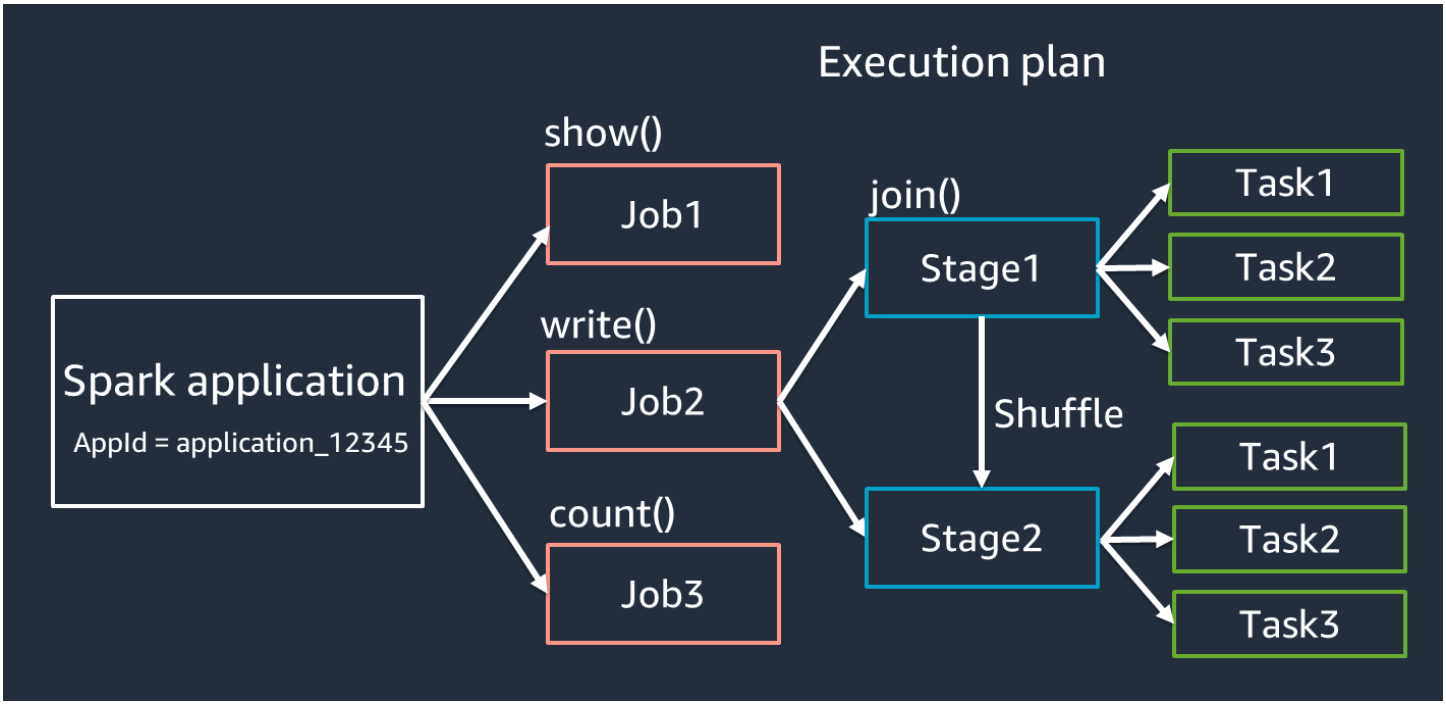
  Alle Transformationen in Spark sind faul, da sie ihre Ergebnisse nicht sofort berechnen. Stattdessen erinnert sich Spark an eine Reihe von Transformationen, die auf einige

Basisdatensätze angewendet wurden, z. B. auf Amazon Simple Storage Service (Amazon S3) -Objekte. Die Transformationen werden nur berechnet, wenn für eine Aktion ein Ergebnis an den Treiber zurückgegeben werden muss. Dieses Design ermöglicht es Spark, effizienter zu arbeiten. Stellen Sie sich beispielsweise die Situation vor, in der ein durch die `map` Transformation erstellter Datensatz nur für eine Transformation verwendet wird, die die Anzahl der Zeilen erheblich reduziert, wie `reduce` z. Sie können dann den kleineren Datensatz, der beide Transformationen durchlaufen hat, an den Treiber übergeben, anstatt den größeren zugewiesenen Datensatz zu übergeben.

# Terminologie der Spark-Anwendungen

In diesem Abschnitt wird die Terminologie von Spark-Anwendungen behandelt. Der Spark-Treiber erstellt einen Ausführungsplan und steuert das Verhalten von Anwendungen in verschiedenen Abstraktionen. Die folgenden Begriffe sind wichtig für die Entwicklung, das Debugging und die Leistungsoptimierung mit der Spark-Benutzeroberfläche.

- Anwendung — Basiert auf einer Spark-Sitzung (Spark-Kontext). Identifiziert durch eine eindeutige ID wie`<application_XXX>`.
- Jobs — Basierend auf den Aktionen, die für ein RDD erstellt wurden. Ein Job besteht aus einer oder mehreren Phasen.
- Stufen — Basierend auf den für ein RDD erstellten Shuffles. Eine Phase besteht aus einer oder mehreren Aufgaben. Shuffle ist der Mechanismus von Spark zur Umverteilung von Daten, sodass sie auf RDD-Partitionen unterschiedlich gruppiert werden. Bestimmte Transformationen, wie z. B.`join()`, erfordern einen Shuffle. Die Zufallswiedergabe wird in der Übung zur Optimierung der Zufallswiedergabe optimieren ausführlicher behandelt.
- Aufgaben — Eine Aufgabe ist die von Spark geplante Mindestverarbeitungseinheit. Aufgaben werden für jede RDD-Partition erstellt, und die Anzahl der Aufgaben entspricht der maximalen Anzahl gleichzeitiger Ausführungen in der Phase.

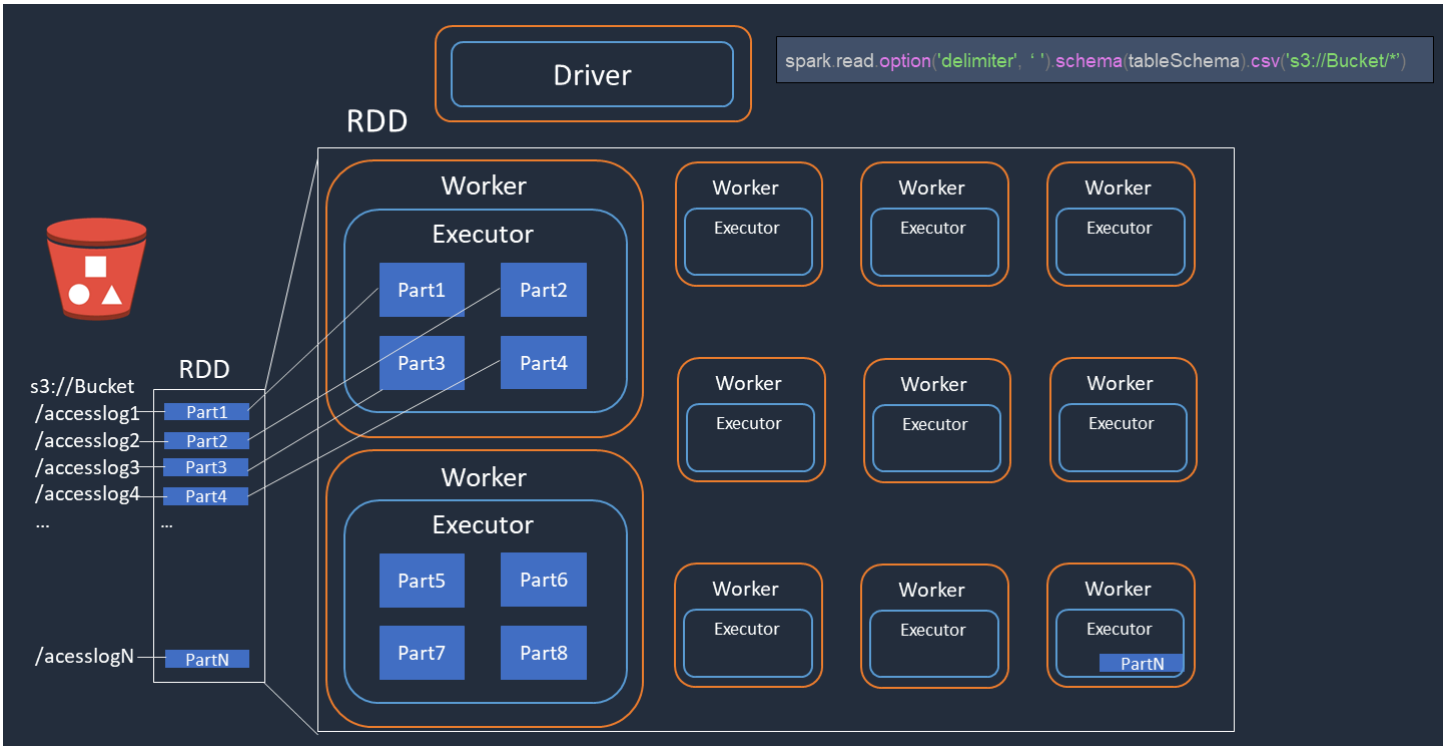



> **Note**
>
> Aufgaben sind das Wichtigste, was bei der Optimierung der Parallelität berücksichtigt werden muss. Die Anzahl der Aufgaben skaliert mit der Anzahl der RDD

## Parallelism

Spark parallelisiert Aufgaben zum Laden und Transformieren von Daten.

Stellen Sie sich ein Beispiel vor, in dem Sie eine verteilte Verarbeitung von Zugriffsprotokolldateien (benannt`accesslog1 ... accesslogN`) auf Amazon S3 durchführen. Das folgende Diagramm zeigt den Ablauf der verteilten Verarbeitung.

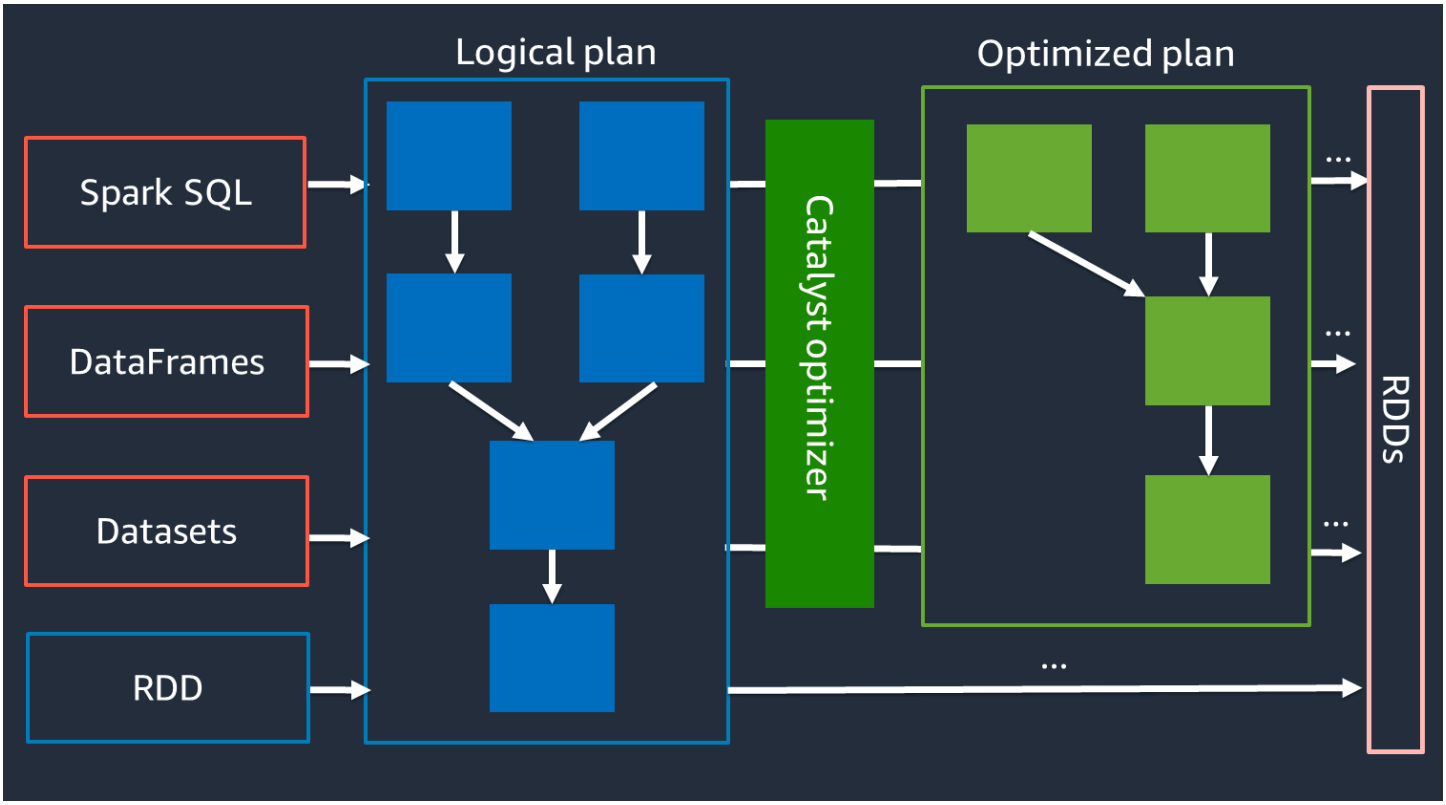



1. Der Spark-Treiber erstellt einen Ausführungsplan für die verteilte Verarbeitung auf viele Spark-Executoren.
2. Der Spark-Treiber weist jedem Executor auf der Grundlage des Ausführungsplans Aufgaben zu. Standardmäßig erstellt der Spark-Treiber RDD-Partitionen (jede entspricht einer Spark-Aufgabe) für jedes S3-Objekt (). `Part1 ... N` Anschließend weist der Spark-Treiber jedem Executor Aufgaben zu.
3. Jede Spark-Aufgabe lädt das zugewiesene S3-Objekt herunter und speichert es im Speicher der RDD-Partition. Auf diese Weise laden mehrere Spark-Executoren ihre zugewiesene Aufgabe herunter und verarbeiten sie parallel.

Weitere Informationen zur anfänglichen Anzahl von Partitionen und zur Optimierung finden Sie im Abschnitt Aufgaben parallelisieren.

## Catalyst-Optimierer

Intern verwendet Spark eine Engine namens Catalyst Optimizer, um Ausführungspläne zu optimieren. Catalyst verfügt über einen Abfrageoptimierer, den Sie verwenden können, wenn Sie Spark-APIs auf hoher Ebene wie Spark SQL und Datasets ausführen, wie im folgenden Diagramm beschrieben. DataFrame




Da der Catalyst-Optimierer nicht direkt mit der RDD-API zusammenarbeitet, sind die High-Level-APIs im Allgemeinen schneller als die Low-Level-RDD-API. Bei komplexen Verknüpfungen kann der Catalyst-Optimierer die Leistung erheblich verbessern, indem er den Job-Ausführungsplan optimiert. Sie können den optimierten Plan Ihres Spark-Jobs auf der Registerkarte SQL der Spark-Benutzeroberfläche sehen.

Adaptive Abfrageausführung

Der Catalyst-Optimierer führt die Laufzeitoptimierung mithilfe eines Prozesses namens Adaptive Query Execution durch. Adaptive Query Execution verwendet Laufzeitstatistiken, um den Ausführungsplan der Abfragen erneut zu optimieren, während Ihr Job ausgeführt wird. Adaptive Query Execution bietet mehrere Lösungen für Performance-Probleme, darunter das Zusammenführen von Partitionen nach dem Zufälligen Zusammenführen, das Konvertieren von Sort-Merge-Join in Broadcast-Join und die Optimierung von Skew-Joins, wie in den folgenden Abschnitten beschrieben.

Adaptive Query Execution ist in AWS Glue 3.0 und höher verfügbar und in AWS Glue 4.0 (Spark 3.3.0) und höher standardmäßig aktiviert. Adaptive Query Execution kann `spark.conf.set("spark.sql.adaptive.enabled", "true")` in Ihrem Code ein- und ausgeschaltet werden.

Zusammenführung von Partitionen nach dem Shuffle-Modus

Diese Funktion reduziert die Anzahl der RDD-Partitionen (Zusammenführung) nach jedem Shuffle auf der Grundlage der Ausgabestatistiken. `map` Es vereinfacht die Einstellung der Shuffle-Partitionsnummer bei der Ausführung von Abfragen. Sie müssen keine Shuffle-Partitionsnummer festlegen, die zu Ihrem Datensatz passt. Spark kann zur Laufzeit die richtige Shuffle-Partitionsnummer auswählen, wenn Sie eine ausreichend große anfängliche Anzahl von Shuffle-Partitionen haben.

Das Zusammenführen von Post-Shuffle-Partitionen ist aktiviert, wenn sowohl als auch `spark.sql.adaptive.enabled` auf true gesetzt sind. `spark.sql.adaptive.coalescePartitions.enabled` Weitere Informationen finden Sie in der Apache Spark-Dokumentation.

Sort-Merge-Join in Broadcast-Join konvertieren

Diese Funktion erkennt, wenn Sie zwei Datensätze mit wesentlich unterschiedlicher Größe verbinden, und verwendet auf der Grundlage dieser Informationen einen effizienteren Verbindungsalgorithmus. Weitere Informationen finden Sie in der Apache Spark-Dokumentation. Strategien zum Zusammenfügen werden im Abschnitt Optimize shuffles besprochen.

Optimierung von Skew-Joins

Datenverzerrung ist einer der häufigsten Engpässe bei Spark-Jobs. Es beschreibt eine Situation, in der Daten auf bestimmte RDD-Partitionen (und folglich auf bestimmte Aufgaben) verschoben werden, was die Gesamtverarbeitungszeit der Anwendung verzögert. Dadurch kann die Leistung von Verbindungsvorgängen häufig beeinträchtigt werden. Mit der Funktion zur Optimierung von Schrägverknüpfungen werden Verzerrungen bei Sort-Merge-Verknüpfungen dynamisch behandelt, indem schiefe Aufgaben in etwa gleich große Aufgaben aufgeteilt (und bei Bedarf repliziert) werden.

Diese Funktion ist aktiviert, wenn sie auf „true“ gesetzt ist. `spark.sql.adaptive.skewJoin.enabled` Weitere Informationen finden Sie in der Apache Spark-Dokumentation. Datenverzerrungen werden im Abschnitt „Shuffles optimieren“ näher erläutert.

# Untersuchen Sie Leistungsprobleme mithilfe der Spark-Benutzeroberfläche

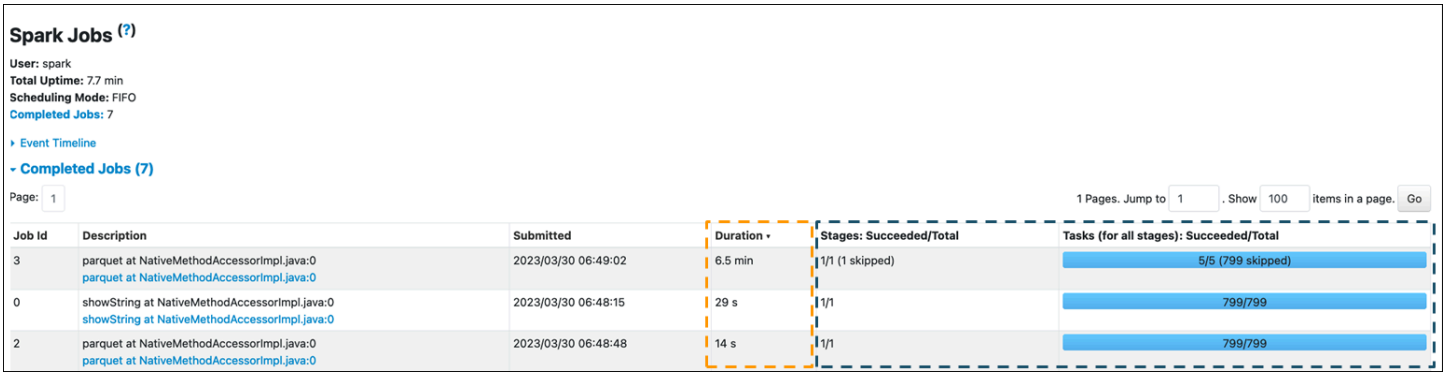
Bevor Sie bewährte Methoden anwenden, um die Leistung Ihrer AWS Glue Jobs zu optimieren, empfehlen wir Ihnen dringend, ein Leistungsprofil zu erstellen und die Engpässe zu identifizieren. Dies wird Ihnen helfen, sich auf die richtigen Dinge zu konzentrieren.

Für eine schnelle Analyse bieten die CloudWatch Amazon-Statistiken einen grundlegenden Überblick über Ihre Jobkennzahlen. Die Spark-Benutzeroberfläche bietet einen tieferen Einblick in die Leistungsoptimierung. Um die Spark-Benutzeroberfläche mit verwenden zu können AWS Glue, müssen Sie die Spark-Benutzeroberfläche für Ihre AWS Glue Jobs aktivieren. Nachdem Sie sich mit der Spark-Benutzeroberfläche vertraut gemacht haben, folgen Sie den Strategien zur Optimierung der Spark-Auftragsleistung, um die Auswirkungen von Engpässen auf der Grundlage Ihrer Ergebnisse zu identifizieren und zu reduzieren.

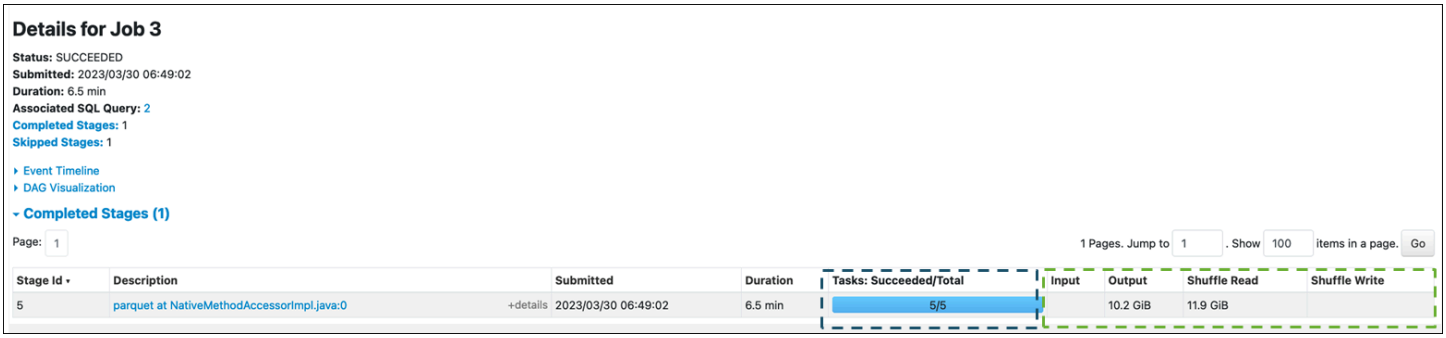
## Identifizieren Sie Engpässe mithilfe der Spark-Benutzeroberfläche

Wenn Sie die Spark-Benutzeroberfläche öffnen, werden Spark-Anwendungen in einer Tabelle aufgeführt. Standardmäßig lautet der App-Name eines AWS Glue Jobs`nativespark-<Job Name>-<Job Run ID>`. Wählen Sie die Spark-Ziel-App basierend auf der Job-Ausführungs-ID aus, um die Registerkarte Jobs zu öffnen. Unvollständige Auftragsausführungen, wie z. B. Streaming-Jobläufe, sind unter Unvollständige Anwendungen anzeigen aufgeführt.

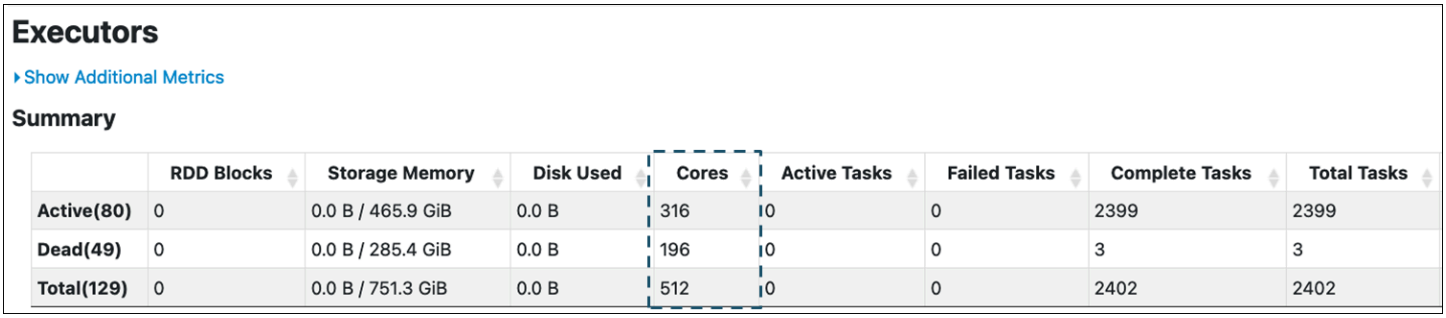
Auf der Registerkarte Jobs wird eine Zusammenfassung aller Jobs in der Spark-Anwendung angezeigt. Um festzustellen, ob Phasen- oder Aufgabenfehler aufgetreten sind, überprüfen Sie die Gesamtzahl der Aufgaben. Um die Engpässe zu finden, sortieren Sie, indem Sie Dauer wählen. Rufen Sie die Details von Jobs mit langer Laufzeit auf, indem Sie auf den Link in der Spalte Beschreibung klicken.




Auf der Seite „Details für den Job“ werden die Phasen aufgeführt. Auf dieser Seite finden Sie allgemeine Informationen wie die Dauer, die Anzahl der erfolgreich abgeschlossenen Aufgaben und die Gesamtzahl der Aufgaben, die Anzahl der Ein- und Ausgaben sowie die Anzahl der Lesevorgänge und der Zufallsschreibvorgänge.




Auf der Registerkarte Executor wird die Spark-Cluster-Kapazität im Detail angezeigt. Sie können die Gesamtzahl der Kerne überprüfen. Der im folgenden Screenshot gezeigte Cluster enthält 316 aktive Kerne und insgesamt 512 Kerne. Standardmäßig kann jeder Kern eine Spark-Aufgabe gleichzeitig verarbeiten.




Basierend auf dem Wert, 5/5 der auf der Seite „Auftragsdetails“ angezeigt wird, ist Phase 5 die längste Phase, sie verwendet jedoch nur 5 von 512 Kernen. Da die Parallelität für diese Phase so gering ist, sie aber viel Zeit in Anspruch nimmt, können Sie sie als Engpass identifizieren. Um die

Leistung zu verbessern, sollten Sie verstehen, warum. Weitere Informationen darüber, wie Sie häufig auftretende Leistungsengpässe erkennen und deren Auswirkungen reduzieren können, finden Sie unter Strategien zur Optimierung der Spark-Jobleistung.

# Strategien zur Optimierung der Spark-Jobleistung

Wenden Sie die folgenden bewährten Methoden an, wenn Sie das Optimieren von Parametern vorbereiten:

- Legen Sie Ihre Leistungsziele fest, bevor Sie beginnen, Probleme zu identifizieren.
- Identifizieren Sie Probleme anhand von Metriken, bevor Sie versuchen, die Optimierungsparameter zu ändern.

Damit Sie beim Optimieren eines Auftrags möglichst konsistente Ergebnisse erzielen, sollten Sie eine Basisstrategie für Ihre Optimierungsarbeit entwickeln.

## Basisstrategie für die Leistungsoptimierung

Im Allgemeinen wird die Leistungsoptimierung im folgenden Workflow durchgeführt:

1. Leistungsziele ermitteln.
2. Metriken messen.
3. Engpässe identifizieren.
4. Die Auswirkung von Engpässen verringern.
5. Wiederholen Sie die Schritte 2 bis 4, bis Sie das beabsichtigte Ziel erreicht haben.

Legen Sie zunächst Ihre Leistungsziele fest. Eines Ihrer Ziele könnte beispielsweise darin bestehen, die Ausführung eines AWS Glue Jobs innerhalb von 3 Stunden abzuschließen. Nachdem Sie Ihre Ziele definiert haben, messen Sie die Arbeitsleistungskennzahlen. Identifizieren Sie Trends bei den Kennzahlen und Engpässen, um die Ziele zu erreichen. Insbesondere die Identifizierung von Engpässen ist für die Fehlerbehebung, das Debugging und die Leistungsoptimierung von größter Bedeutung. Während der Ausführung einer Spark-Anwendung zeichnet Spark den Status und die Statistiken jeder Aufgabe im Spark-Ereignisprotokoll auf.

In AWS Glue können Sie Spark-Metriken über die Spark-Weboberfläche anzeigen, die vom Spark-History-Server bereitgestellt wird. AWS Glue für Spark-Jobs kann Spark-Ereignisprotokolle an einen Speicherort senden, den Sie in Amazon S3 angeben. AWS Glue bietet auch eine AWS CloudFormation Beispielvorlage und ein Dockerfile, um den Spark-History-Server auf einer EC2

Amazon-Instance oder Ihrem lokalen Computer zu starten, sodass Sie die Spark-Benutzeroberfläche mit Ereignisprotokollen verwenden können.

Nachdem Sie Ihre Leistungsziele festgelegt und Kennzahlen zur Bewertung dieser Ziele identifiziert haben, können Sie mit der Identifizierung und Behebung von Engpässen beginnen, indem Sie die Strategien in den folgenden Abschnitten verwenden.

## Optimierungspraktiken für die Arbeitsleistung von Spark

Sie können die folgenden Strategien zur Leistungsoptimierung AWS Glue für Spark-Jobs verwenden:

- AWS Glue Ressourcen:
  - Skalieren Sie die Clusterkapazität
  - Verwenden Sie die neueste Version AWS Glue
- Spark-Anwendungen:
  - Reduzieren Sie den Umfang der gescannten Daten
  - Aufgaben parallelisieren
  - Optimieren Sie Shuffles
  - Minimieren Sie den Planungsaufwand
  - Optimieren Sie benutzerdefinierte Funktionen

Bevor Sie diese Strategien verwenden können, müssen Sie Zugriff auf Metriken und Konfigurationen für Ihren Spark-Job haben. Sie finden diese Informationen in der AWS Glue Dokumentation.

Aus Sicht der AWS Glue Ressourcen können Sie Leistungsverbesserungen erzielen, indem Sie AWS Glue Mitarbeiter hinzufügen und die neueste AWS Glue Version verwenden.

Aus Sicht der Apache Spark-Anwendung haben Sie Zugriff auf verschiedene Strategien, mit denen Sie die Leistung verbessern können. Wenn nicht benötigte Daten in den Spark-Cluster geladen werden, können Sie sie entfernen, um die Menge der geladenen Daten zu reduzieren. Wenn Sie die Spark-Cluster-Ressourcen nicht ausreichend genutzt haben und nur über geringe Daten-I/O verfügen, können Sie Aufgaben identifizieren, die parallelisiert werden sollen. Möglicherweise möchten Sie auch umfangreiche Datenübertragungsvorgänge wie Verknüpfungen optimieren, wenn sie viel Zeit in Anspruch nehmen. Sie können auch Ihren Plan für die Jobabfrage optimieren oder die Rechenkomplexität einzelner Spark-Aufgaben reduzieren.

Um diese Strategien effizient anwenden zu können, müssen Sie anhand Ihrer Kennzahlen ermitteln, wann sie anwendbar sind. Weitere Informationen finden Sie in den folgenden Abschnitten. Diese Techniken eignen sich nicht nur zur Leistungsoptimierung, sondern auch zur Lösung typischer Probleme wie out-of-memory (OOM) -Fehler.

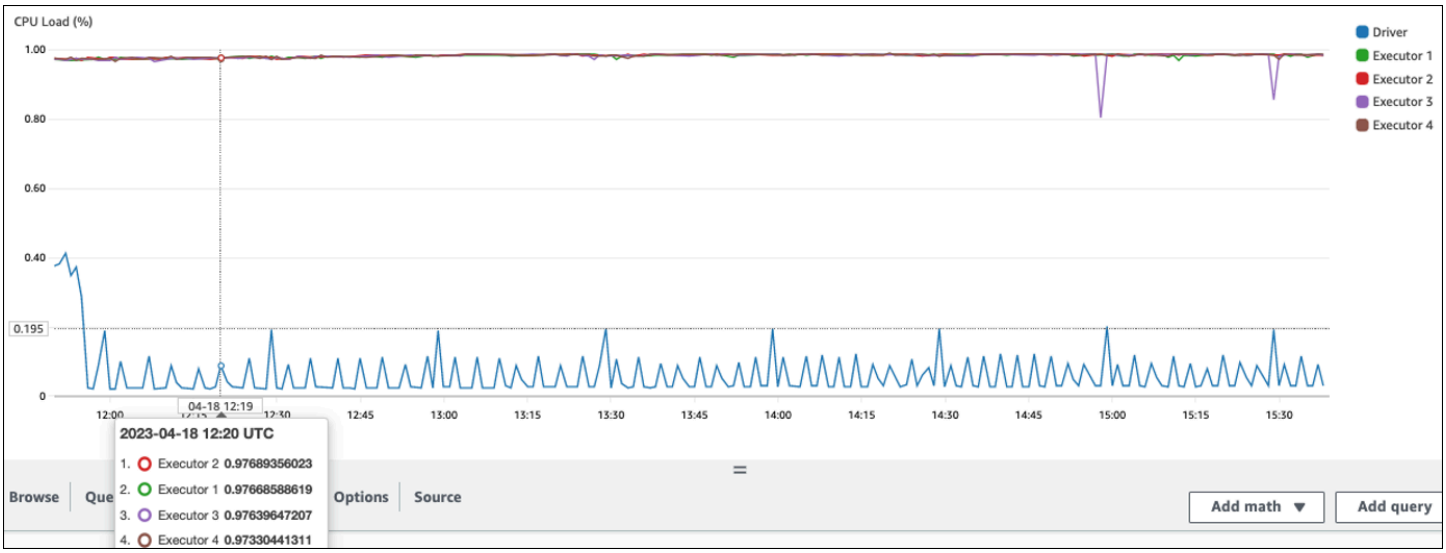
# Skalieren Sie die Clusterkapazität

Wenn Ihr Job zu viel Zeit in Anspruch nimmt, die Executoren jedoch ausreichend Ressourcen verbrauchen und Spark im Verhältnis zu den verfügbaren Kernen eine große Menge an Aufgaben erstellt, sollten Sie eine Skalierung der Clusterkapazität in Betracht ziehen. Verwenden Sie die folgenden Kennzahlen, um zu beurteilen, ob dies angemessen ist.

## CloudWatch Metriken

- Überprüfen Sie CPULast und Speicherauslastung, um festzustellen, ob die Executoren ausreichend Ressourcen verbrauchen.
- Prüfen Sie, wie lange der Job ausgeführt wurde, um festzustellen, ob die Verarbeitungszeit zu lang ist, um Ihre Leistungsziele zu erreichen.

Im folgenden Beispiel werden vier Executoren mit einer CPU Auslastung von mehr als 97 Prozent ausgeführt, aber die Verarbeitung ist nach etwa drei Stunden noch nicht abgeschlossen.

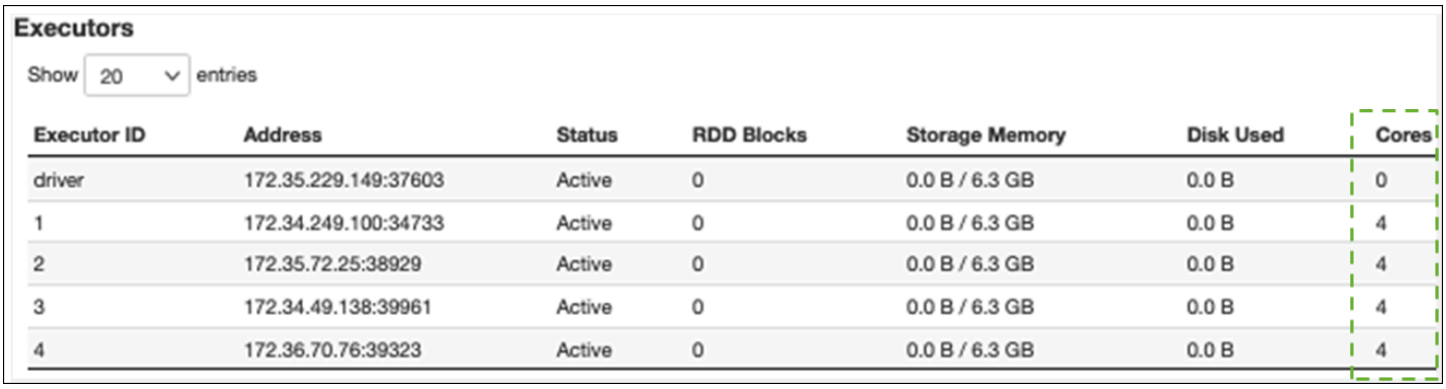


> **Note**
>
> Wenn die CPU Auslastung gering ist, werden Sie wahrscheinlich nicht von der Skalierung der Clusterkapazität profitieren.

## Spark-Benutzeroberfläche

Auf der Registerkarte Job oder der Registerkarte Phase können Sie die Anzahl der Aufgaben für jeden Job oder jede Phase sehen. Im folgenden Beispiel hat Spark `58100` Aufgaben erstellt.


Auf der Registerkarte Executor können Sie die Gesamtzahl der Executoren und Aufgaben sehen. Im folgenden Screenshot hat jeder Spark-Executor vier Kerne und kann vier Aufgaben gleichzeitig ausführen.




In diesem Beispiel `58100`) ist die Anzahl der Spark-Aufgaben () viel größer als die 16 Aufgaben, die die Executoren gleichzeitig verarbeiten können (4 Executors × 4 Cores).

Wenn Sie diese Symptome beobachten, sollten Sie eine Skalierung des Clusters in Betracht ziehen. Sie können die Clusterkapazität mithilfe der folgenden Optionen skalieren:

- AWS Glue Auto Scaling aktivieren — Auto Scaling ist für Ihre AWS Glue Extraktions-, Transformations- und Load (ETL) - und Streaming-Jobs in AWS Glue Version 3.0 oder höher verfügbar. AWS Glue fügt dem Cluster automatisch Worker hinzu oder entfernt sie aus dem

Cluster, abhängig von der Anzahl der Partitionen in jeder Phase oder der Geschwindigkeit, mit der Mikrobatches bei der Jobausführung generiert werden.

Wenn Sie eine Situation beobachten, in der die Anzahl der Mitarbeiter nicht zunimmt, obwohl Auto Scaling aktiviert ist, sollten Sie erwägen, Mitarbeiter manuell hinzuzufügen. Beachten Sie jedoch, dass die manuelle Skalierung für eine Phase dazu führen kann, dass viele Mitarbeiter in späteren Phasen untätig sind, was mehr kostet, ohne dass die Leistung gesteigert wird.

Nachdem Sie Auto Scaling aktiviert haben, können Sie die Anzahl der Executors in den Executor-Metriken CloudWatch sehen. Verwenden Sie die folgenden Metriken, um die Nachfrage nach Executoren in Spark-Anwendungen zu überwachen:

- `glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`
- `glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`

Weitere Informationen zu Metriken finden Sie unter Überwachung AWS Glue mithilfe von CloudWatch Amazon-Metriken.

- Skalieren: Erhöhen Sie die Anzahl der AWS Glue Mitarbeiter — Sie können die Anzahl der AWS Glue Mitarbeiter manuell erhöhen. Fügen Sie nur so lange Arbeitskräfte hinzu, bis Sie untätige Mitarbeiter beobachten. Zu diesem Zeitpunkt erhöht das Hinzufügen weiterer Mitarbeiter die Kosten, ohne dass sich die Ergebnisse verbessern. Weitere Informationen finden Sie unter Aufgaben parallelisieren.
- Skalieren: Verwenden Sie einen größeren Worker-Typ — Sie können den Instance-Typ Ihrer Worker manuell ändern, um AWS Glue Worker mit mehr Kernen, Arbeitsspeicher und Speicher zu verwenden. Größere Workertypen ermöglichen Ihnen die vertikale Skalierung und Ausführung intensiver Datenintegrationsaufgaben, wie z. B. speicherintensive Datentransformationen, schiefe Aggregationen und Entitätserkennungsprüfungen mit Petabytes an Daten.

  Die Skalierung hilft auch in Fällen, in denen der Spark-Treiber mehr Kapazität benötigt, z. B. weil der Job-Abfrageplan recht umfangreich ist. Weitere Informationen zu Arbeitstypen und Leistung finden Sie im AWS Big-Data-Blogbeitrag Scale Your AWS Glue für Apache Spark-Jobs mit den neuen größeren Workertypen G.4X und G.8X.

  Durch den Einsatz größerer Arbeitskräfte kann auch die Gesamtzahl der benötigten Arbeitskräfte reduziert werden, wodurch die Leistung gesteigert wird, da bei intensiven Vorgängen wie dem Zusammenführen weniger häufig gemischt wird.

# Verwenden Sie die neueste Version AWS Glue

Wir empfehlen, die neueste AWS Glue Version zu verwenden. In jeder Version sind mehrere Optimierungen und Upgrades integriert, die die Arbeitsleistung automatisch verbessern können. AWS Glue 4.0 bietet beispielsweise die folgenden neuen Funktionen:

- Neue optimierte Apache Spark 3.3.0-Laufzeit — AWS Glue 4.0 baut auf der Apache Spark 3.3.0-Laufzeit auf und bietet vergleichbare Leistungsverbesserungen wie Open-Source-Spark. Die Spark 3.3.0-Laufzeit baut auf vielen der Innovationen von Spark 2.x auf.
- Verbesserter Amazon Redshift Redshift-Connector — AWS Glue 4.0 und höhere Versionen bieten Amazon Redshift Redshift-Integration für Apache Spark. Die Integration baut auf einem vorhandenen Open-Source-Konnektor auf und verbessert ihn in Bezug auf Leistung und Sicherheit. Durch die Integration können Anwendungen bis zu zehnmal schneller ausgeführt werden. Weitere Informationen finden Sie im Blogbeitrag zur Amazon Redshift Redshift-Integration mit Apache Spark.
- SIMDbasierte Ausführung für vektorisierte Lesevorgänge mit CSV und JSON Daten — AWS Glue Version 3.0 und spätere Versionen bieten optimierte Lesegeräte, die die allgemeine Arbeitsleistung im Vergleich zu zeilenbasierten Lesegeräten erheblich beschleunigen können. Weitere Informationen zu CSV Daten finden Sie unter Optimieren der Leseleistung mit vektorisiertem Reader. SIMD CSV Weitere Informationen zu JSON Daten finden Sie unter Verwenden eines vektorisierten SIMD JSON Lesegeräts mit dem Apache Arrow-Spaltenformat.

Jede AWS Glue Version wird unter anderem Upgrades dieser Art enthalten, darunter Konnektoren, Treiber- und Bibliotheksupdates. Weitere Informationen finden Sie unter AWS Glue Versionen und AWS Glue Jobs auf AWS Glue Version 4.0 migrieren.

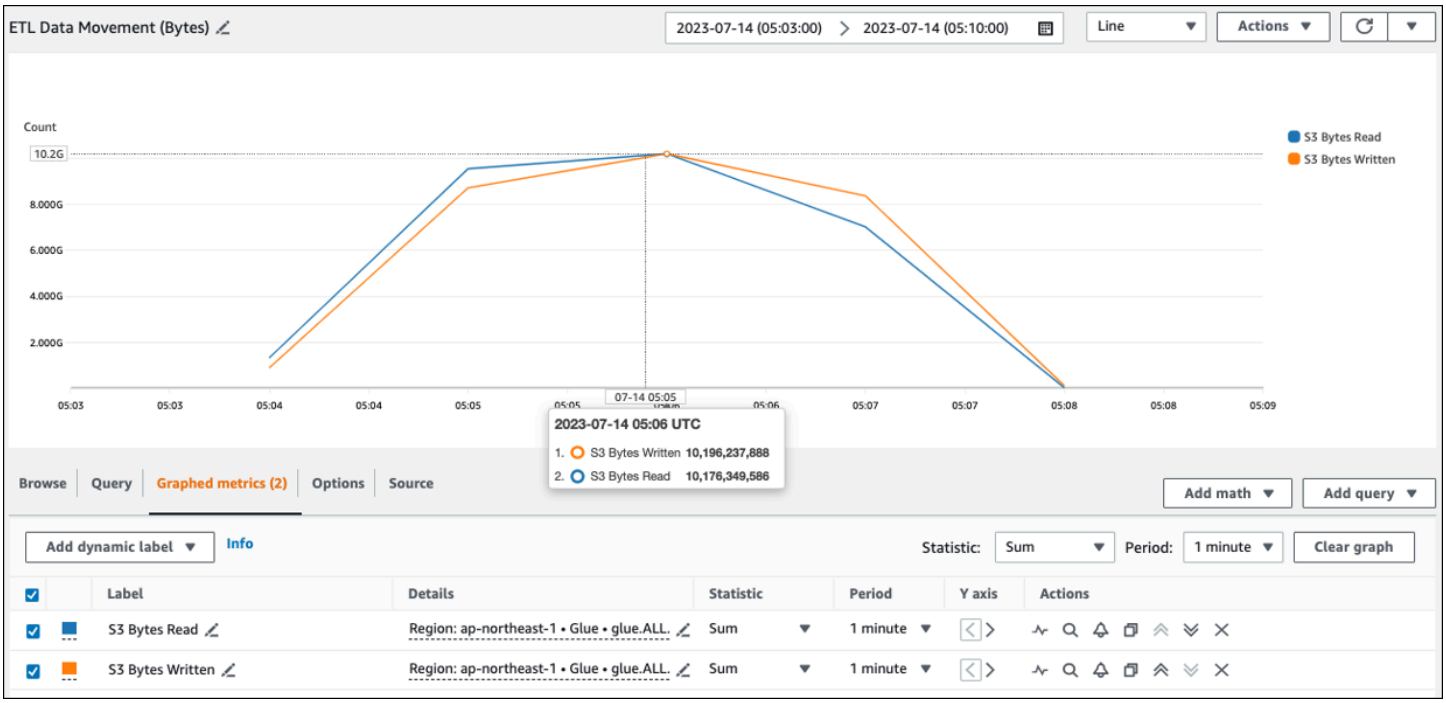
# Reduzieren Sie den Umfang der gescannten Daten

Erwägen Sie zunächst, nur die Daten zu laden, die Sie benötigen. Sie können die Leistung verbessern, indem Sie einfach die Datenmenge reduzieren, die für jede Datenquelle in Ihren Spark-Cluster geladen wird. Verwenden Sie die folgenden Kennzahlen, um zu beurteilen, ob dieser Ansatz angemessen ist.

Sie können die von Amazon S3 gelesenen Bytes in CloudWatchMetriken und weiteren Details in der Spark-Benutzeroberfläche überprüfen, wie im Abschnitt Spark-Benutzeroberfläche beschrieben.

## CloudWatch Metriken

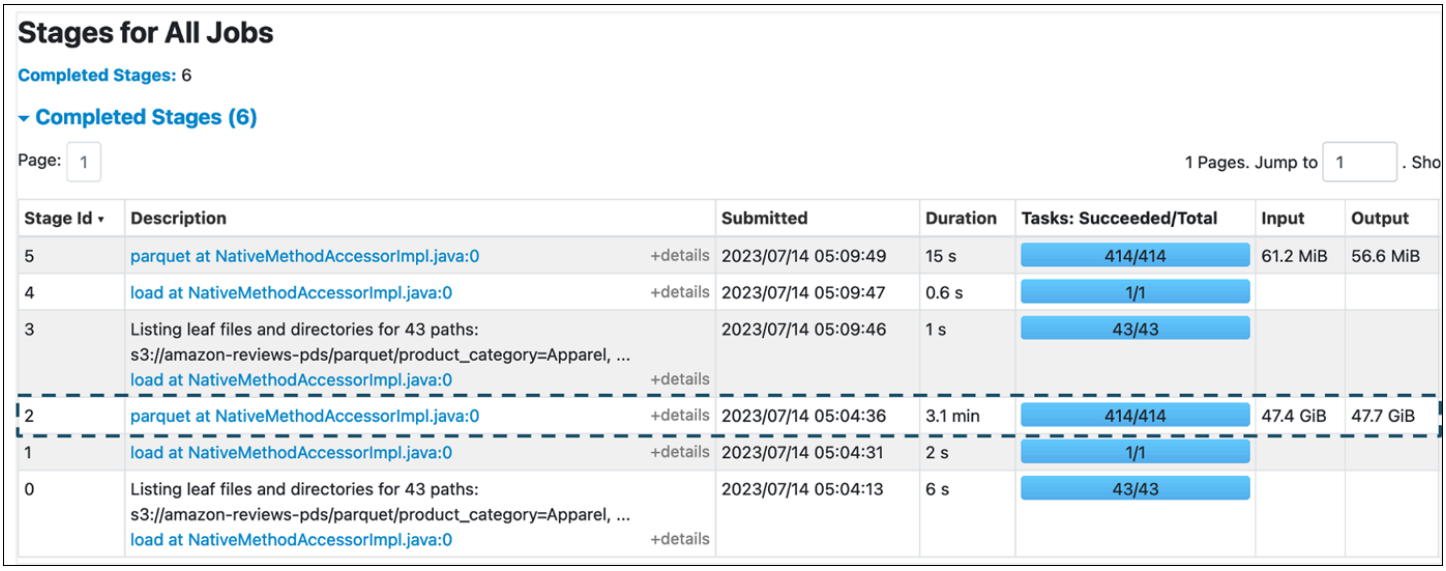
Sie können die ungefähre Lesegröße von Amazon S3 unter ETLDatenbewegung (Byte) sehen. Diese Metrik zeigt die Anzahl der Byte, die seit dem letzten Bericht von allen Ausführern aus Amazon S3 gelesen wurden. Sie können damit die ETL Datenbewegung aus Amazon S3 überwachen und Lesevorgänge mit Aufnahmeraten aus externen Datenquellen vergleichen.




Wenn Sie einen größeren S3-Byte-Lesedatenpunkt als erwartet beobachten, sollten Sie die folgenden Lösungen in Betracht ziehen.

## Spark-Benutzeroberfläche

Auf der Registerkarte Stage in der AWS Glue Benutzeroberfläche von Spark können Sie die Eingabe - und Ausgabegröße sehen. Im folgenden Beispiel liest Stufe 2 47,4 GiB Eingang und 47,7 GiB Ausgang, während Stufe 5 61,2 MiB Eingang und 56,6 MiB Ausgang liest.



**Stages for All Jobs**

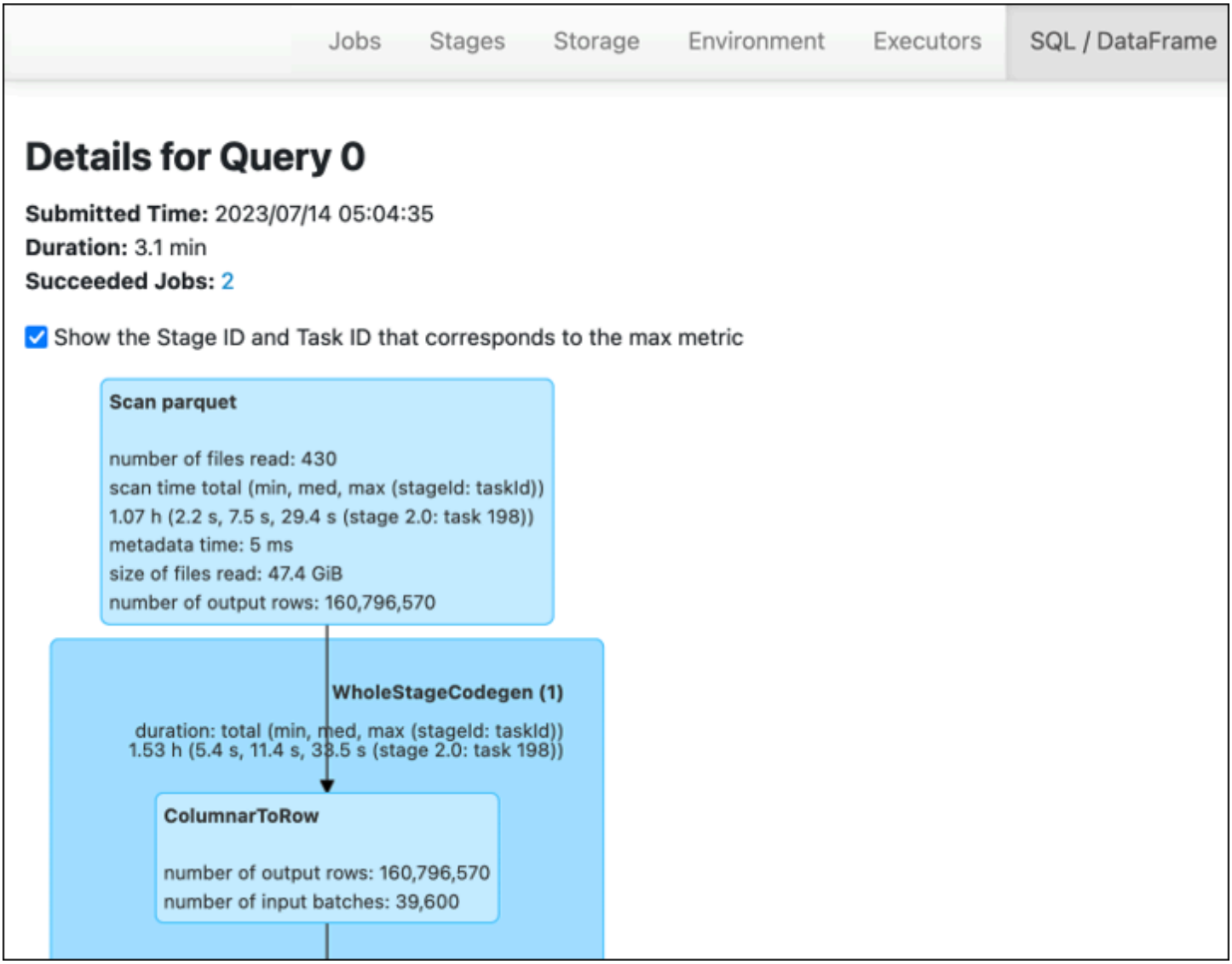
**Completed Stages:** 6

**Completed Stages (6)**

Page: 1 — 1 Pages. Jump to 1 . Sho

| Stage Id | Description | Submitted | Duration | Tasks: Succeeded/Total | Input | Output |
|---|---|---|---|---|---|---|
| 5 | parquet at NativeMethodAccessorImpl.java:0 +details | 2023/07/14 05:09:49 | 15 s | 414/414 | 61.2 MiB | 56.6 MiB |
| 4 | load at NativeMethodAccessorImpl.java:0 +details | 2023/07/14 05:09:47 | 0.6 s | 1/1 | | |
| 3 | Listing leaf files and directories for 43 paths: s3://amazon-reviews-pds/parquet/product_category=Apparel, ... load at NativeMethodAccessorImpl.java:0 +details | 2023/07/14 05:09:46 | 1 s | 43/43 | | |
| 2 | parquet at NativeMethodAccessorImpl.java:0 +details | 2023/07/14 05:04:36 | 3.1 min | 414/414 | 47.4 GiB | 47.7 GiB |
| 1 | load at NativeMethodAccessorImpl.java:0 +details | 2023/07/14 05:04:31 | 2 s | 1/1 | | |
| 0 | Listing leaf files and directories for 43 paths: s3://amazon-reviews-pds/parquet/product_category=Apparel, ... load at NativeMethodAccessorImpl.java:0 +details | 2023/07/14 05:04:13 | 6 s | 43/43 | | |

Wenn Sie Spark SQL oder DataFrame Approaches in Ihrem AWS Glue Job verwenden, werden auf der ataFrame Registerkarte SQL/D weitere Statistiken zu diesen Phasen angezeigt. In diesem Fall zeigt Stufe 2 die Anzahl der gelesenen Dateien: 430, die Größe der gelesenen Dateien: 47,4 GiB und die Anzahl der Ausgabezeilen: 160.796.570.




Wenn Sie feststellen, dass zwischen den Daten, die Sie einlesen, und den Daten, die Sie verwenden, ein erheblicher Größenunterschied besteht, probieren Sie die folgenden Lösungen aus.

## Amazon S3

Um die Datenmenge zu reduzieren, die beim Lesen aus Amazon S3 in Ihren Job geladen wird, sollten Sie Dateigröße, Komprimierung, Dateiformat und Dateilayout (Partitionen) für Ihren Datensatz berücksichtigen. AWS Glue Für Spark-Jobs werden häufig Rohdaten verwendet, aber für eine effiziente verteilte Verarbeitung müssen Sie die Funktionen Ihres Datenquellenformats überprüfen. ETL

- Dateigröße — Wir empfehlen, die Dateigröße der Ein- und Ausgaben in einem moderaten Bereich zu halten (z. B. 128 MB). Zu kleine und zu große Dateien können Probleme verursachen.

Eine große Anzahl kleiner Dateien verursacht die folgenden Probleme:

- Starke Netzwerk-I/O-Last auf Amazon S3 aufgrund des Overheads, der erforderlich ist, um Anfragen (wie `ListGet`, oder`Head`) für viele Objekte zu stellen (im Vergleich zu einigen wenigen Objekten, die dieselbe Datenmenge speichern).
- Starke I/O- und Verarbeitungslast auf dem Spark-Treiber, was viele Partitionen und Aufgaben generiert und zu übermäßiger Parallelität führt.

Wenn Ihr Dateityp jedoch nicht teilbar ist (z. B. Gzip) und die Dateien zu groß sind, muss die Spark-Anwendung warten, bis eine einzelne Aufgabe das Lesen der gesamten Datei abgeschlossen hat.

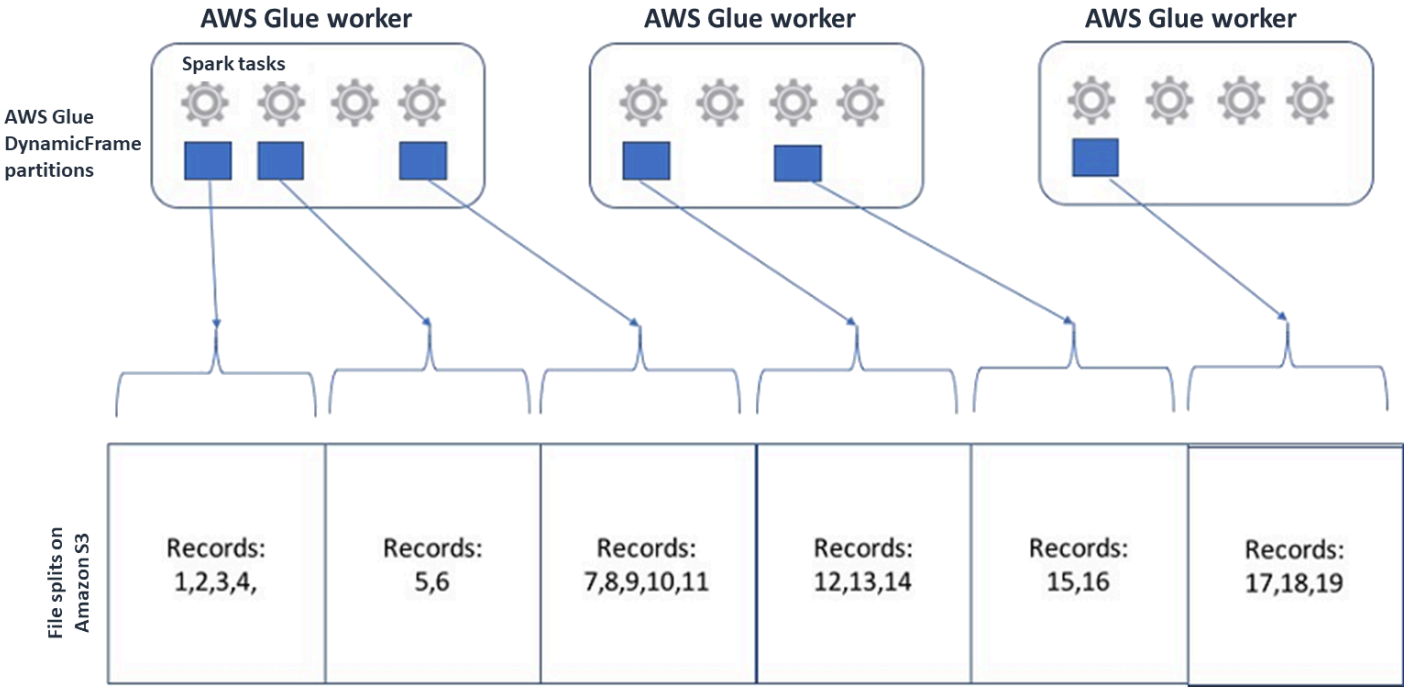
Um die übermäßige Parallelität zu reduzieren, die entsteht, wenn für jede kleine Datei eine Apache Spark-Aufgabe erstellt wird, verwenden Sie die Dateigruppierung für. DynamicFrames Dieser Ansatz verringert die Wahrscheinlichkeit, dass es beim Spark-Treiber zu einer OOM Ausnahme kommt. Um die Gruppierung von Dateien zu konfigurieren, legen Sie die `groupSize` Parameter `groupFiles` und fest. Im folgenden Codebeispiel wird das AWS Glue DynamicFrame API in einem ETL Skript mit diesen Parametern verwendet.

```
dyf = glueContext.create_dynamic_frame_from_options("s3",
    {'paths': ["s3://input-s3-path/"],
    'recurse':True,
    'groupFiles': 'inPartition',
    'groupSize': '1048576'},
    format="json")
```

- Komprimierung — Wenn Ihre S3-Objekte Hunderte von Megabyte groß sind, sollten Sie erwägen, sie zu komprimieren. Es gibt verschiedene Komprimierungsformate, die grob in zwei Typen eingeteilt werden können:
  - Nicht aufteilbare Komprimierungsformate wie Gzip erfordern, dass die gesamte Datei von einem Worker dekomprimiert wird.
  - Aufteilbare Komprimierungsformate wie bzip2 oder LZO (indexiert) ermöglichen die teilweise Dekomprimierung einer Datei, die parallelisiert werden kann.

  Bei Spark (und anderen gängigen Engines für verteilte Verarbeitung) teilen Sie Ihre Quelldatendatei in Teile auf, die Ihre Engine parallel verarbeiten kann. Diese Einheiten werden oft als Splits bezeichnet. Sobald Ihre Daten in einem teilbaren Format vorliegen, können die optimierten AWS Glue Reader Splits aus einem S3-Objekt abrufen, indem sie ihnen die `Range`

Option bieten, nur bestimmte `GetObject` API Blöcke abzurufen. Sehen Sie sich das folgende Diagramm an, um zu sehen, wie dies in der Praxis funktionieren würde.




Komprimierte Daten können Ihre Anwendung erheblich beschleunigen, sofern die Dateien entweder eine optimale Größe haben oder die Dateien aufgeteilt werden können. Die kleineren Datengrößen reduzieren die von Amazon S3 gescannten Daten und den Netzwerkverkehr von Amazon S3 zu Ihrem Spark-Cluster. Andererseits CPU ist mehr erforderlich, um Daten zu komprimieren und zu dekomprimieren. Der Umfang der erforderlichen Rechenleistung hängt vom Komprimierungsverhältnis Ihres Komprimierungsalgorithmus ab. Berücksichtigen Sie diesen Kompromiss bei der Auswahl Ihres teilbaren Komprimierungsformats.

> **Note**
>
> GZIP-Dateien sind zwar im Allgemeinen nicht splittbar, aber Sie können einzelne Parkettblöcke mit gzip komprimieren und diese Blöcke können parallelisiert werden.

- Dateiformat — Verwenden Sie ein Spaltenformat. Apache Parquet und Apache ORC sind beliebte spaltenförmige Datenformate. ORCSpeichern und speichern Sie Daten effizient, indem Sie spaltenbasierte Komprimierung verwenden, jede Spalte auf der Grundlage ihres Datentyps kodieren und komprimieren. Weitere Informationen zu Parquet-Kodierungen finden Sie unter Parquet-Kodierungsdefinitionen. Parquet-Dateien können auch aufgeteilt werden.

Spaltenformate gruppieren Werte nach Spalten und speichern sie zusammen in Blöcken. Wenn Sie Spaltenformate verwenden, können Sie Datenblöcke überspringen, die Spalten entsprechen, die Sie nicht verwenden möchten. Spark-Anwendungen können nur die Spalten abrufen, die Sie benötigen. Im Allgemeinen bedeuten bessere Komprimierungsraten oder das Überspringen von Datenblöcken, dass weniger Byte aus Amazon S3 gelesen werden, was zu einer besseren Leistung führt. Beide Formate unterstützen auch die folgenden Pushdown-Ansätze zur Reduzierung von I/O:

- Projection Pushdown — Projection Pushdown ist eine Technik, mit der nur die in Ihrer Anwendung angegebenen Spalten abgerufen werden. Sie geben Spalten in Ihrer Spark-Anwendung an, wie in den folgenden Beispielen gezeigt:
  - DataFrame Beispiel: `df.select("star_rating")`
  - SQLSpark-Beispiel: `spark.sql("select start_rating from <table>")`
- Predicate Pushdown — Predicate Pushdown ist eine Technik zur effizienten Verarbeitung von Klauseln. `WHERE GROUP BY` Beide Formate enthalten Datenblöcke, die Spaltenwerte darstellen. Jeder Block enthält Statistiken für den Block, z. B. Maximal- und Minimalwerte. Spark kann anhand dieser Statistiken bestimmen, ob der Block gelesen oder übersprungen werden soll, abhängig vom in der Anwendung verwendeten Filterwert. Um diese Funktion zu verwenden, fügen Sie den Bedingungen weitere Filter hinzu, wie in den folgenden Beispielen wie folgt dargestellt:
  - DataFrame Beispiel: `df.select("star_rating").filter("star_rating < 2")`
  - SQLSpark-Beispiel: `spark.sql("select * from <table> where star_rating < 2")`
- Dateilayout — Indem Sie Ihre S3-Daten in Objekten in unterschiedlichen Pfaden speichern, je nachdem, wie die Daten verwendet werden, können Sie relevante Daten effizient abrufen. Weitere Informationen finden Sie unter Organisieren von Objekten mithilfe von Präfixen in der Amazon S3 S3-Dokumentation. AWS Glue unterstützt das Speichern von Schlüsseln und Werten in Amazon S3 S3-Präfixen im Format`key=value`, wobei Ihre Daten nach dem Amazon S3 S3-Pfad partitioniert werden. Durch die Partitionierung Ihrer Daten können Sie die Menge der Daten einschränken, die von jeder nachgelagerten Analyseanwendung gescannt werden, wodurch die Leistung verbessert und die Kosten gesenkt werden. Weitere Informationen finden Sie unter Verwalten von Partitionen für die ETL Ausgabe in AWS Glue.

Durch die Partitionierung wird Ihre Tabelle in verschiedene Teile unterteilt, und die zugehörigen Daten werden in gruppierten Dateien gespeichert, die auf Spaltenwerten wie Jahr, Monat und Tag basieren, wie im folgenden Beispiel gezeigt.

```
# Partitioning by /YYYY/MM/DD
s3://<YourBucket>/year=2023/month=03/day=31/0000.gz
s3://<YourBucket>/year=2023/month=03/day=01/0000.gz
s3://<YourBucket>/year=2023/month=03/day=02/0000.gz
s3://<YourBucket>/year=2023/month=03/day=03/0000.gz
...
```

Sie können Partitionen für Ihren Datensatz definieren, indem Sie ihn mit einer Tabelle in der modellieren. AWS Glue Data Catalog Anschließend können Sie den Umfang der gescannten Daten einschränken, indem Sie Partitionen wie folgt bereinigen:

- Für AWS Glue DynamicFrame, setzen Sie `push_down_predicate` (oder`catalogPartitionPredicate`).

  ```
  dyf = Glue_context.create_dynamic_frame.from_catalog(
      database=src_database_name,
      table_name=src_table_name,
      push_down_predicate = "year='2023' and month ='03'",
  )
  ```

- Legen Sie für Spark DataFrame einen festen Pfad zum Bereinigen von Partitionen fest.

  ```
  df = spark.read.format("json").load("s3://<YourBucket>/year=2023/month=03/*/*.gz")
  ```

- Für Spark SQL können Sie die WHERE-Klausel festlegen, um Partitionen aus dem Datenkatalog zu löschen.

  ```
  df = spark.sql("SELECT * FROM <Table> WHERE year= '2023' and month = '03'")
  ```

- Um beim Schreiben Ihrer Daten nach Datum zu partitionieren AWS Glue, geben Sie partitionKeys DynamicFrame oder partitionBy() DataFrame zusammen mit den Datumsinformationen in Ihren Spalten wie folgt ein.

  - DynamicFrame

    ```
    glue_context.write_dynamic_frame_from_options(
    ```

```
    frame= dyf, connection_type='s3',format='parquet'
    connection_options= {
        'partitionKeys': ["year", "month", "day"],
        'path': 's3://<YourBucket>/<Prefix>/'
    }
)
```

- DataFrame

```
df.write.mode('append')\
    .partitionBy('year','month','day')\
    .parquet('s3://<YourBucket>/<Prefix>/')
```

Dies kann die Leistung der Nutzer Ihrer Ausgabedaten verbessern.

Wenn Sie nicht über die Möglichkeit verfügen, die Pipeline zu ändern, mit der Ihr Eingabe-Dataset erstellt wird, ist Partitionierung keine Option. Stattdessen können Sie nicht benötigte S3-Pfade ausschließen, indem Sie Glob-Muster verwenden. Legen Sie beim Einlesen Ausnahmen fest. DynamicFrame Der folgende Code schließt beispielsweise Tage in den Monaten 01 bis 09 im Jahr 2023 aus.

```
dyf = glueContext.create_dynamic_frame.from_catalog(
    database=db,
    table_name=table,
    additional_options = { "exclusions":"[\"**year=2023/month=0[1-9]/**\"]" },
    transformation_ctx='dyf'
)
```

Sie können Ausnahmen auch in den Tabelleneigenschaften im Datenkatalog festlegen:

- Schlüssel: `exclusions`
- Wert: `["**year=2023/month=0[1-9]/**"]`

- Zu viele Amazon S3 S3-Partitionen — Vermeiden Sie es, Ihre Amazon S3 S3-Daten in Spalten zu partitionieren, die einen großen Wertebereich enthalten, wie z. B. eine ID-Spalte mit Tausenden von Werten. Dies kann die Anzahl der Partitionen in Ihrem Bucket erheblich erhöhen, da die Anzahl der möglichen Partitionen das Produkt aller Felder ist, nach denen Sie partitioniert haben. Zu viele Partitionen können zu folgenden Problemen führen:
  - Erhöhte Latenz beim Abrufen von Partitionsmetadaten aus dem Datenkatalog

- Höhere Anzahl kleiner Dateien, was mehr Amazon S3 API S3-Anfragen erfordert (`ListGet`, und`Head`)

Wenn Sie beispielsweise einen Datumstyp in `partitionBy` oder festlegen`partitionKeys`, eignet sich eine Partitionierung auf Datumsebene für viele Anwendungsfälle. `yyyy/mm/dd` Es `yyyy/mm/dd/<ID>` könnten jedoch so viele Partitionen generiert werden, dass sich dies negativ auf die Gesamtleistung auswirken würde.

Andererseits erfordern einige Anwendungsfälle, wie z. B. Echtzeitverarbeitungsanwendungen, viele Partitionen wie`yyyy/mm/dd/hh`. Wenn Ihr Anwendungsfall umfangreiche Partitionen erfordert, sollten Sie die Verwendung von AWS Glue Partitionsindizes in Betracht ziehen, um die Latenz beim Abrufen von Partitionsmetadaten aus dem Datenkatalog zu reduzieren.

## Datenbanken und JDBC

Um den Datenscan beim Abrufen von Informationen aus einer Datenbank zu reduzieren, können Sie in einer Abfrage ein `where` Prädikat (oder eine SQL Klausel) angeben. Datenbanken, die keine SQL Schnittstelle bereitstellen, bieten ihren eigenen Mechanismus zum Abfragen oder Filtern.

Wenn Sie Verbindungen mit Java Database Connectivity (JDBC) verwenden, stellen Sie eine Auswahlabfrage mit der `where` Klausel für die folgenden Parameter bereit:

- Verwenden DynamicFrame Sie für die sampleQueryOption. Wenn Sie das `additional_options` Argument verwenden`create_dynamic_frame.from_catalog`, konfigurieren Sie es wie folgt.

```
query = "SELECT * FROM <TableName> where id = 'XX' AND"
datasource0 = glueContext.create_dynamic_frame.from_catalog(
  database = db,
  table_name = table,
  additional_options={
      "sampleQuery": query,
      "hashexpression": key,
      "hashpartitions": 10,
      "enablePartitioningForSampleQuery": True
    },
  transformation_ctx = "datasource0"
)
```

Wennusing `create_dynamic_frame.from_options`, konfigurieren Sie das `connection_options` Argument wie folgt.

```
query = "SELECT * FROM <TableName> where id = 'XX' AND"
datasource0 = glueContext.create_dynamic_frame.from_options(
    connection_type = connection,
    connection_options={
        "url": url,
        "user": user,
        "password": password,
        "dbtable": table,
        "sampleQuery": query,
        "hashexpression": key,
        "hashpartitions": 10,
        "enablePartitioningForSampleQuery": True
    }
)
```

- Verwenden Sie für DataFrame die Abfrageoption.

```
query = "SELECT * FROM <TableName> where id = 'XX'"
jdbcDF = spark.read \
.format('jdbc') \
.option('url', url) \
.option('user', user) \
.option('password', pwd) \
.option('query', query) \
.load()
```

- Verwenden Sie für Amazon Redshift AWS Glue 4.0 oder höher, um die Pushdown-Unterstützung im Amazon Redshift Spark-Connector zu nutzen.

```
dyf = glueContext.create_dynamic_frame.from_catalog(
    database = "redshift-dc-database-name",
    table_name = "redshift-table-name",
    redshift_tmp_dir = args["temp-s3-dir"],
    additional_options = {"aws_iam_role": "arn:aws:iam::role-account-id:role/rs-role-name"}
)
```

- Informationen zu anderen Datenbanken finden Sie in der Dokumentation zu dieser Datenbank.

### AWS Glue Optionen

- Um einen vollständigen Scan für alle fortlaufenden Auftragsausführungen zu vermeiden und nur Daten zu verarbeiten, die während der letzten Auftragsausführung nicht vorhanden waren, aktivieren Sie Job-Lesezeichen.
- Um die Menge der zu verarbeitenden Eingabedaten zu begrenzen, aktivieren Sie die begrenzte Ausführung mit Job-Lesezeichen. Dies trägt dazu bei, die Menge der gescannten Daten bei jeder Auftragsausführung zu reduzieren.

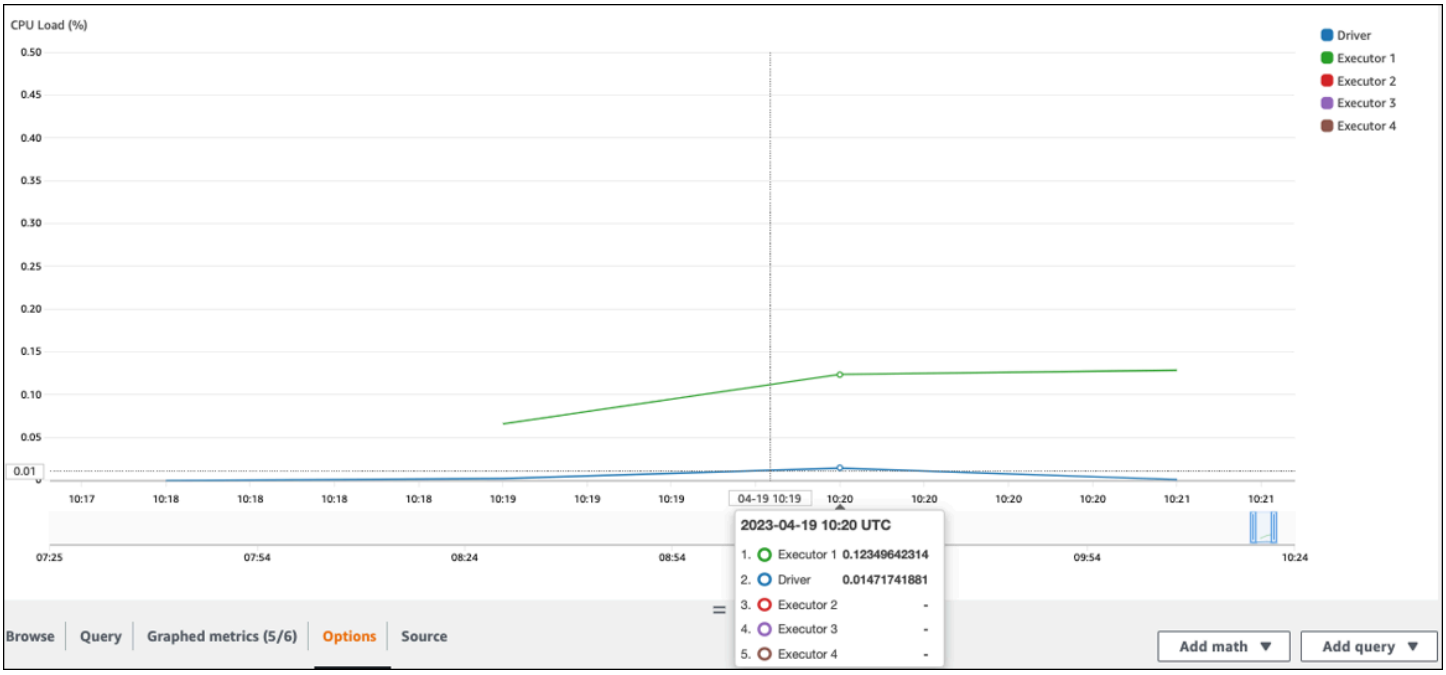
## Aufgaben parallelisieren

Um die Leistung zu optimieren, ist es wichtig, Aufgaben für das Laden und Transformieren von Daten zu parallelisieren. Wie wir unter Wichtige Themen in Apache Spark besprochen haben, ist die Anzahl der belastbaren Partitionen mit verteilten Datensätzen (RDD) wichtig, da sie den Grad der Parallelität bestimmt. Jede Aufgabe, die Spark erstellt, entspricht einer RDD Partition im Verhältnis 1:1. Um die beste Leistung zu erzielen, müssen Sie verstehen, wie die Anzahl der RDD Partitionen bestimmt und wie diese Anzahl optimiert wird.

Wenn Sie nicht genug Parallelität haben, werden die folgenden Symptome in CloudWatchMetriken und der Spark-Benutzeroberfläche aufgezeichnet.

### CloudWatch Metriken

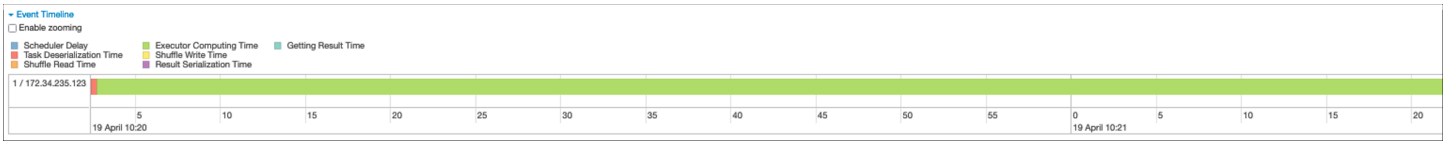
Überprüfen Sie die CPUAuslastung und die Speicherauslastung. Wenn einige Executoren während einer Phase Ihres Jobs keine Verarbeitung durchführen, ist es angebracht, die Parallelität zu verbessern. In diesem Fall führte Executor 1 während des visualisierten Zeitrahmens eine Aufgabe aus, die übrigen Executoren (2, 3 und 4) jedoch nicht. Sie können daraus schließen, dass diesen Executoren vom Spark-Treiber keine Aufgaben zugewiesen wurden.




## Spark-Benutzeroberfläche

Auf der Registerkarte Phase in der Spark-Benutzeroberfläche können Sie die Anzahl der Aufgaben in einer Phase sehen. In diesem Fall hat Spark nur eine Aufgabe ausgeführt.


Darüber hinaus zeigt die Event-Timeline, dass Executor 1 eine Aufgabe bearbeitet. Das bedeutet, dass die Arbeit in dieser Phase ausschließlich auf einem Testamentsvollstrecker ausgeführt wurde, während die anderen inaktiv waren.




Wenn Sie diese Symptome beobachten, versuchen Sie es mit den folgenden Lösungen für jede Datenquelle.

## Parallelisieren Sie das Laden von Daten aus Amazon S3

Um Datenladungen aus Amazon S3 zu parallelisieren, überprüfen Sie zunächst die Standardanzahl von Partitionen. Sie können dann manuell eine Zielanzahl von Partitionen bestimmen, achten Sie jedoch darauf, zu viele Partitionen zu vermeiden.

Ermitteln Sie die Standardanzahl von Partitionen

Für Amazon S3 wird die anfängliche Anzahl von RDD Spark-Partitionen (von denen jede einer Spark-Aufgabe entspricht) durch die Funktionen Ihres Amazon S3 S3-Datensatzes (z. B. Format, Komprimierung und Größe) bestimmt. Wenn Sie einen AWS Glue DynamicFrame oder einen Spark DataFrame aus in Amazon S3 gespeicherten CSV Objekten erstellen, kann die anfängliche Anzahl von RDD Partitionen (`NumPartitions`) ungefähr wie folgt berechnet werden:

- Objektgröße **<=** 64 MB: `NumPartitions = Number of Objects`
- Objektgröße **>** 64 MB: `NumPartitions = Total Object Size / 64 MB`
- Nicht teilbar (gzip): `NumPartitions = Number of Objects`

Wie im Abschnitt Reduzieren Sie die Menge der gescannten Daten beschrieben, unterteilt Spark große S3-Objekte in Splits, die parallel verarbeitet werden können. Wenn das Objekt größer als die geteilte Größe ist, teilt Spark das Objekt auf und erstellt für jede RDD Teilung eine Partition (und Aufgabe). Die Split-Größe von Spark basiert auf Ihrem Datenformat und Ihrer Laufzeitumgebung, aber das ist eine vernünftige erste Näherung. Einige Objekte werden mit nicht teilbaren Komprimierungsformaten wie Gzip komprimiert, sodass Spark sie nicht aufteilen kann.

Der `NumPartitions` Wert kann je nach Datenformat, Komprimierung, AWS Glue Version, Anzahl der AWS Glue Worker und Spark-Konfiguration variieren.

Wenn Sie beispielsweise ein einzelnes `csv.gz` 10-GB-Objekt mit einem Spark laden DataFrame, erstellt der Spark-Treiber nur eine RDD Partition (`NumPartitions=1`), da Gzip nicht teilbar ist. Dies führt zu einer starken Belastung eines bestimmten Spark-Executors und den verbleibenden Executoren werden keine Aufgaben zugewiesen, wie in der folgenden Abbildung beschrieben.

Überprüfen Sie die tatsächliche Anzahl der Aufgaben (**`NumPartitions`**) für die Phase auf der Registerkarte Stage der Spark Web UI, oder führen Sie die Ausführung **`df.rdd.getNumPartitions()`** in Ihrem Code aus, um die Parallelität zu überprüfen.

Wenn Sie auf eine 10-GB-GZIP-Datei stoßen, prüfen Sie, ob das System, das diese Datei generiert, sie in einem aufteilbaren Format generieren kann. Wenn dies keine Option ist, müssen

Sie möglicherweise die Clusterkapazität skalieren, um die Datei verarbeiten zu können. Um Transformationen auf effiziente Weise auf den von Ihnen geladenen Daten auszuführen, müssen Sie mithilfe der Repartitionierung eine RDD Neuverteilung zwischen den Workern in Ihrem Cluster vornehmen.

Ermitteln Sie manuell eine Zielanzahl von Partitionen

Abhängig von den Eigenschaften Ihrer Daten und der Implementierung bestimmter Funktionen durch Spark kann es sein, dass Sie am Ende einen niedrigen `NumPartitions` Wert erhalten, obwohl die zugrunde liegende Arbeit immer noch parallelisiert werden kann. Wenn der `NumPartitions` Wert zu klein ist, führen Sie den Befehl aus, `df.repartition(N)` um die Anzahl der Partitionen zu erhöhen, sodass die Verarbeitung auf mehrere Spark-Executoren verteilt werden kann.

In diesem Fall `df.repartition(100)` wird die Anzahl der ausgeführten Partitionen `NumPartitions` von 1 auf 100 erhöht, sodass 100 Partitionen Ihrer Daten erstellt werden, von denen jede eine Aufgabe hat, die den anderen Executoren zugewiesen werden kann.

Bei diesem Vorgang werden die `repartition(N)` gesamten Daten gleichmäßig aufgeteilt (10 GB/100 Partitionen = 100 MB/Partition), wodurch Datenverzerrungen zu bestimmten Partitionen vermieden werden.

> **Note**
>
> Wenn ein Shuffle-Vorgang wie z. B. ausgeführt `join` wird, wird die Anzahl der Partitionen je nach dem Wert von oder dynamisch erhöht oder verringert. `spark.sql.shuffle.partitions` `spark.default.parallelism` Dies ermöglicht einen effizienteren Datenaustausch zwischen Spark-Executoren. Weitere Informationen finden Sie in der Spark-Dokumentation.

Ihr Ziel bei der Festlegung der Zielanzahl von Partitionen besteht darin, die Nutzung der bereitgestellten AWS Glue Worker zu maximieren. Die Anzahl der AWS Glue Worker und die Anzahl der Spark-Aufgaben hängen von der Anzahl der vCPUs ab. Spark unterstützt eine Aufgabe für jeden CPU v-Core. In AWS Glue Version 3.0 oder höher können Sie mithilfe der folgenden Formel eine Zielanzahl von Partitionen berechnen.

```
# Calculate NumPartitions by WorkerType
numExecutors = (NumberOfWorkers - 1)
numSlotsPerExecutor =
```

```
  4 if WorkerType is G.1X
  8 if WorkerType is G.2X
  16 if WorkerType is G.4X
  32 if WorkerType is G.8X
NumPartitions = numSlotsPerExecutor * numExecutors

# Example: Glue 4.0 / G.1X / 10 Workers
numExecutors = ( 10 - 1 ) = 9  # 1 Worker reserved on Spark Driver
numSlotsPerExecutor       = 4  # G.1X has 4 vCpu core ( Glue 3.0 or later )
NumPartitions = 9  * 4    = 36
```

In diesem Beispiel stellt jeder G.1X-Worker einem Spark-Executor () vier CPU V-Kerne zur Verfügung. `spark.executor.cores = 4` Spark unterstützt eine Aufgabe für jeden CPU v-Core, sodass G.1X-Spark-Executoren vier Aufgaben gleichzeitig ausführen können (). `numSlotPerExecutor` Diese Anzahl von Partitionen nutzt den Cluster voll aus, wenn Aufgaben die gleiche Zeit in Anspruch nehmen. Einige Aufgaben dauern jedoch länger als andere, wodurch Kerne im Leerlauf entstehen. In diesem Fall sollten Sie eine Multiplikation mit `numPartitions` 2 oder 3 in Betracht ziehen, um Engpassaufgaben aufzuschlüsseln und effizient zu planen.

Zu viele Partitionen

Eine übermäßige Anzahl von Partitionen führt zu einer übermäßigen Anzahl von Aufgaben. Dies führt zu einer starken Belastung des Spark-Treibers aufgrund des Mehraufwands im Zusammenhang mit der verteilten Verarbeitung, wie z. B. Verwaltungsaufgaben und dem Datenaustausch zwischen Spark-Executoren.

Wenn die Anzahl der Partitionen in Ihrem Job wesentlich größer ist als Ihre Zielanzahl von Partitionen, sollten Sie erwägen, die Anzahl der Partitionen zu reduzieren. Sie können Partitionen mithilfe der folgenden Optionen reduzieren:

- Wenn Ihre Dateien sehr klein sind, verwenden Sie AWS Glue groupFiles. Sie können die übermäßige Parallelität reduzieren, die sich aus dem Start einer Apache Spark-Aufgabe zur Verarbeitung jeder Datei ergibt.
- Wird verwendet`coalesce(N)`, um Partitionen zusammenzuführen. Dies ist ein kostengünstiger Prozess. Beim Reduzieren der Anzahl der Partitionen `coalesce(N)` wird der Vorzug `repartition(N)` gegeben, `repartition(N)` da durch Mischen die Anzahl der Datensätze in jeder Partition gleichmäßig verteilt wird. Das erhöht die Kosten und den Verwaltungsaufwand.
- Verwenden Sie Spark 3.x Adaptive Query Execution. Wie im Abschnitt Die wichtigsten Themen in Apache Spark beschrieben, bietet Adaptive Query Execution eine Funktion zum automatischen

Zusammenführen der Anzahl von Partitionen. Sie können diesen Ansatz verwenden, wenn Sie die Anzahl der Partitionen erst kennen, wenn Sie die Ausführung durchgeführt haben.

## Parallelisieren Sie das Laden von Daten JDBC

Die Anzahl der RDD Spark-Partitionen wird durch die Konfiguration bestimmt. Beachten Sie, dass standardmäßig nur eine einzige Aufgabe ausgeführt wird, um einen gesamten Quelldatensatz mithilfe einer `SELECT` Abfrage zu scannen.

AWS Glue DynamicFrames Sowohl Spark als auch Spark DataFrames unterstützen das parallelisierte Laden JDBC von Daten über mehrere Aufgaben hinweg. Dies wird erreicht, indem `where` Prädikate verwendet werden, um eine `SELECT` Abfrage in mehrere Abfragen aufzuteilen. Um Lesevorgänge von zu parallelisierenJDBC, konfigurieren Sie die folgenden Optionen:

- Für AWS Glue DynamicFrame, setze `hashfield` (oder`hashexpression`)) und. `hashpartition` Weitere Informationen finden Sie unter Paralleles Lesen aus JDBC Tabellen.

```
connection_mysql8_options = {
    "url": "jdbc:mysql://XXXXXXXXXX.XXXXXXX.us-east-1.rds.amazonaws.com:3306/test",
    "dbtable": "medicare_tb",
    "user": "test",
    "password": "XXXXXXXXX",
    "hashexpression":"id",
    "hashpartitions":"10"
}
datasource0 = glueContext.create_dynamic_frame.from_options(
    'mysql',
    connection_options=connection_mysql8_options,
    transformation_ctx= "datasource0"
)
```

- Für Spark legen Sie DataFramenum`Partitions`, `partitionColumnlowerBound`, und festupper`Bound`. Weitere Informationen finden Sie unter JDBCZu anderen Datenbanken.

```
df = spark.read \
    .format("jdbc") \
    .option("url", "jdbc:mysql://XXXXXXXXXX.XXXXXXX.us-east-1.rds.amazonaws.com:3306/
test") \
    .option("dbtable", "medicare_tb") \
    .option("user", "test") \
```

```
    .option("password", "XXXXXXXXXX") \
    .option("partitionColumn", "id") \
    .option("numPartitions", "10") \
    .option("lowerBound", "0") \
    .option("upperBound", "1141455") \
    .load()

df.write.format("json").save("s3://bucket_name/Tests/sparkjdbc/with_parallel/")
```

## Parallelisieren Sie das Laden von Daten aus DynamoDB, wenn Sie den Connector verwenden ETL

Die Anzahl der RDD Spark-Partitionen wird durch den Parameter bestimmt. `dynamodb.splits` Um Lesevorgänge aus Amazon DynamoDB zu parallelisieren, konfigurieren Sie die folgenden Optionen:

- Erhöhen Sie den Wert von. `dynamodb.splits`
- Optimieren Sie den Parameter, indem Sie der Formel folgen, die unter Verbindungstypen und Optionen für ETL in AWS Glue for Spark erklärt wird.

## Parallelisieren Sie das Laden von Daten aus Kinesis Data Streams

Die Anzahl der RDD Spark-Partitionen wird durch die Anzahl der Shards im Amazon Kinesis Data Streams Streams-Quelldatenstream bestimmt. Wenn Sie nur wenige Shards in Ihrem Datenstream haben, wird es nur wenige Spark-Aufgaben geben. Dies kann zu einer geringen Parallelität bei nachgelagerten Prozessen führen. Um Lesevorgänge aus Kinesis Data Streams zu parallelisieren, konfigurieren Sie die folgenden Optionen:

- Erhöhen Sie die Anzahl der Shards, um mehr Parallelität beim Laden von Daten aus Kinesis Data Streams zu erreichen.
- Wenn Ihre Logik im Mikro-Batch komplex genug ist, sollten Sie erwägen, die Daten zu Beginn des Batches neu zu partitionieren, nachdem Sie nicht benötigte Spalten gelöscht haben.

Weitere Informationen finden Sie unter Bewährte Methoden zur Kosten- und Leistungsoptimierung für Streaming-Jobs. AWS Glue ETL

### Parallelisieren Sie Aufgaben nach dem Laden der Daten

Um Aufgaben nach dem Laden von Daten zu parallelisieren, erhöhen Sie die Anzahl der RDD Partitionen, indem Sie die folgenden Optionen verwenden:

- Partitionieren Sie Daten neu, um eine größere Anzahl von Partitionen zu generieren, insbesondere direkt nach dem ersten Laden, wenn der Ladevorgang selbst nicht parallelisiert werden konnte.

  Rufen Sie `repartition()` entweder auf DynamicFrame oder DataFrame auf und geben Sie dabei die Anzahl der Partitionen an. Eine gute Faustregel lautet, dass die Anzahl der verfügbaren Kerne zwei- oder dreimal so hoch ist.

  Beim Schreiben einer partitionierten Tabelle kann dies jedoch zu einer Explosion von Dateien führen (jede Partition kann potenziell eine Datei in jeder Tabellenpartition generieren). Um dies zu vermeiden, können Sie Ihre Datei DataFrame nach Spalten neu partitionieren. Dabei werden die Spalten der Tabellenpartition verwendet, sodass die Daten vor dem Schreiben organisiert werden. Sie können eine höhere Anzahl von Partitionen angeben, ohne dass kleine Dateien auf den Tabellenpartitionen gespeichert werden. Achten Sie jedoch darauf, Datenverzerrungen zu vermeiden, bei denen einige Partitionswerte die meisten Daten enthalten und die Ausführung der Aufgabe verzögern.
- Wenn es zu Shuffles kommt, erhöhen Sie den Wert. `spark.sql.shuffle.partitions` Dies kann auch bei Speicherproblemen beim Mischen helfen.

  Wenn Sie mehr als 2.001 Shuffle-Partitionen haben, verwendet Spark ein komprimiertes Speicherformat. Wenn Sie eine Zahl haben, die dieser Zahl nahe kommt, möchten Sie vielleicht den `spark.sql.shuffle.paritions` Wert über dieser Grenze setzen, um eine effizientere Darstellung zu erhalten.

## Optimieren Sie Shuffles

Bestimmte Operationen, wie z. B. `join()` und, erforderngroupByKey(), dass Spark eine Zufallswiedergabe durchführt. Der Shuffle ist der Mechanismus von Spark zur Umverteilung von Daten, sodass sie zwischen den Partitionen unterschiedlich gruppiert werden. RDD Shuffling kann dabei helfen, Leistungsengpässe zu beheben. Da das Mischen jedoch in der Regel das Kopieren von Daten zwischen Spark-Executoren beinhaltet, ist das Mischen ein komplexer und kostspieliger Vorgang. Beim Mischen fallen beispielsweise die folgenden Kosten an:

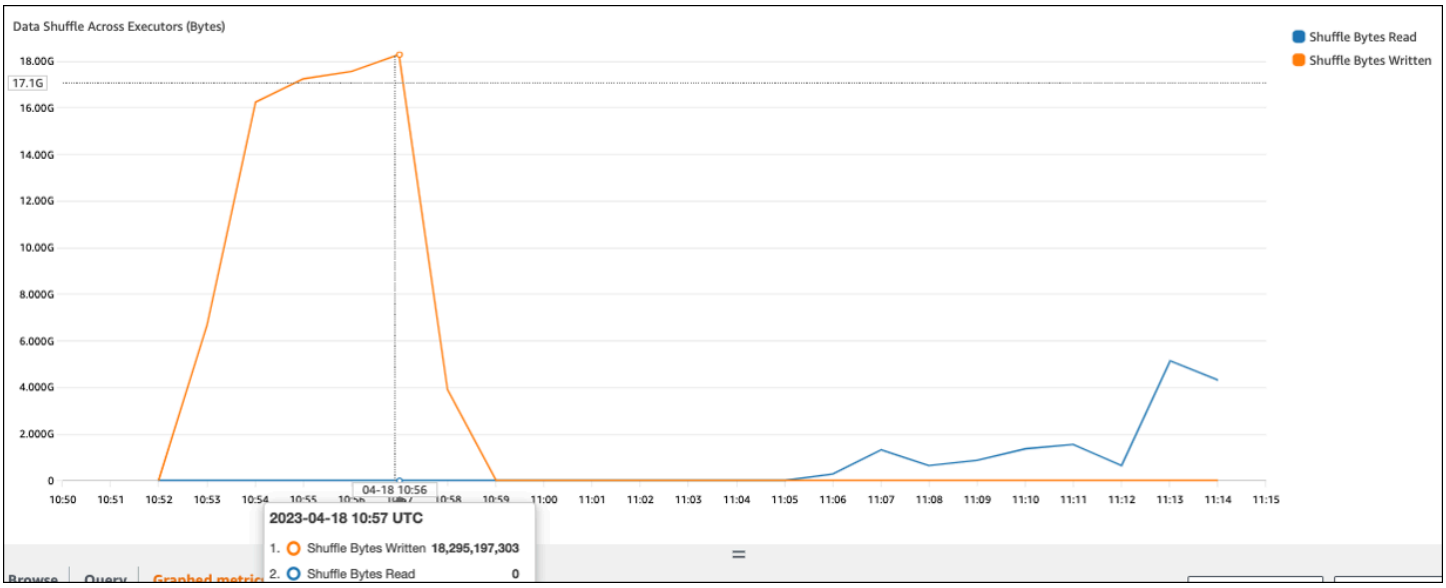
- Festplatten-I/O:
  - Generiert eine große Anzahl von Zwischendateien auf der Festplatte.
- Netzwerk-I/O:
  - Benötigt viele Netzwerkverbindungen (Anzahl der Verbindungen =`Mapper × Reducer`).
  - Da Datensätze zu neuen RDD Partitionen zusammengefasst werden, die möglicherweise auf einem anderen Spark-Executor gehostet werden, kann ein erheblicher Teil Ihres Datensatzes zwischen Spark-Executoren über das Netzwerk verschoben werden.
- CPUund Speicherlast:
  - Sortiert Werte und führt Datensätze zusammen. Diese Operationen sind für den Testamentsvollstrecker geplant, wodurch der Testamentsvollstrecker stark belastet wird.

Shuffle ist einer der wichtigsten Faktoren für Leistungseinbußen Ihrer Spark-Anwendung. Beim Speichern der Zwischendaten kann dadurch Speicherplatz auf der lokalen Festplatte des Executors belegt werden, wodurch der Spark-Job fehlschlägt.

Sie können Ihre Shuffle-Leistung anhand von CloudWatch Metriken und in der Spark-Benutzeroberfläche beurteilen.

## CloudWatch Metriken

Wenn der Wert für Shuffle Bytes Written im Vergleich zu Shuffle Bytes Read hoch ist, verwendet Ihr Spark-Job möglicherweise Shuffle-Operationen wie oder. `join()` `groupByKey()`

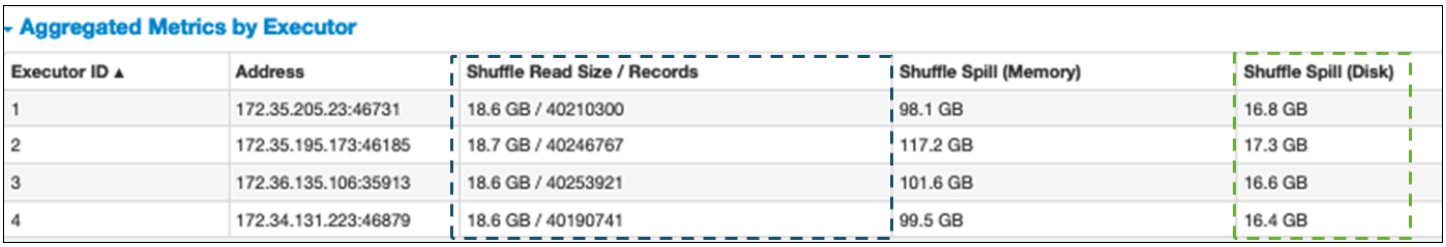


## Spark-Benutzeroberfläche

Auf der Registerkarte Stage der Spark-Benutzeroberfläche können Sie die Werte für Shuffle Read Size und Records überprüfen. Sie können es auch auf der Registerkarte Executors sehen.

Im folgenden Screenshot tauscht jeder Executor etwa 18,6 GB/4020.000 Datensätze mit dem Shuffle-Verfahren aus, was einer Gesamtgröße von etwa 75 GB beim Shuffle-Lesen entspricht.

In der Spalte Shuffle Spill (Disk) wird angezeigt, dass große Datenmengen auf die Festplatte übertragen werden, was zu einer vollen Festplatte oder zu Leistungseinbußen führen kann.



Aggregated Metrics by Executor

| Executor ID ▲ | Address | Shuffle Read Size / Records | Shuffle Spill (Memory) | Shuffle Spill (Disk) |
|---|---|---|---|---|
| 1 | 172.35.205.23:46731 | 18.6 GB / 40210300 | 98.1 GB | 16.8 GB |
| 2 | 172.35.195.173:46185 | 18.7 GB / 40246767 | 117.2 GB | 17.3 GB |
| 3 | 172.36.135.106:35913 | 18.6 GB / 40253921 | 101.6 GB | 16.6 GB |
| 4 | 172.34.131.223:46879 | 18.6 GB / 40190741 | 99.5 GB | 16.4 GB |

Wenn Sie diese Symptome beobachten und die Phase im Vergleich zu Ihren Leistungszielen zu lange dauert, oder wenn sie mit `Out Of Memory` oder ohne `No space left on device` Fehler fehlschlägt, sollten Sie die folgenden Lösungen in Betracht ziehen.

### Optimieren Sie die Verbindung

Die `join()` Operation, die Tabellen verknüpft, ist die am häufigsten verwendete Shuffle-Operation, stellt jedoch häufig einen Leistungsengpass dar. Da das Zusammenführen ein kostspieliger Vorgang ist, empfehlen wir, ihn nicht zu verwenden, es sei denn, er ist für Ihre Geschäftsanforderungen unerlässlich. Vergewissern Sie sich, dass Sie Ihre Datenpipeline effizient nutzen, indem Sie die folgenden Fragen stellen:

- Berechnen Sie eine Verknüpfung neu, die auch in anderen Jobs ausgeführt wird, die Sie wiederverwenden können?
- Führen Sie eine Verbindung durch, um Fremdschlüssel in Werte aufzulösen, die von den Benutzern Ihrer Ausgabe nicht verwendet werden?

Nachdem Sie sich vergewissert haben, dass Ihre Verbindungsvorgänge für Ihre Geschäftsanforderungen unerlässlich sind, sehen Sie sich die folgenden Optionen an, um Ihre Verknüpfung so zu optimieren, dass sie Ihren Anforderungen entspricht.

### Verwenden Sie vor dem Beitritt Pushdown

Filtern Sie unnötige Zeilen und Spalten in der heraus, DataFrame bevor Sie eine Verknüpfung durchführen. Dies hat die folgenden Vorteile:

- Reduziert die Menge der Datenübertragung beim Shuffle
- Reduziert den Verarbeitungsaufwand im Spark-Executor
- Reduziert den Umfang der gescannten Daten

```
# Default
df_joined  = df1.join(df2, ["product_id"])

# Use Pushdown
df1_select =
 df1.select("product_id","product_title","star_rating").filter(col("star_rating")>=4.0)
df2_select = df2.select("product_id","category_id")
df_joined  = df1_select.join(df2_select, ["product_id"])
```

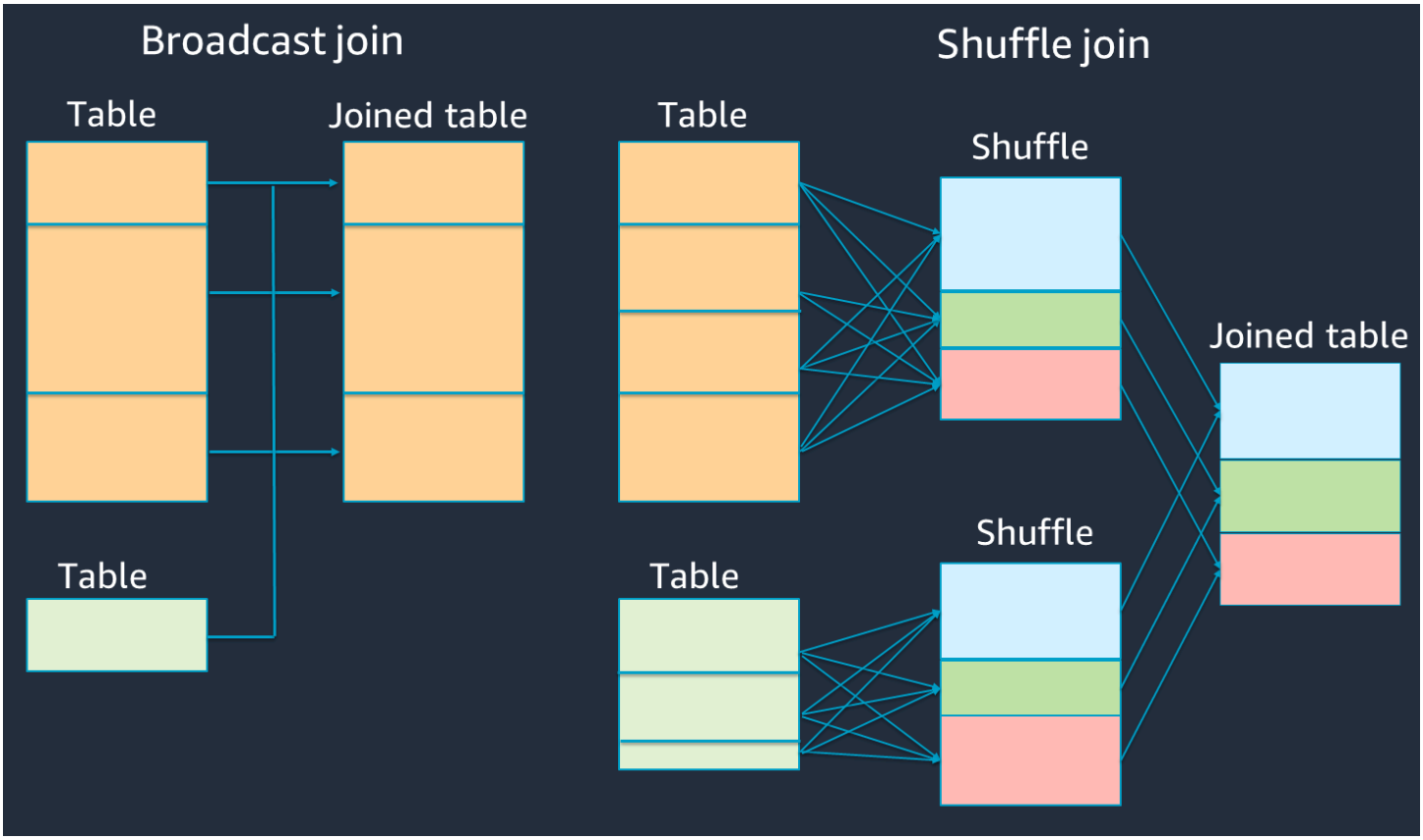
### Verwenden Sie DataFrame Join

Versuchen Sie, anstelle von RDD API oder DynamicFrame join ein Spark-High-Level API wie Spark SQL DataFrame, und Datasets zu verwenden. Sie können DataFrame mit DynamicFrame einem Methodenaufruf wie `dyf.toDF()` z. Wie im Abschnitt Die wichtigsten Themen in Apache Spark beschrieben, nutzen diese Join-Operationen intern die Vorteile der Abfrageoptimierung durch den Catalyst-Optimierer.

### Hash-Joins und Hinweise per Shuffle und Broadcast

Spark unterstützt zwei Arten von Verknüpfungen: Shuffle-Join und Broadcast-Hash-Join. Ein Broadcast-Hash-Join erfordert kein Mischen und kann weniger Verarbeitung erfordern als ein Shuffle-Join. Er ist jedoch nur anwendbar, wenn eine kleine Tabelle mit einer großen Tabelle verknüpft wird. Wenn Sie eine Tabelle verknüpfen, die in den Speicher eines einzelnen Spark-Executors passt, sollten Sie die Verwendung eines Broadcast-Hash-Joins in Betracht ziehen.

Das folgende Diagramm zeigt die allgemeine Struktur und die einzelnen Schritte eines Broadcast-Hash-Joins und eines Shuffle-Joins.




Die Einzelheiten der einzelnen Verknüpfungen lauten wie folgt:

- Shuffle-Join:
  - Der Shuffle-Hash-Join verbindet zwei Tabellen ohne Sortierung und verteilt den Join zwischen den beiden Tabellen. Er eignet sich für Verknüpfungen kleiner Tabellen, die im Speicher des Spark-Executors gespeichert werden können.
  - Der Sort-Merge-Join verteilt die beiden Tabellen, die verknüpft werden sollen, nach Schlüsseln und sortiert sie, bevor sie zusammengefügt werden. Er eignet sich für Verknüpfungen großer Tabellen.
- Broadcast-Hash-Join:
  - Bei einem Broadcast-Hash-Join wird die kleinere RDD OR-Tabelle an jeden der Worker-Knoten übertragen. Dann führt er eine kartenseitige Kombination mit jeder Partition der größeren RDD Partition oder Tabelle durch.

    Es eignet sich für Verknüpfungen, wenn eine Ihrer Tabellen RDDs oder eine Tabelle in den Arbeitsspeicher passt oder so angepasst werden kann, dass sie in den Speicher passt. Wenn möglich, ist es von Vorteil, einen Broadcast-Hash-Join durchzuführen, da dafür kein Shuffle

erforderlich ist. Mithilfe eines Beitrittshinweises können Sie wie folgt einen Broadcast-Join von Spark anfordern.

```
# DataFrame
from pySpark.sql.functions import broadcast
df_joined= df_big.join(broadcast(df_small), right_df[key] == left_df[key],
 how='inner')

-- SparkSQL
SELECT /*+ BROADCAST(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
```

Weitere Informationen zu Hinweisen zum Beitritt finden Sie unter Hinweise zum Beitritt.

In AWS Glue Version 3.0 und höher können Sie die Vorteile von Broadcast-Hash-Joins automatisch nutzen, indem Sie Adaptive Query Execution und zusätzliche Parameter aktivieren. Adaptive Query Execution konvertiert einen Sort-Merge-Join in einen Broadcast-Hash-Join, wenn die Laufzeitstatistiken auf beiden Seiten kleiner als der Schwellenwert für adaptive Broadcast-Hash-Joins sind.

In AWS Glue 3.0 können Sie Adaptive Query Execution aktivieren, indem Sie folgende Einstellungen vornehmen. `spark.sql.adaptive.enabled=true` Adaptive Abfrageausführung ist in AWS Glue 4.0 standardmäßig aktiviert.

Sie können zusätzliche Parameter für Shuffles und Broadcast-Hash-Joins festlegen:

- `spark.sql.adaptive.localShuffleReader.enabled`
- `spark.sql.adaptive.autoBroadcastJoinThreshold`

Weitere Informationen zu verwandten Parametern finden Sie unter Sort-Merge-Join in Broadcast-Join konvertieren.

In AWS Glue Version 3.0 und höher können Sie andere Join-Hinweise für die Zufallswiedergabe verwenden, um Ihr Verhalten zu optimieren.

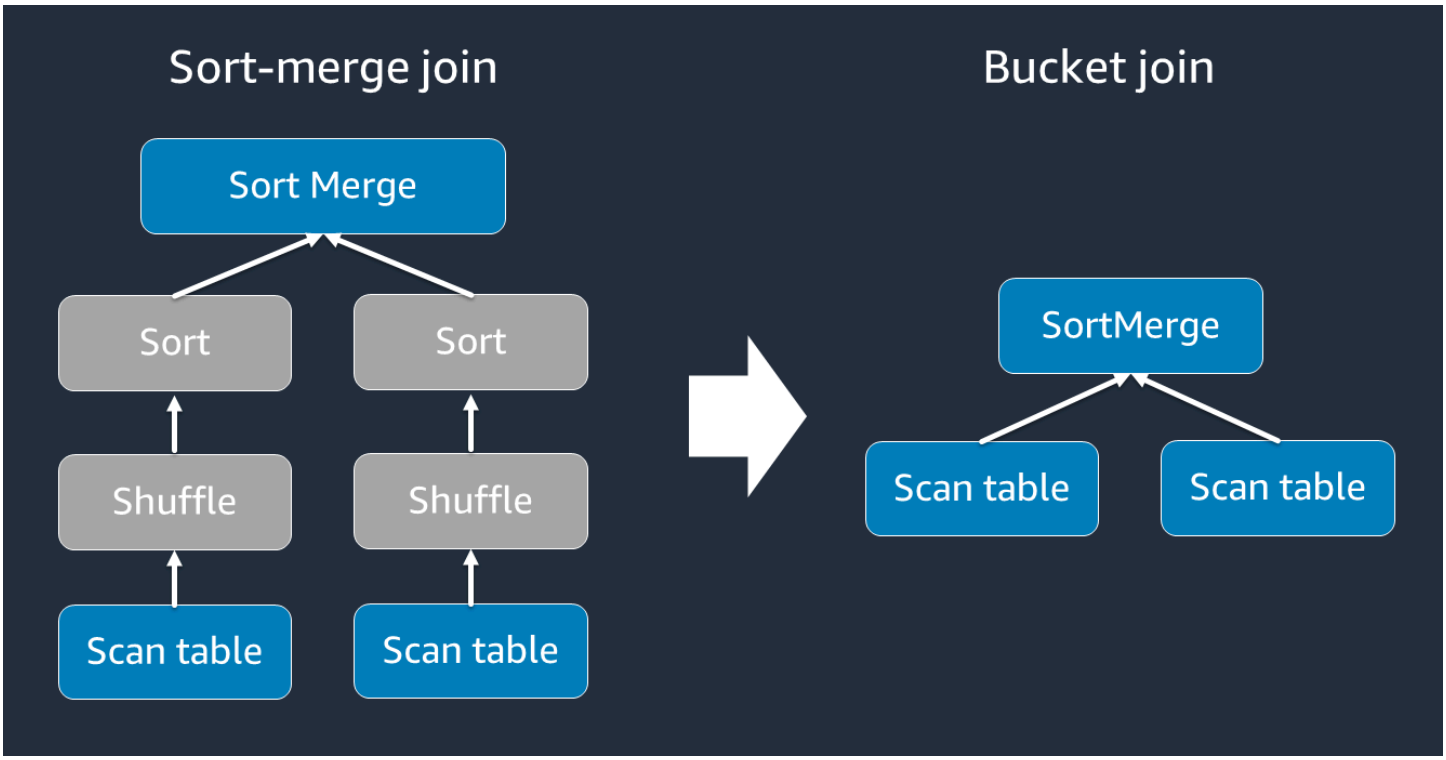
```
-- Join Hints for shuffle sort merge join
SELECT /*+ SHUFFLE_MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGEJOIN(t2) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
SELECT /*+ MERGE(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
```

```
-- Join Hints for shuffle hash join
SELECT /*+ SHUFFLE_HASH(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;

-- Join Hints for shuffle-and-replicate nested loop join
SELECT /*+ SHUFFLE_REPLICATE_NL(t1) /  FROM t1 INNER JOIN t2 ON t1.key = t2.key;
```

### Verwenden Sie Bucketing

Die Verknüpfung zwischen Sortierung und Zusammenführung erfordert zwei Phasen: Mischen und Sortieren und dann Zusammenführen. Diese beiden Phasen können den Spark-Executor überlasten OOM und zu Leistungseinbußen führen, wenn einige der Executoren zusammenführen und andere gleichzeitig sortieren. In solchen Fällen ist es möglicherweise möglich, mithilfe von Bucketing effizient eine Verbindung herzustellen. Beim Bucketing werden Ihre Eingaben auf Join-Schlüsseln vorab gemischt und vorsortiert und diese sortierten Daten dann in eine Zwischentabelle geschrieben. Die Kosten für die Schritte Mischen und Sortieren können beim Zusammenfügen großer Tabellen reduziert werden, indem die sortierten Zwischentabellen im Voraus definiert werden.




Bucket-Tabellen eignen sich für folgende Zwecke:

- Daten, die häufig über denselben Schlüssel verknüpft werden, wie `account_id`

- Es werden kumulative Tagestabellen geladen, z. B. Basis- und Delta-Tabellen, die in einer gemeinsamen Spalte zusammengefasst werden könnten

Mithilfe des folgenden Codes können Sie eine Bucket-Tabelle erstellen.

```
df.write.bucketBy(50, "account_id").sortBy("age").saveAsTable("bucketed_table")
```

Neupartitionierung DataFrames der Join-Schlüssel vor dem Join

Verwenden Sie die folgenden Anweisungen, um die beiden DataFrames auf den Join-Schlüsseln vor dem Join neu zu partitionieren.

```
df1_repartitioned = df1.repartition(N,"join_key")
df2_repartitioned = df2.repartition(N,"join_key")
df_joined  = df1_repartitioned.join(df2_repartitioned,"product_id")
```

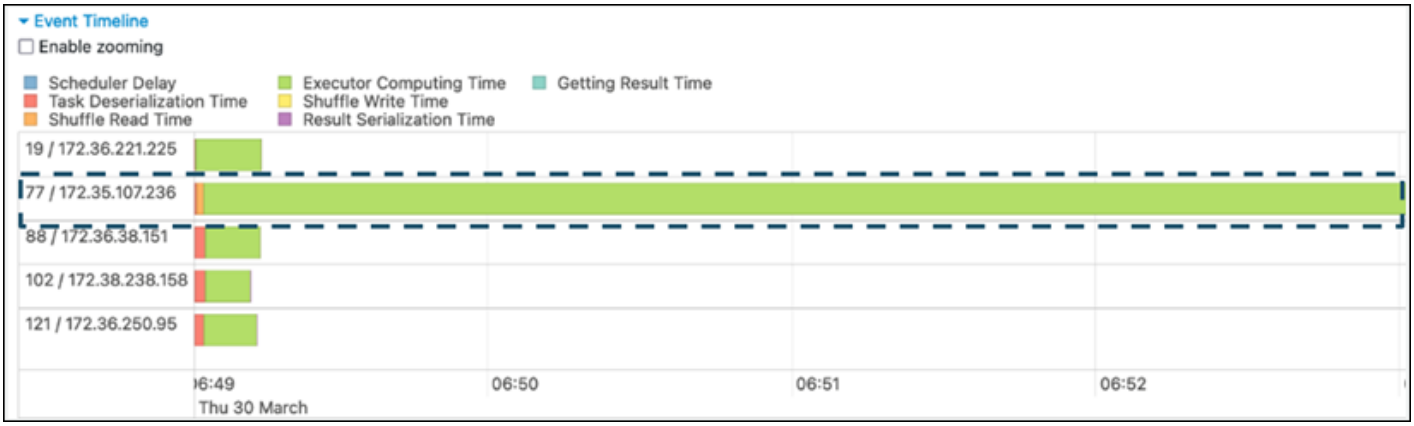
Dadurch werden zwei (immer noch getrennt) RDDs auf dem Join-Schlüssel partitioniert, bevor der Join initiiert wird. Wenn die beiden Dateien auf demselben Schlüssel mit demselben Partitionierungscode partitioniert RDDs sind, besteht eine hohe Wahrscheinlichkeit, dass sich RDD Datensätze, die Sie zusammenfügen möchten, auf demselben Worker befinden, bevor Sie zum Join wechseln. Dies kann die Leistung verbessern, da die Netzwerkaktivität und das Datengefälle während der Verknüpfung reduziert werden.

## Überwinden Sie Datenverzerrungen

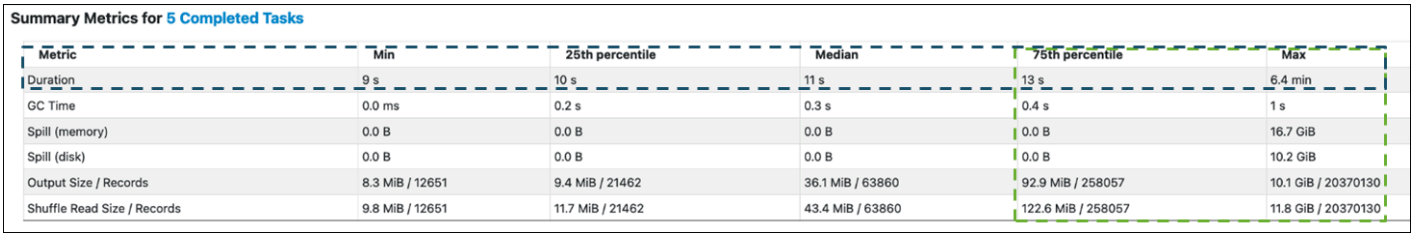
Datenverzerrungen sind eine der häufigsten Ursachen für Engpässe bei Spark-Jobs. Es tritt auf, wenn Daten nicht gleichmäßig auf Partitionen verteilt sind. RDD Dies führt dazu, dass Aufgaben für diese Partition viel länger dauern als für andere, was die Gesamtverarbeitungszeit der Anwendung verzögert.

Analysieren Sie die folgenden Kennzahlen in der Spark-Benutzeroberfläche, um Datenverzerrungen zu identifizieren:

- Sehen Sie sich auf der Registerkarte „Phase“ in der Spark-Benutzeroberfläche die Seite „Event Timeline“ an. Im folgenden Screenshot sehen Sie eine ungleichmäßige Verteilung der Aufgaben. Aufgaben, die ungleichmäßig verteilt sind oder deren Ausführung zu lange dauert, können auf Datenverzerrungen hinweisen.




- Eine weitere wichtige Seite ist Summary Metrics, auf der Statistiken für Spark-Aufgaben angezeigt werden. Der folgende Screenshot zeigt Metriken mit Perzentilen für Dauer, GC-Zeit, Datenverlust (Speicher), Datenverlust (Festplatte) usw.



Summary Metrics for 5 Completed Tasks

| Metric | Min | 25th percentile | Median | 75th percentile | Max |
|---|---|---|---|---|---|
| Duration | 9 s | 10 s | 11 s | 13 s | 6.4 min |
| GC Time | 0.0 ms | 0.2 s | 0.3 s | 0.4 s | 1 s |
| Spill (memory) | 0.0 B | 0.0 B | 0.0 B | 0.0 B | 16.7 GiB |
| Spill (disk) | 0.0 B | 0.0 B | 0.0 B | 0.0 B | 10.2 GiB |
| Output Size / Records | 8.3 MiB / 12651 | 9.4 MiB / 21462 | 36.1 MiB / 63860 | 92.9 MiB / 258057 | 10.1 GiB / 20370130 |
| Shuffle Read Size / Records | 9.8 MiB / 12651 | 11.7 MiB / 21462 | 43.4 MiB / 63860 | 122.6 MiB / 258057 | 11.8 GiB / 20370130 |

  Wenn die Aufgaben gleichmäßig verteilt sind, werden Sie ähnliche Zahlen in allen Perzentilen sehen. Wenn die Daten verzerrt sind, werden Sie in jedem Perzentil stark verzerrte Werte sehen. In diesem Beispiel beträgt die Aufgabendauer in Min., 25. Perzentil, Median und 75. Perzentil weniger als 13 Sekunden. Die Max-Aufgabe verarbeitete zwar 100-mal mehr Daten als das 75. Perzentil, ihre Dauer von 6,4 Minuten ist jedoch etwa 30-mal länger. Das bedeutet, dass mindestens eine Aufgabe (oder bis zu 25 Prozent der Aufgaben) viel länger dauerte als die übrigen Aufgaben.

Wenn Sie Datenverzerrungen feststellen, versuchen Sie Folgendes:

- Wenn Sie AWS Glue 3.0 verwenden, aktivieren Sie die adaptive Abfrageausführung durch Einstellung`spark.sql.adaptive.enabled=true`. Adaptive Abfrageausführung ist in AWS Glue 4.0 standardmäßig aktiviert.

  Sie können Adaptive Query Execution auch für Datenverzerrungen verwenden, die durch Verknüpfungen entstehen, indem Sie die folgenden zugehörigen Parameter festlegen:

  - `spark.sql.adaptive.skewJoin.skewedPartitionFactor`
  - `spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes`

- `spark.sql.adaptive.advisoryPartitionSizeInBytes=128m (128 mebibytes or larger should be good)`
- `spark.sql.adaptive.coalescePartitions.enabled=true (when you want to coalesce partitions)`

Weitere Informationen finden Sie in der Apache Spark-Dokumentation.

- Verwenden Sie Schlüssel mit einem großen Wertebereich für die Join-Schlüssel. Bei einem Shuffle-Join werden Partitionen für jeden Hashwert eines Schlüssels bestimmt. Wenn die Kardinalität eines Join-Schlüssels zu niedrig ist, ist es wahrscheinlicher, dass die Hash-Funktion Ihre Daten schlecht auf Partitionen verteilt. Wenn Ihre Anwendung und Geschäftslogik dies unterstützen, sollten Sie daher die Verwendung eines Schlüssels mit höherer Kardinalität oder eines zusammengesetzten Schlüssels in Betracht ziehen.

```
# Use Single Primary Key
df_joined = df1_select.join(df2_select, ["primary_key"])

# Use Composite Key
df_joined = df1_select.join(df2_select, ["primary_key","secondary_key"])
```

## Verwenden Sie den Cache

Wenn Sie repetitiv verwenden DataFrames, vermeiden Sie zusätzliches Mischen oder Rechnen, indem `df.cache()` Sie die Berechnungsergebnisse im Speicher der einzelnen Spark-Executoren und auf der Festplatte zwischenspeichern oder zwischenspeichern. `df.persist()` Spark unterstützt auch die persistente Speicherung RDDs auf der Festplatte oder die Replikation über mehrere Knoten (Speicherebene).

Sie können das beispielsweise beibehalten, indem Sie hinzufügen. DataFrames `df.persist()` Wenn der Cache nicht mehr benötigt wird, können Sie ihn verwenden, `unpersist` um die zwischengespeicherten Daten zu löschen.

```
df = spark.read.parquet("s3://<Bucket>/parquet/product_category=Books/")
df_high_rate = df.filter(col("star_rating")>=4.0)
df_high_rate.persist()

df_joined1 = df_high_rate.join(<Table1>, ["key"])
```

```
df_joined2 = df_high_rate.join(<Table2>, ["key"])
df_joined3 = df_high_rate.join(<Table3>, ["key"])
...
df_high_rate.unpersist()
```

### Entfernen Sie nicht benötigte Spark-Aktionen

Vermeiden Sie es, unnötige Aktionen wie `countshow`, oder `collect` auszuführen. Wie im Abschnitt Die wichtigsten Themen in Apache Spark beschrieben, ist Spark faul. Jede Transformation RDD kann jedes Mal neu berechnet werden, wenn Sie eine Aktion darauf ausführen. Wenn Sie viele Spark-Aktionen verwenden, werden für jede Aktion mehrere Quellzugriffe, Aufgabenberechnungen und Zufallsläufe aufgerufen.

Wenn Sie andere Aktionen in Ihrer kommerziellen Umgebung nicht benötigen`collect()`, sollten Sie erwägen, sie zu entfernen.

> **Note**
>
> Vermeiden Sie es so weit wie möglich, Spark `collect()` in kommerziellen Umgebungen zu verwenden. Die `collect()` Aktion gibt alle Ergebnisse einer Berechnung im Spark-Executor an den Spark-Treiber zurück, was dazu führen kann, dass der Spark-Treiber einen OOM Fehler zurückgibt. Um einen OOM Fehler zu vermeiden, legt Spark `spark.driver.maxResultSize = 1GB` standardmäßig fest, was die maximale Datengröße, die an den Spark-Treiber zurückgegeben wird, auf 1 GB begrenzt.

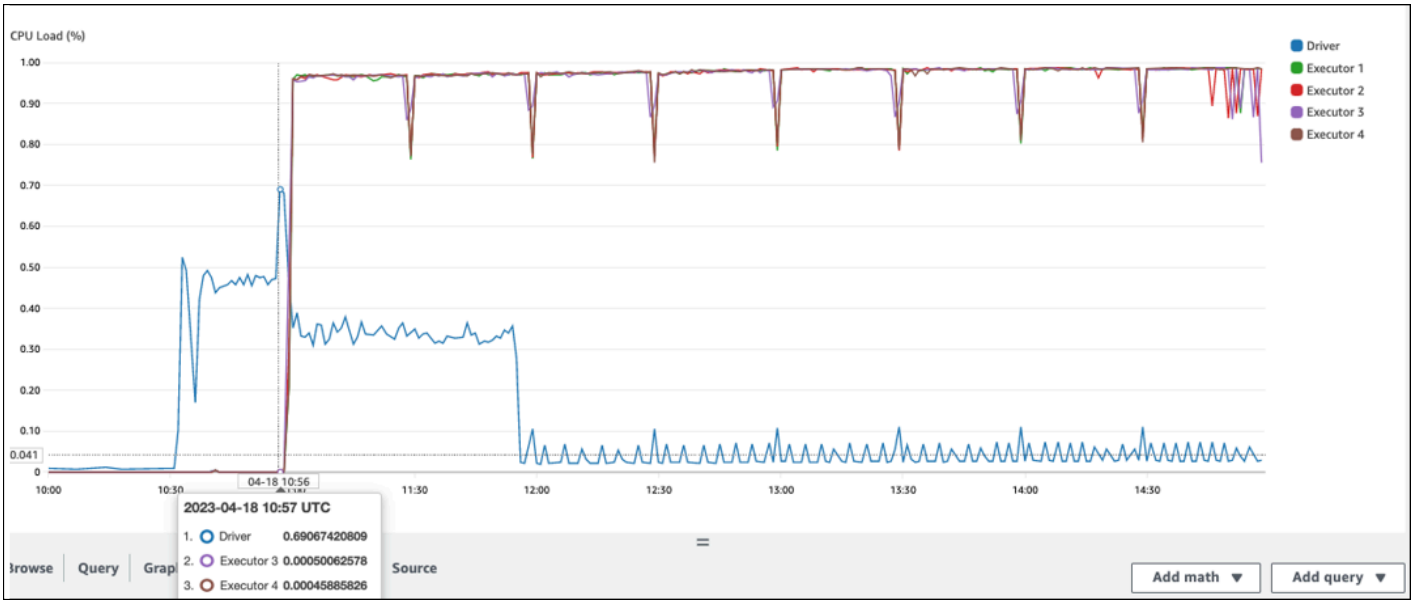
## Minimieren Sie den Planungsaufwand

Wie in den wichtigsten Themen in Apache Spark besprochen, generiert der Spark-Treiber den Ausführungsplan. Auf der Grundlage dieses Plans werden dem Spark-Executor Aufgaben für die verteilte Verarbeitung zugewiesen. Der Spark-Treiber kann jedoch zu einem Engpass werden, wenn es eine große Anzahl kleiner Dateien gibt oder wenn der eine große Anzahl von AWS Glue Data Catalog Partitionen enthält. Analysieren Sie die folgenden Kennzahlen, um einen hohen Planungsaufwand zu ermitteln.

### CloudWatch Metriken

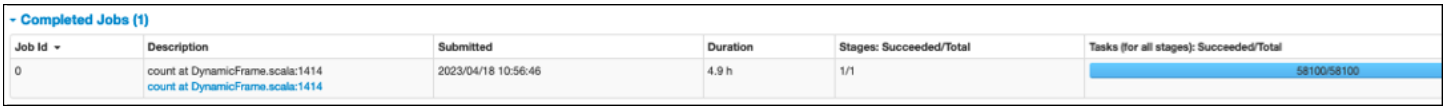
Überprüfen Sie die CPUAuslastung und die Speicherauslastung für die folgenden Situationen:

- Die CPUAuslastung des Spark-Treibers und die Speicherauslastung werden als hoch aufgezeichnet. Normalerweise verarbeitet der Spark-Treiber Ihre Daten nicht, sodass die CPU Last- und Speicherauslastung nicht ansteigt. Wenn die Amazon S3 S3-Datenquelle jedoch zu viele kleine Dateien enthält, kann das Auflisten aller S3-Objekte und das Verwalten einer großen Anzahl von Aufgaben zu einer hohen Ressourcenauslastung führen.
- Es besteht eine lange Pause, bis die Verarbeitung in Spark Executor beginnt. Im folgenden Beispiel-Screenshot ist die CPU Last des Spark-Executors bis 10:57 Uhr zu niedrig, obwohl der Job um 10:00 Uhr gestartet wurde. AWS Glue Dies deutet darauf hin, dass der Spark-Treiber möglicherweise lange braucht, um einen Ausführungsplan zu generieren. In diesem Beispiel dauert das Abrufen der großen Anzahl von Partitionen im Datenkatalog und das Auflisten der großen Anzahl kleiner Dateien im Spark-Treiber sehr lange.




## Spark-Benutzeroberfläche

Auf der Registerkarte Job in der Spark-Benutzeroberfläche können Sie die Zeit für die Einreichung sehen. Im folgenden Beispiel hat der Spark-Treiber Job0 um 10:56:46 Uhr gestartet, obwohl der Job um 10:00:00 Uhr gestartet wurde. AWS Glue



Sie können auch die Aufgaben (für alle Phasen): Erfolgreich/Gesamtzeit auf der Registerkarte Job sehen. In diesem Fall wird die Anzahl der Aufgaben als aufgezeichnet. `58100` Wie im Abschnitt Amazon S3 auf der Seite Aufgaben parallelisieren erklärt, entspricht die Anzahl der Aufgaben in etwa der Anzahl der S3-Objekte. Das bedeutet, dass es in Amazon S3 etwa 58.100 Objekte gibt.

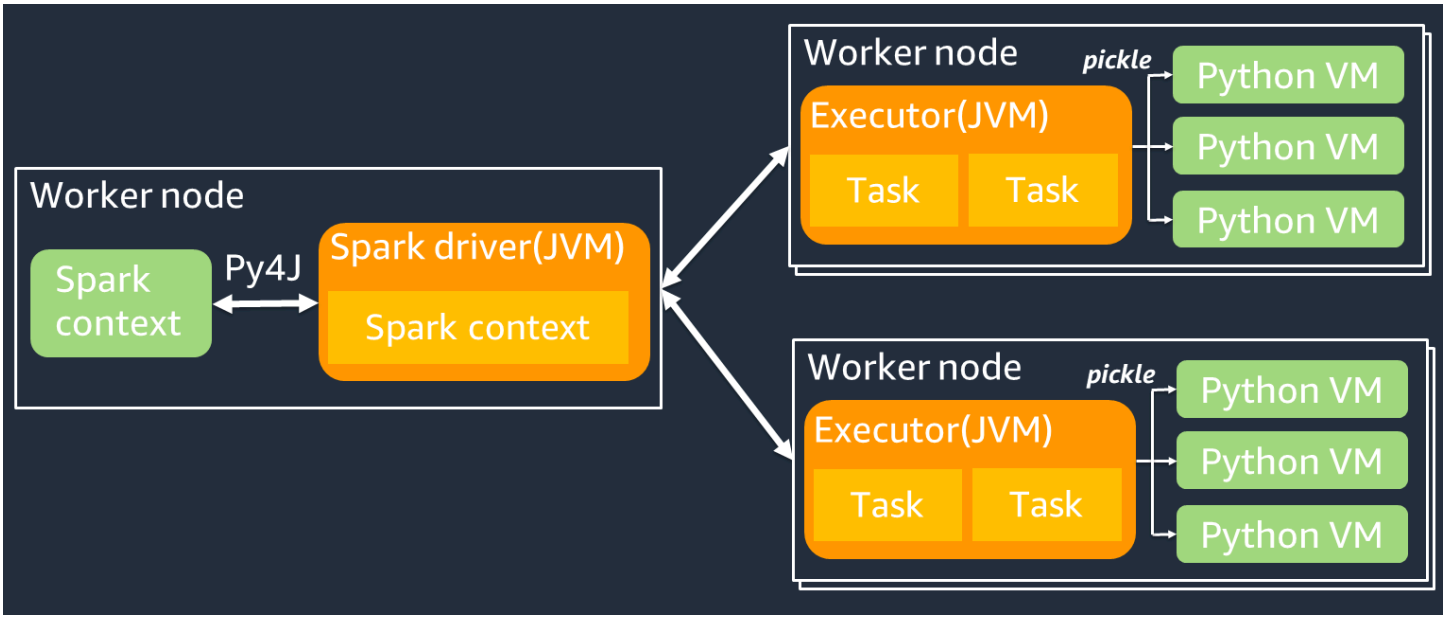
Weitere Informationen zu diesem Job und zum Zeitplan finden Sie auf der Registerkarte Phase. Wenn Sie einen Engpass beim Spark-Treiber feststellen, sollten Sie die folgenden Lösungen in Betracht ziehen:

- Wenn Amazon S3 zu viele Dateien enthält, beachten Sie die Hinweise zur übermäßigen Parallelität im Abschnitt Zu viele Partitionen auf der Seite Aufgaben parallelisieren.
- Wenn Amazon S3 über zu viele Partitionen verfügt, beachten Sie die Hinweise zur übermäßigen Partitionierung im Abschnitt Zu viele Amazon S3 S3-Partitionen auf der Seite Menge der gescannten Daten reduzieren. Aktivieren Sie AWS Glue Partitionsindizes, wenn es viele Partitionen gibt, um die Latenz beim Abrufen von Partitionsmetadaten aus dem Datenkatalog zu reduzieren. Weitere Informationen finden Sie unter Verbessern der Abfrageleistung mithilfe von AWS Glue Partitionsindizes.
- Wenn zu JDBC viele Partitionen vorhanden sind, verringern Sie den `hashpartition` Wert.
- Wenn DynamoDB zu viele Partitionen hat, verringern Sie den `dynamodb.splits` Wert.
- Wenn Streaming-Jobs zu viele Partitionen haben, verringern Sie die Anzahl der Shards.

## Optimieren Sie benutzerdefinierte Funktionen

Benutzerdefinierte Funktionen (UDFs) und `RDD.map` in beeinträchtigen die Leistung PySpark häufig erheblich. Dies liegt an dem Aufwand, der erforderlich ist, um Ihren Python-Code in der Scala-Implementierung, die Spark zugrunde liegt, korrekt darzustellen.

Das folgende Diagramm zeigt die Architektur von PySpark Jobs. Wenn Sie verwenden PySpark, verwendet der Spark-Treiber die Py4j-Bibliothek, um Java-Methoden von Python aus aufzurufen. Beim Aufrufen von Spark SQL oder DataFrame integrierten Funktionen gibt es kaum Leistungsunterschiede zwischen Python und Scala, da die Funktionen auf jedem Executor JVM mit einem optimierten Ausführungsplan ausgeführt werden.




Wenn Sie Ihre eigene Python-Logik verwenden, z. B. using`map/ mapPartitions/ udf`, wird die Aufgabe in einer Python-Laufzeitumgebung ausgeführt. Die Verwaltung von zwei Umgebungen verursacht Gemeinkosten. Darüber hinaus müssen Ihre Daten im Speicher transformiert werden, damit sie von den integrierten Funktionen der JVM Laufzeitumgebung verwendet werden können. Pickle ist ein Serialisierungsformat, das standardmäßig für den Austausch zwischen der JVM und der Python-Laufzeit verwendet wird. Die Kosten für diese Serialisierung und Deserialisierung sind jedoch sehr hoch, sodass in Java oder Scala UDFs geschriebene Versionen schneller sind als in Python. UDFs

Beachten Sie Folgendes, um den Aufwand für Serialisierung und Deserialisierung zu vermeiden: PySpark

- Verwenden Sie die integrierten SQL Spark-Funktionen — Erwägen Sie, Ihre eigene Funktion UDF oder Map-Funktion durch Spark SQL oder DataFrame integrierte Funktionen zu ersetzen. Bei der Ausführung von Spark SQL oder DataFrame integrierten Funktionen gibt es kaum Leistungsunterschiede zwischen Python und Scala, da die Aufgaben von jedem Executor ausgeführt werden. JVM
- Implementieren Sie UDFs in Scala oder Java — Erwägen Sie die Verwendung von aUDF, die in Java oder Scala geschrieben ist, da sie auf dem ausgeführt werden. JVM
- Verwenden Sie Apache Arrow-based UDFs für vektorisierte Workloads — Erwägen Sie die Verwendung von Arrow-based. UDFs Diese Funktion wird auch als Vectorized (Pandas) bezeichnet. UDF UDF Apache Arrow ist ein sprachunabhängiges In-Memory-Datenformat, mit dem

Daten effizient AWS Glue zwischen Python-Prozessen übertragen werden können. JVM Dies ist derzeit für Python-Benutzer, die mit Pandas oder NumPy Daten arbeiten, am vorteilhaftesten.

Arrow ist ein spaltenförmiges (vektorisiertes) Format. Die Verwendung erfolgt nicht automatisch und erfordert möglicherweise einige geringfügige Änderungen an der Konfiguration oder am Code, um alle Vorteile zu nutzen und die Kompatibilität sicherzustellen. Weitere Informationen und Einschränkungen finden Sie unter Apache Arrow unter PySpark.

Im folgenden Beispiel wird eine einfache inkrementelle Version UDF in Standard-Python, als Vectorized UDF und in Spark verglichen. SQL

## Standardpython UDF

Die Beispielzeit beträgt 3,20 (Sekunden).

Beispiel-Code

```
# DataSet
df = spark.range(10000000).selectExpr("id AS a","id AS b")

# UDF Example
def plus(a,b):
    return a+b
spark.udf.register("plus",plus)

df.selectExpr("count(plus(a,b))").collect()
```

Ausführungsplan

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[], functions=[count(pythonUDF0#124)])
+- Exchange SinglePartition, ENSURE_REQUIREMENTS, [id=#580]
+- HashAggregate(keys=[], functions=[partial_count(pythonUDF0#124)])
+- Project [pythonUDF0#124]
+- BatchEvalPython [plus(a#116L, b#117L)], [pythonUDF0#124]
+- Project [id#114L AS a#116L, id#114L AS b#117L]
+- Range (0, 10000000, step=1, splits=16)
```

## Vektorisiert UDF

Die Beispielzeit beträgt 0,59 (Sekunden).

Vectorized UDF ist fünfmal schneller als das vorherige Beispiel. UDF Wenn Sie `Physical Plan` das überprüfen, können Sie sehen`ArrowEvalPython`, was zeigt, dass diese Anwendung von Apache Arrow vektorisiert wurde. Um Vectorized zu aktivierenUDF, müssen Sie in Ihrem Code Folgendes angeben`spark.sql.execution.arrow.pyspark.enabled = true`.

Beispiel-Code

```
# Vectorized UDF
from pyspark.sql.types import LongType
from pyspark.sql.functions import count, pandas_udf

# Enable Apache Arrow Support
spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")

# DataSet
df = spark.range(10000000).selectExpr("id AS a","id AS b")

# Annotate pandas_udf to use Vectorized UDF
@pandas_udf(LongType())
def pandas_plus(a,b):
    return a+b
spark.udf.register("pandas_plus",pandas_plus)

df.selectExpr("count(pandas_plus(a,b))").collect()
```

Ausführungsplan

```
== Physical Plan ==
AdaptiveSparkPlan isFinalPlan=false
+- HashAggregate(keys=[], functions=[count(pythonUDF0#1082L)],
 output=[count(pandas_plus(a, b))#1080L])
+- Exchange SinglePartition, ENSURE_REQUIREMENTS, [id=#5985]
+- HashAggregate(keys=[], functions=[partial_count(pythonUDF0#1082L)],
 output=[count#1084L])
+- Project [pythonUDF0#1082L]
+- ArrowEvalPython [pandas_plus(a#1074L, b#1075L)], [pythonUDF0#1082L], 200
+- Project [id#1072L AS a#1074L, id#1072L AS b#1075L]
```

```
+- Range (0, 10000000, step=1, splits=16)
```

## Funke SQL

Die Beispielzeit beträgt 0,087 (Sekunden).

Spark SQL ist viel schneller als VectorizedUDF, da die Aufgaben auf jedem Executor JVM ohne Python-Laufzeit ausgeführt werden. Wenn Sie Ihre UDF durch eine integrierte Funktion ersetzen können, empfehlen wir, dies zu tun.

Beispiel-Code

```
df.createOrReplaceTempView("test")
spark.sql("select count(a+b) from test").collect()
```

## Pandas für Big Data verwenden

Wenn Sie bereits mit Pandas vertraut sind und Spark für Big Data verwenden möchten, können Sie die Pandas API auf Spark verwenden. AWS Glue 4.0 und höher unterstützen es. Um loszulegen, können Sie das offizielle Notizbuch Quickstart: Pandas API on Spark verwenden. Weitere Informationen finden Sie in der PySpark Dokumentation.

# Ressourcen

- AWS Glue
- Leistungsoptimierung (Spark-SQL-Leitfaden)
- AWS Glue Workshop zur Optimierung

# Dokumentverlauf

In der folgenden Tabelle werden wichtige Änderungen in diesem Leitfaden beschrieben. Um Benachrichtigungen über zukünftige Aktualisierungen zu erhalten, können Sie einen RSS-Feed abonnieren.

| Änderung | Beschreibung | Datum |
| --- | --- | --- |
| Erste Veröffentlichung | — | 2. Januar 2024 |

# AWS Glossar zu präskriptiven Leitlinien

Im Folgenden finden Sie häufig verwendete Begriffe in Strategien, Leitfäden und Mustern von AWS Prescriptive Guidance. Um Einträge vorzuschlagen, verwenden Sie bitte den Link Feedback geben am Ende des Glossars.

## Zahlen

7 Rs

Sieben gängige Migrationsstrategien für die Verlagerung von Anwendungen in die Cloud. Diese Strategien bauen auf den 5 Rs auf, die Gartner 2011 identifiziert hat, und bestehen aus folgenden Elementen:

- Faktorwechsel/Architekturwechsel – Verschieben Sie eine Anwendung und ändern Sie ihre Architektur, indem Sie alle Vorteile cloudnativer Feature nutzen, um Agilität, Leistung und Skalierbarkeit zu verbessern. Dies beinhaltet in der Regel die Portierung des Betriebssystems und der Datenbank. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank auf die Amazon Aurora PostgreSQL-kompatible Edition.
- Plattformwechsel (Lift and Reshape) – Verschieben Sie eine Anwendung in die Cloud und führen Sie ein gewisses Maß an Optimierung ein, um die Cloud-Funktionen zu nutzen. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank zu Amazon Relational Database Service (Amazon RDS) für Oracle in der. AWS Cloud
- Neukauf (Drop and Shop) – Wechseln Sie zu einem anderen Produkt, indem Sie typischerweise von einer herkömmlichen Lizenz zu einem SaaS-Modell wechseln. Beispiel: Migrieren Sie Ihr CRM-System (Customer Relationship Management) zu Salesforce.com.
- Hostwechsel (Lift and Shift) – Verschieben Sie eine Anwendung in die Cloud, ohne Änderungen vorzunehmen, um die Cloud-Funktionen zu nutzen. Beispiel: Migrieren Sie Ihre lokale Oracle-Datenbank zu Oracle auf einer EC2-Instanz in der. AWS Cloud
- Verschieben (Lift and Shift auf Hypervisor-Ebene) – Verlagern Sie die Infrastruktur in die Cloud, ohne neue Hardware kaufen, Anwendungen umschreiben oder Ihre bestehenden Abläufe ändern zu müssen. Sie migrieren Server von einer lokalen Plattform zu einem Cloud-Dienst für dieselbe Plattform. Beispiel: Migrieren Sie eine Microsoft Hyper-V Anwendung zu AWS.
- Beibehaltung (Wiederaufgreifen) – Bewahren Sie Anwendungen in Ihrer Quellumgebung auf. Dazu können Anwendungen gehören, die einen umfangreichen Faktorwechsel erfordern und

die Sie auf einen späteren Zeitpunkt verschieben möchten, sowie ältere Anwendungen, die Sie beibehalten möchten, da es keine geschäftliche Rechtfertigung für ihre Migration gibt.

- Außerbetriebnahme – Dekommissionierung oder Entfernung von Anwendungen, die in Ihrer Quellumgebung nicht mehr benötigt werden.

# A

ABAC

Siehe attributbasierte Zugriffskontrolle.

abstrahierte Dienste

Weitere Informationen finden Sie unter Managed Services.

ACID

Siehe Atomarität, Konsistenz, Isolierung und Haltbarkeit.

Aktiv-Aktiv-Migration

Eine Datenbankmigrationsmethode, bei der die Quell- und Zieldatenbanken synchron gehalten werden (mithilfe eines bidirektionalen Replikationstools oder dualer Schreibvorgänge) und beide Datenbanken Transaktionen von miteinander verbundenen Anwendungen während der Migration verarbeiten. Diese Methode unterstützt die Migration in kleinen, kontrollierten Batches, anstatt einen einmaligen Cutover zu erfordern. Es ist flexibler, erfordert aber mehr Arbeit als eine aktiv-passive Migration.

Aktiv-Passiv-Migration

Eine Datenbankmigrationsmethode, bei der die Quell- und Zieldatenbanken synchron gehalten werden, aber nur die Quelldatenbank Transaktionen von verbindenden Anwendungen verarbeitet, während Daten in die Zieldatenbank repliziert werden. Die Zieldatenbank akzeptiert während der Migration keine Transaktionen.

Aggregatfunktion

Eine SQL-Funktion, die mit einer Gruppe von Zeilen arbeitet und einen einzelnen Rückgabewert für die Gruppe berechnet. Beispiele für Aggregatfunktionen sind `SUM` und`MAX`.

AI

Siehe künstliche Intelligenz.

AIOps

Siehe Operationen mit künstlicher Intelligenz.

Anonymisierung

Der Prozess des dauerhaften Löschens personenbezogener Daten in einem Datensatz. Anonymisierung kann zum Schutz der Privatsphäre beitragen. Anonymisierte Daten gelten nicht mehr als personenbezogene Daten.

Anti-Muster

Eine häufig verwendete Lösung für ein wiederkehrendes Problem, bei dem die Lösung kontraproduktiv, ineffektiv oder weniger wirksam als eine Alternative ist.

Anwendungssteuerung

Ein Sicherheitsansatz, bei dem nur zugelassene Anwendungen verwendet werden können, um ein System vor Schadsoftware zu schützen.

Anwendungsportfolio

Eine Sammlung detaillierter Informationen zu jeder Anwendung, die von einer Organisation verwendet wird, einschließlich der Kosten für die Erstellung und Wartung der Anwendung und ihres Geschäftswerts. Diese Informationen sind entscheidend für den Prozess der Portfoliofindung und -analyse und hilft bei der Identifizierung und Priorisierung der Anwendungen, die migriert, modernisiert und optimiert werden sollen.

künstliche Intelligenz (KI)

Das Gebiet der Datenverarbeitungswissenschaft, das sich der Nutzung von Computertechnologien zur Ausführung kognitiver Funktionen widmet, die typischerweise mit Menschen in Verbindung gebracht werden, wie Lernen, Problemlösen und Erkennen von Mustern. Weitere Informationen finden Sie unter Was ist künstliche Intelligenz?

Operationen mit künstlicher Intelligenz (AIOps)

Der Prozess des Einsatzes von Techniken des Machine Learning zur Lösung betrieblicher Probleme, zur Reduzierung betrieblicher Zwischenfälle und menschlicher Eingriffe sowie zur Steigerung der Servicequalität. Weitere Informationen zur Verwendung von AIOps in der AWS -Migrationsstrategie finden Sie im Leitfaden zur Betriebsintegration.

Asymmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der ein Schlüsselpaar, einen öffentlichen Schlüssel für die Verschlüsselung und einen privaten Schlüssel für die Entschlüsselung verwendet. Sie können den öffentlichen Schlüssel teilen, da er nicht für die Entschlüsselung verwendet wird. Der Zugriff auf den privaten Schlüssel sollte jedoch stark eingeschränkt sein.

Atomizität, Konsistenz, Isolierung, Haltbarkeit (ACID)

Eine Reihe von Softwareeigenschaften, die die Datenvalidität und betriebliche Zuverlässigkeit einer Datenbank auch bei Fehlern, Stromausfällen oder anderen Problemen gewährleisten.

Attributbasierte Zugriffskontrolle (ABAC)

Die Praxis, detaillierte Berechtigungen auf der Grundlage von Benutzerattributen wie Abteilung, Aufgabenrolle und Teamname zu erstellen. Weitere Informationen finden Sie unter ABAC AWS in der AWS Identity and Access Management (IAM-) Dokumentation.

autoritative Datenquelle

Ein Ort, an dem Sie die primäre Version der Daten speichern, die als die zuverlässigste Informationsquelle angesehen wird. Sie können Daten aus der maßgeblichen Datenquelle an andere Speicherorte kopieren, um die Daten zu verarbeiten oder zu ändern, z. B. zu anonymisieren, zu redigieren oder zu pseudonymisieren.

Availability Zone

Ein bestimmter Standort innerhalb einer AWS-Region , der vor Ausfällen in anderen Availability Zones geschützt ist und kostengünstige Netzwerkkonnektivität mit niedriger Latenz zu anderen Availability Zones in derselben Region bietet.

AWS Framework für die Cloud-Einführung (AWS CAF)

Ein Framework mit Richtlinien und bewährten Verfahren, das Unternehmen bei der Entwicklung eines effizienten und effektiven Plans für den erfolgreichen Umstieg auf die Cloud unterstützt. AWS AWS CAF unterteilt die Leitlinien in sechs Schwerpunktbereiche, die als Perspektiven bezeichnet werden: Unternehmen, Mitarbeiter, Unternehmensführung, Plattform, Sicherheit und Betrieb. Die Perspektiven Geschäft, Mitarbeiter und Unternehmensführung konzentrieren sich auf Geschäftskompetenzen und -prozesse, während sich die Perspektiven Plattform, Sicherheit und Betriebsabläufe auf technische Fähigkeiten und Prozesse konzentrieren. Die Personalperspektive zielt beispielsweise auf Stakeholder ab, die sich mit Personalwesen (HR), Personalfunktionen und Personalmanagement befassen. Aus dieser Perspektive bietet AWS CAF Leitlinien für Personalentwicklung, Schulung und Kommunikation, um das Unternehmen auf eine erfolgreiche

Cloud-Einführung vorzubereiten. Weitere Informationen finden Sie auf der AWS -CAF-Webseite und dem AWS -CAF-Whitepaper.

AWS Workload-Qualifizierungsrahmen (AWS WQF)

Ein Tool, das Workloads bei der Datenbankmigration bewertet, Migrationsstrategien empfiehlt und Arbeitsschätzungen bereitstellt. AWS WQF ist in () enthalten. AWS Schema Conversion Tool AWS SCT Es analysiert Datenbankschemas und Codeobjekte, Anwendungscode, Abhängigkeiten und Leistungsmerkmale und stellt Bewertungsberichte bereit.

# B

schlechter Bot

Ein Bot, der Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen soll.

BCP

Siehe Planung der Geschäftskontinuität.

Verhaltensdiagramm

Eine einheitliche, interaktive Ansicht des Ressourcenverhaltens und der Interaktionen im Laufe der Zeit. Sie können ein Verhaltensdiagramm mit Amazon Detective verwenden, um fehlgeschlagene Anmeldeversuche, verdächtige API-Aufrufe und ähnliche Vorgänge zu untersuchen. Weitere Informationen finden Sie unter Daten in einem Verhaltensdiagramm in der Detective-Dokumentation.

Big-Endian-System

Ein System, welches das höchstwertige Byte zuerst speichert. Siehe auch Endianness.

Binäre Klassifikation

Ein Prozess, der ein binäres Ergebnis vorhersagt (eine von zwei möglichen Klassen). Beispielsweise könnte Ihr ML-Modell möglicherweise Probleme wie „Handelt es sich bei dieser E-Mail um Spam oder nicht?“ vorhersagen müssen oder „Ist dieses Produkt ein Buch oder ein Auto?“

Bloom-Filter

Eine probabilistische, speichereffiziente Datenstruktur, mit der getestet wird, ob ein Element Teil einer Menge ist.

Blau/Grün-Bereitstellung

Eine Bereitstellungsstrategie, bei der Sie zwei separate, aber identische Umgebungen erstellen. Sie führen die aktuelle Anwendungsversion in einer Umgebung (blau) und die neue Anwendungsversion in der anderen Umgebung (grün) aus. Mit dieser Strategie können Sie schnell und mit minimalen Auswirkungen ein Rollback durchführen.

Bot

Eine Softwareanwendung, die automatisierte Aufgaben über das Internet ausführt und menschliche Aktivitäten oder Interaktionen simuliert. Manche Bots sind nützlich oder nützlich, wie z. B. Webcrawler, die Informationen im Internet indexieren. Einige andere Bots, die als bösartige Bots bezeichnet werden, sollen Einzelpersonen oder Organisationen stören oder ihnen Schaden zufügen.

Botnetz

Netzwerke von Bots, die mit Malware infiziert sind und unter der Kontrolle einer einzigen Partei stehen, die als Bot-Herder oder Bot-Operator bezeichnet wird. Botnetze sind der bekannteste Mechanismus zur Skalierung von Bots und ihrer Wirkung.

branch

Ein containerisierter Bereich eines Code-Repositorys. Der erste Zweig, der in einem Repository erstellt wurde, ist der Hauptzweig. Sie können einen neuen Zweig aus einem vorhandenen Zweig erstellen und dann Feature entwickeln oder Fehler in dem neuen Zweig beheben. Ein Zweig, den Sie erstellen, um ein Feature zu erstellen, wird allgemein als Feature-Zweig bezeichnet. Wenn das Feature zur Veröffentlichung bereit ist, führen Sie den Feature-Zweig wieder mit dem Hauptzweig zusammen. Weitere Informationen finden Sie unter Über Branches (GitHub Dokumentation).

Zugang durch Glasbruch

Unter außergewöhnlichen Umständen und im Rahmen eines genehmigten Verfahrens ist dies eine schnelle Methode für einen Benutzer, auf einen Bereich zuzugreifen AWS-Konto , für den er normalerweise keine Zugriffsrechte besitzt. Weitere Informationen finden Sie unter dem Indikator Implementation break-glass procedures in den AWS Well-Architected-Leitlinien.

Brownfield-Strategie

Die bestehende Infrastruktur in Ihrer Umgebung. Wenn Sie eine Brownfield-Strategie für eine Systemarchitektur anwenden, richten Sie sich bei der Gestaltung der Architektur nach den

Einschränkungen der aktuellen Systeme und Infrastruktur. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und Greenfield-Strategien mischen.

Puffer-Cache

Der Speicherbereich, in dem die am häufigsten abgerufenen Daten gespeichert werden.

Geschäftsfähigkeit

Was ein Unternehmen tut, um Wert zu generieren (z. B. Vertrieb, Kundenservice oder Marketing). Microservices-Architekturen und Entwicklungsentscheidungen können von den Geschäftskapazitäten beeinflusst werden. Weitere Informationen finden Sie im Abschnitt Organisiert nach Geschäftskapazitäten des Whitepapers Ausführen von containerisierten Microservices in AWS.

Planung der Geschäftskontinuität (BCP)

Ein Plan, der die potenziellen Auswirkungen eines störenden Ereignisses, wie z. B. einer groß angelegten Migration, auf den Betrieb berücksichtigt und es einem Unternehmen ermöglicht, den Betrieb schnell wieder aufzunehmen.

# C

CAF

Weitere Informationen finden Sie unter Framework für die AWS Cloud-Einführung.

Bereitstellung auf Kanaren

Die langsame und schrittweise Veröffentlichung einer Version für Endbenutzer. Wenn Sie sich sicher sind, stellen Sie die neue Version bereit und ersetzen die aktuelle Version vollständig.

CCoE

Weitere Informationen finden Sie im Cloud Center of Excellence.

CDC

Siehe Erfassung von Änderungsdaten.

Erfassung von Datenänderungen (CDC)

Der Prozess der Nachverfolgung von Änderungen an einer Datenquelle, z. B. einer Datenbanktabelle, und der Aufzeichnung von Metadaten zu der Änderung. Sie können CDC für

verschiedene Zwecke verwenden, z. B. für die Prüfung oder Replikation von Änderungen in einem Zielsystem, um die Synchronisation aufrechtzuerhalten.

Chaos-Technik

Absichtliches Einführen von Ausfällen oder Störungsereignissen, um die Widerstandsfähigkeit eines Systems zu testen. Sie können AWS Fault Injection Service (AWS FIS) verwenden, um Experimente durchzuführen, die Ihre AWS Workloads stress, und deren Reaktion zu bewerten.

CI/CD

Siehe Continuous Integration und Continuous Delivery.

Klassifizierung

Ein Kategorisierungsprozess, der bei der Erstellung von Vorhersagen hilft. ML-Modelle für Klassifikationsprobleme sagen einen diskreten Wert voraus. Diskrete Werte unterscheiden sich immer voneinander. Beispielsweise muss ein Modell möglicherweise auswerten, ob auf einem Bild ein Auto zu sehen ist oder nicht.

clientseitige Verschlüsselung

Lokale Verschlüsselung von Daten, bevor das Ziel sie AWS-Service empfängt.

Cloud-Kompetenzzentrum (CCoE)

Ein multidisziplinäres Team, das die Cloud-Einführung in der gesamten Organisation vorantreibt, einschließlich der Entwicklung bewährter Cloud-Methoden, der Mobilisierung von Ressourcen, der Festlegung von Migrationszeitplänen und der Begleitung der Organisation durch groß angelegte Transformationen. Weitere Informationen finden Sie in den CCoE-Beiträgen im AWS Cloud Enterprise Strategy Blog.

Cloud Computing

Die Cloud-Technologie, die typischerweise für die Ferndatenspeicherung und das IoT-Gerätemanagement verwendet wird. Cloud Computing ist häufig mit Edge-Computing-Technologie verbunden.

Cloud-Betriebsmodell

In einer IT-Organisation das Betriebsmodell, das zum Aufbau, zur Weiterentwicklung und Optimierung einer oder mehrerer Cloud-Umgebungen verwendet wird. Weitere Informationen finden Sie unter Aufbau Ihres Cloud-Betriebsmodells.

Phasen der Einführung der Cloud

Die vier Phasen, die Unternehmen bei der Migration in der Regel durchlaufen AWS Cloud:

- Projekt – Durchführung einiger Cloud-bezogener Projekte zu Machbarkeitsnachweisen und zu Lernzwecken
- Fundament – Grundlegende Investitionen tätigen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer Landing Zone, Definition eines CCoE, Einrichtung eines Betriebsmodells)
- Migration – Migrieren einzelner Anwendungen
- Neuentwicklung – Optimierung von Produkten und Services und Innovation in der Cloud

Diese Phasen wurden von Stephen Orban im Blogbeitrag The Journey Toward Cloud-First & the Stages of Adoption im AWS Cloud Enterprise Strategy-Blog definiert. Informationen darüber, wie sie mit der AWS Migrationsstrategie zusammenhängen, finden Sie im Leitfaden zur Vorbereitung der Migration.

CMDB

Siehe Datenbank für das Konfigurationsmanagement.

Code-Repository

Ein Ort, an dem Quellcode und andere Komponenten wie Dokumentation, Beispiele und Skripts gespeichert und im Rahmen von Versionskontrollprozessen aktualisiert werden. Zu den gängigen Cloud-Repositorys gehören GitHub oder AWS CodeCommit. Jede Version des Codes wird Zweig genannt. In einer Microservice-Struktur ist jedes Repository einer einzelnen Funktionalität gewidmet. Eine einzelne CI/CD-Pipeline kann mehrere Repositorien verwenden.

Kalter Cache

Ein Puffer-Cache, der leer oder nicht gut gefüllt ist oder veraltete oder irrelevante Daten enthält. Dies beeinträchtigt die Leistung, da die Datenbank-Instance aus dem Hauptspeicher oder der Festplatte lesen muss, was langsamer ist als das Lesen aus dem Puffercache.

Kalte Daten

Daten, auf die selten zugegriffen wird und die in der Regel historisch sind. Bei der Abfrage dieser Art von Daten sind langsame Abfragen in der Regel akzeptabel. Durch die Verlagerung dieser Daten auf leistungsschwächere und kostengünstigere Speicherstufen oder -klassen können Kosten gesenkt werden.

Computer Vision (CV)

Ein Bereich der KI, der maschinelles Lernen nutzt, um Informationen aus visuellen Formaten wie digitalen Bildern und Videos zu analysieren und zu extrahieren. AWS Panorama Bietet beispielsweise Geräte an, die CV zu lokalen Kameranetzwerken hinzufügen, und Amazon SageMaker stellt Bildverarbeitungsalgorithmen für CV bereit.

Drift in der Konfiguration

Bei einer Arbeitslast eine Änderung der Konfiguration gegenüber dem erwarteten Zustand. Dies kann dazu führen, dass der Workload nicht mehr richtlinienkonform wird, und zwar in der Regel schrittweise und unbeabsichtigt.

Verwaltung der Datenbankkonfiguration (CMDB)

Ein Repository, das Informationen über eine Datenbank und ihre IT-Umgebung speichert und verwaltet, inklusive Hardware- und Softwarekomponenten und deren Konfigurationen. In der Regel verwenden Sie Daten aus einer CMDB in der Phase der Portfolioerkennung und -analyse der Migration.

Konformitätspaket

Eine Sammlung von AWS Config Regeln und Abhilfemaßnahmen, die Sie zusammenstellen können, um Ihre Konformitäts- und Sicherheitsprüfungen individuell anzupassen. Mithilfe einer YAML-Vorlage können Sie ein Conformance Pack als einzelne Entität in einer AWS-Konto AND-Region oder unternehmensweit bereitstellen. Weitere Informationen finden Sie in der Dokumentation unter Conformance Packs. AWS Config

Kontinuierliche Bereitstellung und kontinuierliche Integration (CI/CD)

Der Prozess der Automatisierung der Quell-, Build-, Test-, Staging- und Produktionsphasen des Softwareveröffentlichungsprozesses. CI/CD wird allgemein als Pipeline beschrieben. CI/CD kann Ihnen helfen, Prozesse zu automatisieren, die Produktivität zu steigern, die Codequalität zu verbessern und schneller zu liefern. Weitere Informationen finden Sie unter Vorteile der kontinuierlichen Auslieferung. CD kann auch für kontinuierliche Bereitstellung stehen. Weitere Informationen finden Sie unterKontinuierliche Auslieferung im Vergleich zu kontinuierlicher Bereitstellung.

CV

Siehe Computer Vision.

# D

Daten im Ruhezustand

Daten, die in Ihrem Netzwerk stationär sind, z. B. Daten, die sich im Speicher befinden.

Datenklassifizierung

Ein Prozess zur Identifizierung und Kategorisierung der Daten in Ihrem Netzwerk auf der Grundlage ihrer Kritikalität und Sensitivität. Sie ist eine wichtige Komponente jeder Strategie für das Management von Cybersecurity-Risiken, da sie Ihnen hilft, die geeigneten Schutz- und Aufbewahrungskontrollen für die Daten zu bestimmen. Die Datenklassifizierung ist ein Bestandteil der Sicherheitssäule im AWS Well-Architected Framework. Weitere Informationen finden Sie unter Datenklassifizierung.

Datendrift

Eine signifikante Abweichung zwischen den Produktionsdaten und den Daten, die zum Trainieren eines ML-Modells verwendet wurden, oder eine signifikante Änderung der Eingabedaten im Laufe der Zeit. Datendrift kann die Gesamtqualität, Genauigkeit und Fairness von ML-Modellvorhersagen beeinträchtigen.

Daten während der Übertragung

Daten, die sich aktiv durch Ihr Netzwerk bewegen, z. B. zwischen Netzwerkressourcen.

Datennetz

Ein architektonisches Framework, das verteilte, dezentrale Dateneigentum mit zentraler Verwaltung und Steuerung ermöglicht.

Datenminimierung

Das Prinzip, nur die Daten zu sammeln und zu verarbeiten, die unbedingt erforderlich sind. Durch Datenminimierung im AWS Cloud können Datenschutzrisiken, Kosten und der CO2-Fußabdruck Ihrer Analysen reduziert werden.

Datenperimeter

Eine Reihe präventiver Schutzmaßnahmen in Ihrer AWS Umgebung, die sicherstellen, dass nur vertrauenswürdige Identitäten auf vertrauenswürdige Ressourcen von erwarteten Netzwerken zugreifen. Weitere Informationen finden Sie unter Aufbau eines Datenperimeters auf. AWS

Vorverarbeitung der Daten

Rohdaten in ein Format umzuwandeln, das von Ihrem ML-Modell problemlos verarbeitet werden kann. Die Vorverarbeitung von Daten kann bedeuten, dass bestimmte Spalten oder Zeilen entfernt und fehlende, inkonsistente oder doppelte Werte behoben werden.

Herkunft der Daten

Der Prozess der Nachverfolgung des Ursprungs und der Geschichte von Daten während ihres gesamten Lebenszyklus, z. B. wie die Daten generiert, übertragen und gespeichert wurden.

betroffene Person

Eine Person, deren Daten gesammelt und verarbeitet werden.

Data Warehouse

Ein Datenverwaltungssystem, das Business Intelligence wie Analysen unterstützt. Data Warehouses enthalten in der Regel große Mengen an historischen Daten und werden in der Regel für Abfragen und Analysen verwendet.

Datenbankdefinitionssprache (DDL)

Anweisungen oder Befehle zum Erstellen oder Ändern der Struktur von Tabellen und Objekten in einer Datenbank.

Datenbankmanipulationssprache (DML)

Anweisungen oder Befehle zum Ändern (Einfügen, Aktualisieren und Löschen) von Informationen in einer Datenbank.

DDL

Siehe Datenbankdefinitionssprache.

Deep-Ensemble

Mehrere Deep-Learning-Modelle zur Vorhersage kombinieren. Sie können Deep-Ensembles verwenden, um eine genauere Vorhersage zu erhalten oder um die Unsicherheit von Vorhersagen abzuschätzen.

Deep Learning

Ein ML-Teilbereich, der mehrere Schichten künstlicher neuronaler Netzwerke verwendet, um die Zuordnung zwischen Eingabedaten und Zielvariablen von Interesse zu ermitteln.

defense-in-depth

Ein Ansatz zur Informationssicherheit, bei dem eine Reihe von Sicherheitsmechanismen und -kontrollen sorgfältig in einem Computernetzwerk verteilt werden, um die Vertraulichkeit, Integrität und Verfügbarkeit des Netzwerks und der darin enthaltenen Daten zu schützen. Wenn Sie diese Strategie anwenden AWS, fügen Sie mehrere Steuerelemente auf verschiedenen Ebenen der AWS Organizations Struktur hinzu, um die Ressourcen zu schützen. Ein defense-in-depth Ansatz könnte beispielsweise Multi-Faktor-Authentifizierung, Netzwerksegmentierung und Verschlüsselung kombinieren.

delegierter Administrator

In AWS Organizations kann ein kompatibler Dienst ein AWS Mitgliedskonto registrieren, um die Konten der Organisation und die Berechtigungen für diesen Dienst zu verwalten. Dieses Konto wird als delegierter Administrator für diesen Service bezeichnet. Weitere Informationen und eine Liste kompatibler Services finden Sie unter Services, die mit AWS Organizations funktionieren in der AWS Organizations -Dokumentation.

Bereitstellung

Der Prozess, bei dem eine Anwendung, neue Feature oder Codekorrekturen in der Zielumgebung verfügbar gemacht werden. Die Bereitstellung umfasst das Implementieren von Änderungen an einer Codebasis und das anschließende Erstellen und Ausführen dieser Codebasis in den Anwendungsumgebungen.

Entwicklungsumgebung

Siehe Umgebung.

Detektivische Kontrolle

Eine Sicherheitskontrolle, die darauf ausgelegt ist, ein Ereignis zu erkennen, zu protokollieren und zu warnen, nachdem ein Ereignis eingetreten ist. Diese Kontrollen stellen eine zweite Verteidigungslinie dar und warnen Sie vor Sicherheitsereignissen, bei denen die vorhandenen präventiven Kontrollen umgangen wurden. Weitere Informationen finden Sie unter Detektivische Kontrolle in Implementierung von Sicherheitskontrollen in AWS.

Abbildung des Wertstroms in der Entwicklung (DVSM)

Ein Prozess zur Identifizierung und Priorisierung von Einschränkungen, die sich negativ auf Geschwindigkeit und Qualität im Lebenszyklus der Softwareentwicklung auswirken. DVSM erweitert den Prozess der Wertstromanalyse, der ursprünglich für Lean-Manufacturing-Praktiken

konzipiert wurde. Es konzentriert sich auf die Schritte und Teams, die erforderlich sind, um durch den Softwareentwicklungsprozess Mehrwert zu schaffen und zu steigern.

digitaler Zwilling

Eine virtuelle Darstellung eines realen Systems, z. B. eines Gebäudes, einer Fabrik, einer Industrieanlage oder einer Produktionslinie. Digitale Zwillinge unterstützen vorausschauende Wartung, Fernüberwachung und Produktionsoptimierung.

Maßtabelle

In einem Sternschema eine kleinere Tabelle, die Datenattribute zu quantitativen Daten in einer Faktentabelle enthält. Bei Attributen von Dimensionstabellen handelt es sich in der Regel um Textfelder oder diskrete Zahlen, die sich wie Text verhalten. Diese Attribute werden häufig zum Einschränken von Abfragen, zum Filtern und zur Kennzeichnung von Ergebnismengen verwendet.

Katastrophe

Ein Ereignis, das verhindert, dass ein Workload oder ein System seine Geschäftsziele an seinem primären Einsatzort erfüllt. Diese Ereignisse können Naturkatastrophen, technische Ausfälle oder das Ergebnis menschlichen Handelns sein, z. B. unbeabsichtigte Fehlkonfigurationen oder Malware-Angriffe.

Notfallwiederherstellung (DR)

Die Strategie und der Prozess, die Sie zur Minimierung von Ausfallzeiten und Datenverlusten aufgrund einer Katastrophe anwenden. Weitere Informationen finden Sie unter Disaster Recovery von Workloads unter AWS: Wiederherstellung in der Cloud im AWS Well-Architected Framework.

DML

Siehe Sprache zur Datenbankmanipulation.

Domainorientiertes Design

Ein Ansatz zur Entwicklung eines komplexen Softwaresystems, bei dem seine Komponenten mit sich entwickelnden Domains oder Kerngeschäftszielen verknüpft werden, denen jede Komponente dient. Dieses Konzept wurde von Eric Evans in seinem Buch Domaingesteuertes Design: Bewältigen der Komplexität im Herzen der Software (Boston: Addison-Wesley Professional, 2003) vorgestellt. Informationen darüber, wie Sie domaingesteuertes Design mit dem Strangler-Fig-Muster verwenden können, finden Sie unter Schrittweises Modernisieren älterer Microsoft ASP.NET (ASMX)-Webservices mithilfe von Containern und Amazon API Gateway.

DR

Siehe Disaster Recovery.

Erkennung von Driften

Verfolgung von Abweichungen von einer Basiskonfiguration Sie können es beispielsweise verwenden, AWS CloudFormation um Abweichungen bei den Systemressourcen zu erkennen, oder Sie können AWS Control Tower damit Änderungen in Ihrer landing zone erkennen, die sich auf die Einhaltung von Governance-Anforderungen auswirken könnten.

DVSM

Siehe Abbildung des Wertstroms in der Entwicklung.

# E

EDA

Siehe explorative Datenanalyse.

Edge-Computing

Die Technologie, die die Rechenleistung für intelligente Geräte an den Rändern eines IoT-Netzwerks erhöht. Im Vergleich zu Cloud Computing kann Edge Computing die Kommunikationslatenz reduzieren und die Reaktionszeit verbessern.

Verschlüsselung

Ein Rechenprozess, der Klartextdaten, die für Menschen lesbar sind, in Chiffretext umwandelt.

Verschlüsselungsschlüssel

Eine kryptografische Zeichenfolge aus zufälligen Bits, die von einem Verschlüsselungsalgorithmus generiert wird. Schlüssel können unterschiedlich lang sein, und jeder Schlüssel ist so konzipiert, dass er unvorhersehbar und einzigartig ist.

Endianismus

Die Reihenfolge, in der Bytes im Computerspeicher gespeichert werden. Big-Endian-Systeme speichern das höchstwertige Byte zuerst. Little-Endian-Systeme speichern das niedrigwertigste Byte zuerst.

Endpunkt

Siehe Service-Endpunkt.

Endpunkt-Services

Ein Service, den Sie in einer Virtual Private Cloud (VPC) hosten können, um ihn mit anderen Benutzern zu teilen. Sie können einen Endpunktdienst mit anderen AWS-Konten oder AWS Identity and Access Management (IAM AWS PrivateLink -) Prinzipalen erstellen und diesen Berechtigungen gewähren. Diese Konten oder Prinzipale können sich privat mit Ihrem Endpunktservice verbinden, indem sie Schnittstellen-VPC-Endpunkte erstellen. Weitere Informationen finden Sie unter Einen Endpunkt-Service erstellen in der Amazon Virtual Private Cloud (Amazon VPC)-Dokumentation.

Unternehmensressourcenplanung (ERP)

Ein System, das wichtige Geschäftsprozesse (wie Buchhaltung, MES und Projektmanagement) für ein Unternehmen automatisiert und verwaltet.

Envelope-Verschlüsselung

Der Prozess der Verschlüsselung eines Verschlüsselungsschlüssels mit einem anderen Verschlüsselungsschlüssel. Weitere Informationen finden Sie unter Envelope-Verschlüsselung in der AWS Key Management Service (AWS KMS) -Dokumentation.

Umgebung

Eine Instance einer laufenden Anwendung. Die folgenden Arten von Umgebungen sind beim Cloud-Computing üblich:

- Entwicklungsumgebung – Eine Instance einer laufenden Anwendung, die nur dem Kernteam zur Verfügung steht, das für die Wartung der Anwendung verantwortlich ist. Entwicklungsumgebungen werden verwendet, um Änderungen zu testen, bevor sie in höhere Umgebungen übertragen werden. Diese Art von Umgebung wird manchmal als Testumgebung bezeichnet.
- Niedrigere Umgebungen – Alle Entwicklungsumgebungen für eine Anwendung, z. B. solche, die für erste Builds und Tests verwendet wurden.
- Produktionsumgebung – Eine Instance einer laufenden Anwendung, auf die Endbenutzer zugreifen können. In einer CI/CD-Pipeline ist die Produktionsumgebung die letzte Bereitstellungsumgebung.

- Höhere Umgebungen – Alle Umgebungen, auf die auch andere Benutzer als das Kernentwicklungsteam zugreifen können. Dies kann eine Produktionsumgebung, Vorproduktionsumgebungen und Umgebungen für Benutzerakzeptanztests umfassen.

Epics

In der agilen Methodik sind dies funktionale Kategorien, die Ihnen helfen, Ihre Arbeit zu organisieren und zu priorisieren. Epics bieten eine allgemeine Beschreibung der Anforderungen und Implementierungsaufgaben. Zu den Sicherheitsthemen AWS von CAF gehören beispielsweise Identitäts- und Zugriffsmanagement, Detektivkontrollen, Infrastruktursicherheit, Datenschutz und Reaktion auf Vorfälle. Weitere Informationen zu Epics in der AWS - Migrationsstrategie finden Sie im Leitfaden zur Programm-Implementierung.

ERP

Siehe Enterprise Resource Planning.

Explorative Datenanalyse (EDA)

Der Prozess der Analyse eines Datensatzes, um seine Hauptmerkmale zu verstehen. Sie sammeln oder aggregieren Daten und führen dann erste Untersuchungen durch, um Muster zu finden, Anomalien zu erkennen und Annahmen zu überprüfen. EDA wird durchgeführt, indem zusammenfassende Statistiken berechnet und Datenvisualisierungen erstellt werden.

# F

Faktentabelle

Die zentrale Tabelle in einem Sternschema. Sie speichert quantitative Daten über den Geschäftsbetrieb. In der Regel enthält eine Faktentabelle zwei Arten von Spalten: Spalten, die Kennzahlen enthalten, und Spalten, die einen Fremdschlüssel für eine Dimensionstabelle enthalten.

schnell scheitern

Eine Philosophie, die häufige und inkrementelle Tests verwendet, um den Entwicklungslebenszyklus zu verkürzen. Dies ist ein wichtiger Bestandteil eines agilen Ansatzes.

Grenze zur Fehlerisolierung

Dabei handelt es sich um eine Grenze AWS Cloud, z. B. eine Availability Zone AWS-Region, eine Steuerungsebene oder eine Datenebene, die die Auswirkungen eines Fehlers begrenzt und die

Widerstandsfähigkeit von Workloads verbessert. Weitere Informationen finden Sie unter Grenzen zur AWS Fehlerisolierung.

Feature-Zweig

Siehe Zweig.

Features

Die Eingabedaten, die Sie verwenden, um eine Vorhersage zu treffen. In einem Fertigungskontext könnten Feature beispielsweise Bilder sein, die regelmäßig von der Fertigungslinie aus aufgenommen werden.

Bedeutung der Feature

Wie wichtig ein Feature für die Vorhersagen eines Modells ist. Dies wird in der Regel als numerischer Wert ausgedrückt, der mit verschiedenen Techniken wie Shapley Additive Explanations (SHAP) und integrierten Gradienten berechnet werden kann. Weitere Informationen finden Sie unter Interpretierbarkeit von Modellen für maschinelles Lernen mit:AWS.

Featuretransformation

Daten für den ML-Prozess optimieren, einschließlich der Anreicherung von Daten mit zusätzlichen Quellen, der Skalierung von Werten oder der Extraktion mehrerer Informationssätze aus einem einzigen Datenfeld. Das ermöglicht dem ML-Modell, von den Daten profitieren. Wenn Sie beispielsweise das Datum „27.05.2021 00:15:37“ in „2021“, „Mai“, „Donnerstag“ und „15“ aufschlüsseln, können Sie dem Lernalgorithmus helfen, nuancierte Muster zu erlernen, die mit verschiedenen Datenkomponenten verknüpft sind.

FGAC

Weitere Informationen finden Sie unter detaillierter Zugriffskontrolle.

Feinkörnige Zugriffskontrolle (FGAC)

Die Verwendung mehrerer Bedingungen, um eine Zugriffsanfrage zuzulassen oder abzulehnen.

Flash-Cut-Migration

Eine Datenbankmigrationsmethode, bei der eine kontinuierliche Datenreplikation durch Erfassung von Änderungsdaten verwendet wird, um Daten in kürzester Zeit zu migrieren, anstatt einen schrittweisen Ansatz zu verwenden. Ziel ist es, Ausfallzeiten auf ein Minimum zu beschränken.

# G

Geoblocking

Siehe geografische Einschränkungen.

Geografische Einschränkungen (Geoblocking)

Bei Amazon eine Option CloudFront, um zu verhindern, dass Benutzer in bestimmten Ländern auf Inhaltsverteilungen zugreifen. Sie können eine Zulassungsliste oder eine Sperrliste verwenden, um zugelassene und gesperrte Länder anzugeben. Weitere Informationen finden Sie in der Dokumentation unter Beschränkung der geografischen Verteilung Ihrer Inhalte. CloudFront

Gitflow-Workflow

Ein Ansatz, bei dem niedrigere und höhere Umgebungen unterschiedliche Zweige in einem Quellcode-Repository verwenden. Der Gitflow-Workflow gilt als veraltet, und der Trunk-basierte Workflow ist der moderne, bevorzugte Ansatz.

Greenfield-Strategie

Das Fehlen vorhandener Infrastruktur in einer neuen Umgebung. Bei der Einführung einer Neuausrichtung einer Systemarchitektur können Sie alle neuen Technologien ohne Einschränkung der Kompatibilität mit der vorhandenen Infrastruktur auswählen, auch bekannt als Brownfield. Wenn Sie die bestehende Infrastruktur erweitern, könnten Sie Brownfield- und Greenfield-Strategien mischen.

Integritätsschutz

Eine allgemeine Regel, die dabei hilft, Ressourcen, Richtlinien und die Einhaltung von Vorschriften in allen Organisationseinheiten (OUs) zu regeln. Präventiver Integritätsschutz setzt Richtlinien durch, um die Einhaltung von Standards zu gewährleisten. Sie werden mithilfe von Service-Kontrollrichtlinien und IAM-Berechtigungsgrenzen implementiert. Detektivischer Integritätsschutz erkennt Richtlinienverstöße und Compliance-Probleme und generiert Warnmeldungen zur Abhilfe. Sie werden mithilfe von AWS Config, AWS Security Hub, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector und benutzerdefinierten AWS Lambda Prüfungen implementiert.

# H

HEKTAR

Siehe Hochverfügbarkeit.

Heterogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank in eine Zieldatenbank, die eine andere Datenbank-Engine verwendet (z. B. Oracle zu Amazon Aurora). Eine heterogene Migration ist in der Regel Teil einer Neuarchitektur, und die Konvertierung des Schemas kann eine komplexe Aufgabe sein. AWS bietet AWS SCT, welches bei Schemakonvertierungen hilft.

hohe Verfügbarkeit (HA)

Die Fähigkeit eines Workloads, im Falle von Herausforderungen oder Katastrophen kontinuierlich und ohne Eingreifen zu arbeiten. HA-Systeme sind so konzipiert, dass sie automatisch ein Failover durchführen, gleichbleibend hohe Leistung bieten und unterschiedliche Lasten und Ausfälle mit minimalen Leistungseinbußen bewältigen.

historische Modernisierung

Ein Ansatz zur Modernisierung und Aufrüstung von Betriebstechnologiesystemen (OT), um den Bedürfnissen der Fertigungsindustrie besser gerecht zu werden. Ein Historian ist eine Art von Datenbank, die verwendet wird, um Daten aus verschiedenen Quellen in einer Fabrik zu sammeln und zu speichern.

Homogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank zu einer Zieldatenbank, die dieselbe Datenbank-Engine verwendet (z. B. Microsoft SQL Server zu Amazon RDS für SQL Server). Eine homogene Migration ist in der Regel Teil eines Hostwechsels oder eines Plattformwechsels. Sie können native Datenbankserviceprogramme verwenden, um das Schema zu migrieren.

heiße Daten

Daten, auf die häufig zugegriffen wird, z. B. Echtzeitdaten oder aktuelle Translationsdaten. Für diese Daten ist in der Regel eine leistungsstarke Speicherebene oder -klasse erforderlich, um schnelle Abfrageantworten zu ermöglichen.

Hotfix

Eine dringende Lösung für ein kritisches Problem in einer Produktionsumgebung. Aufgrund seiner Dringlichkeit wird ein Hotfix normalerweise außerhalb des typischen DevOps Release-Workflows erstellt.

Hypercare-Phase

Unmittelbar nach dem Cutover, der Zeitraum, in dem ein Migrationsteam die migrierten Anwendungen in der Cloud verwaltet und überwacht, um etwaige Probleme zu beheben. In der Regel dauert dieser Zeitraum 1–4 Tage. Am Ende der Hypercare-Phase überträgt das Migrationsteam in der Regel die Verantwortung für die Anwendungen an das Cloud-Betriebsteam.

## I

IaC

Sehen Sie sich Infrastruktur als Code an.

Identitätsbasierte Richtlinie

Eine Richtlinie, die einem oder mehreren IAM-Prinzipalen zugeordnet ist und deren Berechtigungen innerhalb der AWS Cloud Umgebung definiert.

Leerlaufanwendung

Eine Anwendung mit einer durchschnittlichen CPU- und Arbeitsspeicherauslastung zwischen 5 und 20 Prozent über einen Zeitraum von 90 Tagen. In einem Migrationsprojekt ist es üblich, diese Anwendungen außer Betrieb zu nehmen oder sie On-Premises beizubehalten.

IIoT

Siehe Industrielles Internet der Dinge.

unveränderliche Infrastruktur

Ein Modell, das eine neue Infrastruktur für Produktionsworkloads bereitstellt, anstatt die bestehende Infrastruktur zu aktualisieren, zu patchen oder zu modifizieren. Unveränderliche Infrastrukturen sind von Natur aus konsistenter, zuverlässiger und vorhersehbarer als veränderliche Infrastrukturen. Weitere Informationen finden Sie in der Best Practice Deploy using immutable infrastructure im AWS Well-Architected Framework.

Eingehende (ingress) VPC

In einer Architektur AWS mit mehreren Konten ist dies eine VPC, die Netzwerkverbindungen von außerhalb einer Anwendung akzeptiert, überprüft und weiterleitet. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

Inkrementelle Migration

Eine Cutover-Strategie, bei der Sie Ihre Anwendung in kleinen Teilen migrieren, anstatt eine einziges vollständiges Cutover durchzuführen. Beispielsweise könnten Sie zunächst nur einige Microservices oder Benutzer auf das neue System umstellen. Nachdem Sie sich vergewissert haben, dass alles ordnungsgemäß funktioniert, können Sie weitere Microservices oder Benutzer schrittweise verschieben, bis Sie Ihr Legacy-System außer Betrieb nehmen können. Diese Strategie reduziert die mit großen Migrationen verbundenen Risiken.

Industrie 4.0

Ein Begriff, der 2016 von Klaus Schwab eingeführt wurde und sich auf die Modernisierung von Fertigungsprozessen durch Fortschritte in den Bereichen Konnektivität, Echtzeitdaten, Automatisierung, Analytik und KI/ML bezieht.

Infrastruktur

Alle Ressourcen und Komponenten, die in der Umgebung einer Anwendung enthalten sind.

Infrastructure as Code (IaC)

Der Prozess der Bereitstellung und Verwaltung der Infrastruktur einer Anwendung mithilfe einer Reihe von Konfigurationsdateien. IaC soll Ihnen helfen, das Infrastrukturmanagement zu zentralisieren, Ressourcen zu standardisieren und schnell zu skalieren, sodass neue Umgebungen wiederholbar, zuverlässig und konsistent sind.

Industrielles Internet der Dinge (IIoT)

Einsatz von mit dem Internet verbundenen Sensoren und Geräten in Industriesektoren wie Fertigung, Energie, Automobilindustrie, Gesundheitswesen, Biowissenschaften und Landwirtschaft. Mehr Informationen finden Sie unter Aufbau einer digitalen Transformationsstrategie für das industrielle Internet der Dinge (IIoT).

Inspektions-VPC

In einer Architektur AWS mit mehreren Konten eine zentralisierte VPC, die Inspektionen des Netzwerkverkehrs zwischen VPCs (in derselben oder unterschiedlichen AWS-Regionen), dem Internet und lokalen Netzwerken verwaltet. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

Internet of Things (IoT)

Das Netzwerk verbundener physischer Objekte mit eingebetteten Sensoren oder Prozessoren, das über das Internet oder über ein lokales Kommunikationsnetzwerk mit anderen Geräten und Systemen kommuniziert. Weitere Informationen finden Sie unter Was ist IoT?

Interpretierbarkeit

Ein Merkmal eines Modells für Machine Learning, das beschreibt, inwieweit ein Mensch verstehen kann, wie die Vorhersagen des Modells von seinen Eingaben abhängen. Weitere Informationen finden Sie unter Interpretierbarkeit von Modellen für Machine Learning mit AWS.

IoT

Siehe Internet der Dinge.

IT information library (ITIL, IT-Informationsbibliothek)

Eine Reihe von bewährten Methoden für die Bereitstellung von IT-Services und die Abstimmung dieser Services auf die Geschäftsanforderungen. ITIL bietet die Grundlage für ITSM.

T service management (ITSM, IT-Servicemanagement)

Aktivitäten im Zusammenhang mit der Gestaltung, Implementierung, Verwaltung und Unterstützung von IT-Services für eine Organisation. Informationen zur Integration von Cloud-Vorgängen mit ITSM-Tools finden Sie im Leitfaden zur Betriebsintegration.

BIS

Weitere Informationen finden Sie in der IT-Informationsbibliothek.

ITSM

Siehe IT-Servicemanagement.

# L

Labelbasierte Zugangskontrolle (LBAC)

Eine Implementierung der Mandatory Access Control (MAC), bei der den Benutzern und den Daten selbst jeweils explizit ein Sicherheitslabelwert zugewiesen wird. Die Schnittmenge zwischen der Benutzersicherheitsbeschriftung und der Datensicherheitsbeschriftung bestimmt, welche Zeilen und Spalten für den Benutzer sichtbar sind.

Landing Zone

Eine landing zone ist eine gut strukturierte AWS Umgebung mit mehreren Konten, die skalierbar und sicher ist. Dies ist ein Ausgangspunkt, von dem aus Ihre Organisationen Workloads und Anwendungen schnell und mit Vertrauen in ihre Sicherheits- und Infrastrukturumgebung starten und bereitstellen können. Weitere Informationen zu Landing Zones finden Sie unter Einrichtung einer sicheren und skalierbaren AWS -Umgebung mit mehreren Konten..

Große Migration

Eine Migration von 300 oder mehr Servern.

SCHWARZ

Weitere Informationen finden Sie unter Label-basierte Zugriffskontrolle.

Geringste Berechtigung

Die bewährte Sicherheitsmethode, bei der nur die für die Durchführung einer Aufgabe erforderlichen Mindestberechtigungen erteilt werden. Weitere Informationen finden Sie unter Geringste Berechtigungen anwenden in der IAM-Dokumentation.

Lift and Shift

Siehe 7 Rs.

Little-Endian-System

Ein System, welches das niedrigwertigste Byte zuerst speichert. Siehe auch Endianness.

Niedrigere Umgebungen

Siehe Umwelt.

# M

Machine Learning (ML)

Eine Art künstlicher Intelligenz, die Algorithmen und Techniken zur Mustererkennung und zum Lernen verwendet. ML analysiert aufgezeichnete Daten, wie z. B. Daten aus dem Internet der Dinge (IoT), und lernt daraus, um ein statistisches Modell auf der Grundlage von Mustern zu erstellen. Weitere Informationen finden Sie unter Machine Learning.

Hauptzweig

Siehe Filiale.

Malware

Software, die entwickelt wurde, um die Computersicherheit oder den Datenschutz zu gefährden. Malware kann Computersysteme stören, vertrauliche Informationen durchsickern lassen oder sich unbefugten Zugriff verschaffen. Beispiele für Malware sind Viren, Würmer, Ransomware, Trojaner, Spyware und Keylogger.

verwaltete Dienste

AWS-Services für die die Infrastrukturebene, das Betriebssystem und die Plattformen AWS betrieben werden, und Sie greifen auf die Endgeräte zu, um Daten zu speichern und abzurufen. Amazon Simple Storage Service (Amazon S3) und Amazon DynamoDB sind Beispiele für Managed Services. Diese werden auch als abstrakte Dienste bezeichnet.

Manufacturing Execution System (MES)

Ein Softwaresystem zur Nachverfolgung, Überwachung, Dokumentation und Steuerung von Produktionsprozessen, bei denen Rohstoffe in der Fertigung zu fertigen Produkten umgewandelt werden.

MAP

Siehe Migration Acceleration Program.

Mechanismus

Ein vollständiger Prozess, bei dem Sie ein Tool erstellen, die Akzeptanz des Tools vorantreiben und anschließend die Ergebnisse überprüfen, um Anpassungen vorzunehmen. Ein Mechanismus ist ein Zyklus, der sich im Laufe seiner Tätigkeit selbst verstärkt und verbessert. Weitere Informationen finden Sie unter Aufbau von Mechanismen im AWS Well-Architected Framework.

Mitgliedskonto

Alle AWS-Konten außer dem Verwaltungskonto, die Teil einer Organisation in sind. AWS Organizations Ein Konto kann jeweils nur einer Organisation angehören.

DURCHEINANDER

Siehe Manufacturing Execution System.

Message Queuing-Telemetrietransport (MQTT)

Ein leichtes machine-to-machine (M2M) -Kommunikationsprotokoll, das auf dem Publish/Subscribe-Muster für IoT-Geräte mit beschränkten Ressourcen basiert.

Microservice

Ein kleiner, unabhängiger Service, der über klar definierte APIs kommuniziert und in der Regel kleinen, eigenständigen Teams gehört. Ein Versicherungssystem kann beispielsweise Microservices beinhalten, die Geschäftsfunktionen wie Vertrieb oder Marketing oder Subdomains wie Einkauf, Schadenersatz oder Analytik zugeordnet sind. Zu den Vorteilen von Microservices gehören Agilität, flexible Skalierung, einfache Bereitstellung, wiederverwendbarer Code und Ausfallsicherheit. Weitere Informationen finden Sie unter Integration von Microservices mithilfe serverloser Dienste. AWS

Microservices-Architekturen

Ein Ansatz zur Erstellung einer Anwendung mit unabhängigen Komponenten, die jeden Anwendungsprozess als Microservice ausführen. Diese Microservices kommunizieren über eine klar definierte Schnittstelle mithilfe einfacher APIs. Jeder Microservice in dieser Architektur kann aktualisiert, bereitgestellt und skaliert werden, um den Bedarf an bestimmten Funktionen einer Anwendung zu decken. Weitere Informationen finden Sie unter Implementierung von Microservices auf. AWS

Migration Acceleration Program (MAP)

Ein AWS Programm, das Beratung, Unterstützung, Schulungen und Services bietet, um Unternehmen dabei zu unterstützen, eine solide betriebliche Grundlage für die Umstellung auf die Cloud zu schaffen und die anfänglichen Kosten von Migrationen auszugleichen. MAP umfasst eine Migrationsmethode für die methodische Durchführung von Legacy-Migrationen sowie eine Reihe von Tools zur Automatisierung und Beschleunigung gängiger Migrationsszenarien.

Migration in großem Maßstab

Der Prozess, bei dem der Großteil des Anwendungsportfolios in Wellen in die Cloud verlagert wird, wobei in jeder Welle mehr Anwendungen schneller migriert werden. In dieser Phase werden die bewährten Verfahren und Erkenntnisse aus den früheren Phasen zur Implementierung einer Migrationsfabrik von Teams, Tools und Prozessen zur Optimierung der Migration von Workloads durch Automatisierung und agile Bereitstellung verwendet. Dies ist die dritte Phase der AWS - Migrationsstrategie.

Migrationsfabrik

Funktionsübergreifende Teams, die die Migration von Workloads durch automatisierte, agile Ansätze optimieren. Zu den Teams in der Migrationsabteilung gehören in der Regel Betriebsabläufe, Geschäftsanalysten und Eigentümer, Migrationsingenieure, Entwickler und DevOps Experten, die in Sprints arbeiten. Zwischen 20 und 50 Prozent eines Unternehmensanwendungsportfolios bestehen aus sich wiederholenden Mustern, die durch einen Fabrik-Ansatz optimiert werden können. Weitere Informationen finden Sie in Diskussion über Migrationsfabriken und den Leitfaden zur Cloud-Migration-Fabrik in diesem Inhaltssatz.

Migrationsmetadaten

Die Informationen über die Anwendung und den Server, die für den Abschluss der Migration benötigt werden. Für jedes Migrationsmuster ist ein anderer Satz von Migrationsmetadaten erforderlich. Beispiele für Migrationsmetadaten sind das Zielsubnetz, die Sicherheitsgruppe und AWS das Konto.

Migrationsmuster

Eine wiederholbare Migrationsaufgabe, in der die Migrationsstrategie, das Migrationsziel und die verwendete Migrationsanwendung oder der verwendete Migrationsservice detailliert beschrieben werden. Beispiel: Rehost-Migration zu Amazon EC2 mit AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

Ein Online-Tool, das Informationen zur Validierung des Geschäftsszenarios für die Migration auf das bereitstellt. AWS Cloud MPA bietet eine detaillierte Portfoliobewertung (richtige Servergröße, Preisgestaltung, Gesamtbetriebskostenanalyse, Migrationskostenanalyse) sowie Migrationsplanung (Anwendungsdatenanalyse und Datenerfassung, Anwendungsgruppierung, Migrationspriorisierung und Wellenplanung). Das MPA-Tool (Anmeldung erforderlich) steht allen AWS Beratern und APN-Partnerberatern kostenlos zur Verfügung.

Migration Readiness Assessment (MRA)

Der Prozess, bei dem mithilfe des AWS CAF Erkenntnisse über den Cloud-Bereitschaftsstatus eines Unternehmens gewonnen, Stärken und Schwächen identifiziert und ein Aktionsplan zur Schließung festgestellter Lücken erstellt wird. Weitere Informationen finden Sie im Benutzerhandbuch für Migration Readiness. MRA ist die erste Phase der AWS - Migrationsstrategie.

Migrationsstrategie

Der Ansatz, der verwendet wurde, um einen Workload auf den AWS Cloud zu migrieren. Weitere Informationen finden Sie im Eintrag 7 Rs in diesem Glossar und unter Mobilisieren Sie Ihr Unternehmen, um umfangreiche Migrationen zu beschleunigen.

ML

Siehe maschinelles Lernen.

Modernisierung

Umwandlung einer veralteten (veralteten oder monolithischen) Anwendung und ihrer Infrastruktur in ein agiles, elastisches und hochverfügbares System in der Cloud, um Kosten zu senken, die Effizienz zu steigern und Innovationen zu nutzen. Weitere Informationen finden Sie unter Strategie zur Modernisierung von Anwendungen in der AWS Cloud.

Bewertung der Modernisierungsfähigkeit

Eine Bewertung, anhand derer festgestellt werden kann, ob die Anwendungen einer Organisation für die Modernisierung bereit sind, Vorteile, Risiken und Abhängigkeiten identifiziert und ermittelt wird, wie gut die Organisation den zukünftigen Status dieser Anwendungen unterstützen kann. Das Ergebnis der Bewertung ist eine Vorlage der Zielarchitektur, eine Roadmap, in der die Entwicklungsphasen und Meilensteine des Modernisierungsprozesses detailliert beschrieben werden, sowie ein Aktionsplan zur Behebung festgestellter Lücken. Weitere Informationen finden Sie unter Evaluierung der Modernisierungsbereitschaft von Anwendungen in der AWS Cloud.

Monolithische Anwendungen (Monolithen)

Anwendungen, die als ein einziger Service mit eng gekoppelten Prozessen ausgeführt werden. Monolithische Anwendungen haben verschiedene Nachteile. Wenn ein Anwendungs-Feature stark nachgefragt wird, muss die gesamte Architektur skaliert werden. Das Hinzufügen oder Verbessern der Feature einer monolithischen Anwendung wird ebenfalls komplexer, wenn die Codebasis wächst. Um diese Probleme zu beheben, können Sie eine Microservices-Architektur verwenden. Weitere Informationen finden Sie unter Zerlegen von Monolithen in Microservices.

MPA

Siehe Bewertung des Migrationsportfolios.

MQTT

Siehe Message Queuing-Telemetrietransport.

Mehrklassen-Klassifizierung

Ein Prozess, der dabei hilft, Vorhersagen für mehrere Klassen zu generieren (wobei eines von mehr als zwei Ergebnissen vorhergesagt wird). Ein ML-Modell könnte beispielsweise fragen: „Ist dieses Produkt ein Buch, ein Auto oder ein Telefon?“ oder „Welche Kategorie von Produkten ist für diesen Kunden am interessantesten?“

veränderbare Infrastruktur

Ein Modell, das die bestehende Infrastruktur für Produktionsworkloads aktualisiert und modifiziert. Für eine verbesserte Konsistenz, Zuverlässigkeit und Vorhersagbarkeit empfiehlt das AWS Well-Architected Framework die Verwendung einer unveränderlichen Infrastruktur als bewährte Methode.

# O

OAC

Weitere Informationen finden Sie unter Origin Access Control.

OAI

Siehe Zugriffsidentität von Origin.

COM

Siehe organisatorisches Change-Management.

Offline-Migration

Eine Migrationsmethode, bei der der Quell-Workload während des Migrationsprozesses heruntergefahren wird. Diese Methode ist mit längeren Ausfallzeiten verbunden und wird in der Regel für kleine, unkritische Workloads verwendet.

OI

Siehe Betriebsintegration.

OLA

Siehe Vereinbarung auf operativer Ebene.

Online-Migration

Eine Migrationsmethode, bei der der Quell-Workload auf das Zielsystem kopiert wird, ohne offline genommen zu werden. Anwendungen, die mit dem Workload verbunden sind, können während der Migration weiterhin funktionieren. Diese Methode beinhaltet keine bis minimale Ausfallzeit und wird in der Regel für kritische Produktionsworkloads verwendet.

OPC-UA

Siehe Open Process Communications — Unified Architecture.

Offene Prozesskommunikation — Einheitliche Architektur (OPC-UA)

Ein machine-to-machine (M2M) -Kommunikationsprotokoll für die industrielle Automatisierung. OPC-UA bietet einen Interoperabilitätsstandard mit Datenverschlüsselungs-, Authentifizierungs- und Autorisierungsschemata.

Vereinbarung auf Betriebsebene (OLA)

Eine Vereinbarung, in der klargestellt wird, welche funktionalen IT-Gruppen sich gegenseitig versprechen zu liefern, um ein Service Level Agreement (SLA) zu unterstützen.

Überprüfung der Betriebsbereitschaft (ORR)

Eine Checkliste mit Fragen und zugehörigen bewährten Methoden, die Ihnen helfen, Vorfälle und mögliche Ausfälle zu verstehen, zu bewerten, zu verhindern oder deren Umfang zu reduzieren. Weitere Informationen finden Sie unter Operational Readiness Reviews (ORR) im AWS Well-Architected Framework.

Betriebstechnologie (OT)

Hardware- und Softwaresysteme, die mit der physischen Umgebung zusammenarbeiten, um industrielle Abläufe, Ausrüstung und Infrastruktur zu steuern. In der Fertigung ist die Integration von OT- und Informationstechnologie (IT) -Systemen ein zentraler Schwerpunkt der Industrie 4.0-Transformationen.

Betriebsintegration (OI)

Der Prozess der Modernisierung von Abläufen in der Cloud, der Bereitschaftsplanung, Automatisierung und Integration umfasst. Weitere Informationen finden Sie im Leitfaden zur Betriebsintegration.

Organisationspfad

Ein Pfad, der von erstellt wird und in AWS CloudTrail dem alle Ereignisse für alle AWS-Konten in einer Organisation protokolliert werden. AWS Organizations Diese Spur wird in jedem AWS-Konto , der Teil der Organisation ist, erstellt und verfolgt die Aktivität in jedem Konto. Weitere Informationen finden Sie in der CloudTrail Dokumentation unter Erstellen eines Pfads für eine Organisation.

Organisatorisches Veränderungsmanagement (OCM)

Ein Framework für das Management wichtiger, disruptiver Geschäftstransformationen aus Sicht der Mitarbeiter, der Kultur und der Führung. OCM hilft Organisationen dabei, sich auf neue Systeme und Strategien vorzubereiten und auf diese umzustellen, indem es die Akzeptanz von Veränderungen beschleunigt, Übergangsprobleme angeht und kulturelle und organisatorische Veränderungen vorantreibt. In der AWS Migrationsstrategie wird dieses Framework aufgrund der Geschwindigkeit des Wandels, der bei Projekten zur Cloud-Einführung erforderlich ist, als Mitarbeiterbeschleunigung bezeichnet. Weitere Informationen finden Sie im OCM-Handbuch.

Ursprungszugriffskontrolle (OAC)

In CloudFront, eine erweiterte Option zur Zugriffsbeschränkung, um Ihre Amazon Simple Storage Service (Amazon S3) -Inhalte zu sichern. OAC unterstützt alle S3-Buckets insgesamt AWS-Regionen, serverseitige Verschlüsselung mit AWS KMS (SSE-KMS) sowie dynamische `PUT` und `DELETE` Anfragen an den S3-Bucket.

Ursprungszugriffsidentität (OAI)

In CloudFront, eine Option zur Zugriffsbeschränkung, um Ihre Amazon S3 S3-Inhalte zu sichern. Wenn Sie OAI verwenden, CloudFront erstellt es einen Principal, mit dem sich Amazon S3 authentifizieren kann. Authentifizierte Principals können nur über eine bestimmte Distribution auf Inhalte in einem S3-Bucket zugreifen. CloudFront Siehe auch OAC, das eine detailliertere und verbesserte Zugriffskontrolle bietet.

ODER

Siehe Überprüfung der Betriebsbereitschaft.

NICHT

Siehe Betriebstechnologie.

Ausgehende (egress) VPC

In einer Architektur AWS mit mehreren Konten eine VPC, die Netzwerkverbindungen verarbeitet, die von einer Anwendung aus initiiert werden. Die AWS -Referenzarchitektur für die Sicherheit empfiehlt, Ihr Netzwerkkonto mit eingehenden und ausgehenden VPCs und Inspektions-VPCs einzurichten, um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet zu schützen.

# P

Berechtigungsgrenze

Eine IAM-Verwaltungsrichtlinie, die den IAM-Prinzipalen zugeordnet ist, um die maximalen Berechtigungen festzulegen, die der Benutzer oder die Rolle haben kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitys in der IAM-Dokumentation.

persönlich identifizierbare Informationen (PII)

Informationen, die, wenn sie direkt betrachtet oder mit anderen verwandten Daten kombiniert werden, verwendet werden können, um vernünftige Rückschlüsse auf die Identität einer Person zu ziehen. Beispiele für personenbezogene Daten sind Namen, Adressen und Kontaktinformationen.

Personenbezogene Daten

Siehe persönlich identifizierbare Informationen.

Playbook

Eine Reihe vordefinierter Schritte, die die mit Migrationen verbundenen Aufgaben erfassen, z. B. die Bereitstellung zentraler Betriebsfunktionen in der Cloud. Ein Playbook kann die Form von Skripten, automatisierten Runbooks oder einer Zusammenfassung der Prozesse oder Schritte annehmen, die für den Betrieb Ihrer modernisierten Umgebung erforderlich sind.

PLC

Siehe programmierbare Logiksteuerung.

PLM

Siehe Produktlebenszyklusmanagement.

policy

Ein Objekt, das Berechtigungen definieren (siehe identitätsbasierte Richtlinie), Zugriffsbedingungen spezifizieren (siehe ressourcenbasierte Richtlinie) oder die maximalen Berechtigungen für alle Konten in einer Organisation definieren kann AWS Organizations (siehe Dienststeuerungsrichtlinie).

Polyglotte Beharrlichkeit

Unabhängige Auswahl der Datenspeichertechnologie eines Microservices auf der Grundlage von Datenzugriffsmustern und anderen Anforderungen. Wenn Ihre Microservices über dieselbe Datenspeichertechnologie verfügen, kann dies zu Implementierungsproblemen oder zu Leistungseinbußen führen. Microservices lassen sich leichter implementieren und erzielen eine bessere Leistung und Skalierbarkeit, wenn sie den Datenspeicher verwenden, der ihren Anforderungen am besten entspricht. Weitere Informationen finden Sie unter Datenpersistenz in Microservices aktivieren.

Portfoliobewertung

Ein Prozess, bei dem das Anwendungsportfolio ermittelt, analysiert und priorisiert wird, um die Migration zu planen. Weitere Informationen finden Sie in Bewerten der Migrationsbereitschaft.

predicate

Eine Abfragebedingung, die `true` oder zurückgibt`false`, was üblicherweise in einer Klausel vorkommt. `WHERE`

Prädikat Pushdown

Eine Technik zur Optimierung von Datenbankabfragen, bei der die Daten in der Abfrage vor der Übertragung gefiltert werden. Dadurch wird die Datenmenge reduziert, die aus der relationalen Datenbank abgerufen und verarbeitet werden muss, und die Abfrageleistung wird verbessert.

Präventive Kontrolle

Eine Sicherheitskontrolle, die verhindern soll, dass ein Ereignis eintritt. Diese Kontrollen stellen eine erste Verteidigungslinie dar, um unbefugten Zugriff oder unerwünschte Änderungen an Ihrem Netzwerk zu verhindern. Weitere Informationen finden Sie unter Präventive Kontrolle in Implementierung von Sicherheitskontrollen in AWS.

Prinzipal

Eine Entität AWS , die Aktionen ausführen und auf Ressourcen zugreifen kann. Bei dieser Entität handelt es sich in der Regel um einen Root-Benutzer für eine AWS-Konto, eine IAM-Rolle oder

einen Benutzer. Weitere Informationen finden Sie unter Prinzipal in Rollenbegriffe und -konzepte in der IAM-Dokumentation.

Datenschutz durch Design

Ein Ansatz in der Systemtechnik, der den Datenschutz während des gesamten Engineering-Prozesses berücksichtigt.

Privat gehostete Zonen

Ein Container, der Informationen darüber enthält, wie Amazon Route 53 auf DNS-Abfragen für eine Domain und ihre Subdomains innerhalb einer oder mehrerer VPCs reagieren soll. Weitere Informationen finden Sie unter Arbeiten mit privat gehosteten Zonen in der Route-53-Dokumentation.

proaktive Steuerung

Eine Sicherheitskontrolle, die den Einsatz nicht richtlinienkonformer Ressourcen verhindern soll. Mit diesen Steuerelementen werden Ressourcen gescannt, bevor sie bereitgestellt werden. Wenn die Ressource nicht mit der Steuerung konform ist, wird sie nicht bereitgestellt. Weitere Informationen finden Sie im Referenzhandbuch zu Kontrollen in der AWS Control Tower Dokumentation und unter Proaktive Kontrollen unter Implementierung von Sicherheitskontrollen am AWS.

Produktlebenszyklusmanagement (PLM)

Das Management von Daten und Prozessen für ein Produkt während seines gesamten Lebenszyklus, vom Design, der Entwicklung und Markteinführung über Wachstum und Reife bis hin zur Markteinführung und Markteinführung.

Produktionsumgebung

Siehe Umgebung.

Speicherprogrammierbare Steuerung (SPS)

In der Fertigung ein äußerst zuverlässiger, anpassungsfähiger Computer, der Maschinen überwacht und Fertigungsprozesse automatisiert.

Pseudonymisierung

Der Prozess, bei dem persönliche Identifikatoren in einem Datensatz durch Platzhalterwerte ersetzt werden. Pseudonymisierung kann zum Schutz der Privatsphäre beitragen. Pseudonymisierte Daten gelten weiterhin als personenbezogene Daten.

veröffentlichen/abonnieren (pub/sub)

Ein Muster, das asynchrone Kommunikation zwischen Microservices ermöglicht, um die Skalierbarkeit und Reaktionsfähigkeit zu verbessern. In einem auf Microservices basierenden MES kann ein Microservice beispielsweise Ereignismeldungen in einem Kanal veröffentlichen, den andere Microservices abonnieren können. Das System kann neue Microservices hinzufügen, ohne den Veröffentlichungsservice zu ändern.

# Q

Abfrageplan

Eine Reihe von Schritten, wie Anweisungen, die für den Zugriff auf die Daten in einem relationalen SQL-Datenbanksystem verwendet werden.

Abfrageplanregression

Wenn ein Datenbankserviceoptimierer einen weniger optimalen Plan wählt als vor einer bestimmten Änderung der Datenbankumgebung. Dies kann durch Änderungen an Statistiken, Beschränkungen, Umgebungseinstellungen, Abfrageparameter-Bindungen und Aktualisierungen der Datenbank-Engine verursacht werden.

# R

RACI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

Ransomware

Eine bösartige Software, die entwickelt wurde, um den Zugriff auf ein Computersystem oder Daten zu blockieren, bis eine Zahlung erfolgt ist.

RASCI-Matrix

Siehe verantwortlich, rechenschaftspflichtig, konsultiert, informiert (RACI).

RCAC

Siehe Zugriffskontrolle für Zeilen und Spalten.

Read Replica

Eine Kopie einer Datenbank, die nur für Lesezwecke verwendet wird. Sie können Abfragen an das Lesereplikat weiterleiten, um die Belastung auf Ihrer Primärdatenbank zu reduzieren.

neu strukturieren

Siehe 7 Rs.

Recovery Point Objective (RPO)

Die maximal zulässige Zeitspanne seit dem letzten Datenwiederherstellungspunkt. Dies bestimmt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Betriebsunterbrechung angesehen wird.

Wiederherstellungszeitziel (RTO)

Die maximal zulässige Verzögerung zwischen der Betriebsunterbrechung und der Wiederherstellung des Dienstes.

Refaktorierung

Siehe 7 Rs.

Region

Eine Sammlung von AWS Ressourcen in einem geografischen Gebiet. Jeder AWS-Region ist isoliert und unabhängig von den anderen, um Fehlertoleranz, Stabilität und Belastbarkeit zu gewährleisten. Weitere Informationen finden Sie unter Geben Sie an, was AWS-Regionen Ihr Konto verwenden kann.

Regression

Eine ML-Technik, die einen numerischen Wert vorhersagt. Zum Beispiel, um das Problem „Zu welchem Preis wird dieses Haus verkauft werden?“ zu lösen Ein ML-Modell könnte ein lineares Regressionsmodell verwenden, um den Verkaufspreis eines Hauses auf der Grundlage bekannter Fakten über das Haus (z. B. die Quadratmeterzahl) vorherzusagen.

rehosten

Siehe 7 Rs.

Veröffentlichung

In einem Bereitstellungsprozess der Akt der Förderung von Änderungen an einer Produktionsumgebung.

umziehen

Siehe 7 Rs.

neue Plattform

Siehe 7 Rs.

Rückkauf

Siehe 7 Rs.

Ausfallsicherheit

Die Fähigkeit einer Anwendung, Störungen zu widerstehen oder sich von ihnen zu erholen. Hochverfügbarkeit und Notfallwiederherstellung sind häufig Überlegungen bei der Planung der Ausfallsicherheit in der. AWS Cloud Weitere Informationen finden Sie unter AWS Cloud Resilienz.

Ressourcenbasierte Richtlinie

Eine mit einer Ressource verknüpfte Richtlinie, z. B. ein Amazon-S3-Bucket, ein Endpunkt oder ein Verschlüsselungsschlüssel. Diese Art von Richtlinie legt fest, welchen Prinzipalen der Zugriff gewährt wird, welche Aktionen unterstützt werden und welche anderen Bedingungen erfüllt sein müssen.

RACI-Matrix (verantwortlich, rechenschaftspflichtig, konsultiert, informiert)

Eine Matrix, die die Rollen und Verantwortlichkeiten aller an Migrationsaktivitäten und Cloud-Operationen beteiligten Parteien definiert. Der Matrixname leitet sich von den in der Matrix definierten Zuständigkeitstypen ab: verantwortlich (R), rechenschaftspflichtig (A), konsultiert (C) und informiert (I). Der Unterstützungstyp (S) ist optional. Wenn Sie Unterstützung einbeziehen, wird die Matrix als RASCI-Matrix bezeichnet, und wenn Sie sie ausschließen, wird sie als RACI-Matrix bezeichnet.

Reaktive Kontrolle

Eine Sicherheitskontrolle, die darauf ausgelegt ist, die Behebung unerwünschter Ereignisse oder Abweichungen von Ihren Sicherheitsstandards voranzutreiben. Weitere Informationen finden Sie unter Reaktive Kontrolle in Implementieren von Sicherheitskontrollen in AWS.

Beibehaltung

Siehe 7 Rs.

zurückziehen

Siehe 7 Rs.

Drehung

Der Vorgang, bei dem ein Geheimnis regelmäßig aktualisiert wird, um es einem Angreifer zu erschweren, auf die Anmeldeinformationen zuzugreifen.

Zugriffskontrolle für Zeilen und Spalten (RCAC)

Die Verwendung einfacher, flexibler SQL-Ausdrücke mit definierten Zugriffsregeln. RCAC besteht aus Zeilenberechtigungen und Spaltenmasken.

RPO

Siehe Recovery Point Objective.

RTO

Siehe Ziel der Wiederherstellungszeit.

Runbook

Eine Reihe manueller oder automatisierter Verfahren, die zur Ausführung einer bestimmten Aufgabe erforderlich sind. Diese sind in der Regel darauf ausgelegt, sich wiederholende Operationen oder Verfahren mit hohen Fehlerquoten zu rationalisieren.

# S

SAML 2.0

Ein offener Standard, den viele Identitätsanbieter (IdPs) verwenden. Diese Funktion ermöglicht föderiertes Single Sign-On (SSO), sodass sich Benutzer bei den API-Vorgängen anmelden AWS Management Console oder die AWS API-Operationen aufrufen können, ohne dass Sie einen Benutzer in IAM für alle in Ihrer Organisation erstellen müssen. Weitere Informationen zum SAML-2.0.-basierten Verbund finden Sie unter Über den SAML-2.0-basierten Verbund in der IAM-Dokumentation.

SCADA

Siehe Aufsichtskontrolle und Datenerfassung.

SCP

Siehe Richtlinie zur Dienstkontrolle.

Secret

Interne AWS Secrets Manager, vertrauliche oder eingeschränkte Informationen, wie z. B. ein Passwort oder Benutzeranmeldeinformationen, die Sie in verschlüsselter Form speichern. Es besteht aus dem geheimen Wert und seinen Metadaten. Der geheime Wert kann binär, eine einzelne Zeichenfolge oder mehrere Zeichenketten sein. Weitere Informationen finden Sie unter Was ist in einem Secrets Manager Manager-Geheimnis? in der Secrets Manager Manager-Dokumentation.

Sicherheitskontrolle

Ein technischer oder administrativer Integritätsschutz, der die Fähigkeit eines Bedrohungsakteurs, eine Schwachstelle auszunutzen, verhindert, erkennt oder einschränkt. Es gibt vier Haupttypen von Sicherheitskontrollen: präventiv, detektiv, reaktionsschnell und proaktiv.

Härtung der Sicherheit

Der Prozess, bei dem die Angriffsfläche reduziert wird, um sie widerstandsfähiger gegen Angriffe zu machen. Dies kann Aktionen wie das Entfernen von Ressourcen, die nicht mehr benötigt werden, die Implementierung der bewährten Sicherheitsmethode der Gewährung geringster Berechtigungen oder die Deaktivierung unnötiger Feature in Konfigurationsdateien umfassen.

System zur Verwaltung von Sicherheitsinformationen und Ereignissen (security information and event management – SIEM)

Tools und Services, die Systeme für das Sicherheitsinformationsmanagement (SIM) und das Management von Sicherheitsereignissen (SEM) kombinieren. Ein SIEM-System sammelt, überwacht und analysiert Daten von Servern, Netzwerken, Geräten und anderen Quellen, um Bedrohungen und Sicherheitsverletzungen zu erkennen und Warnmeldungen zu generieren.

Automatisierung von Sicherheitsreaktionen

Eine vordefinierte und programmierte Aktion, die darauf ausgelegt ist, automatisch auf ein Sicherheitsereignis zu reagieren oder es zu beheben. Diese Automatisierungen dienen als detektive oder reaktionsschnelle Sicherheitskontrollen, die Sie bei der Implementierung bewährter AWS Sicherheitsmethoden unterstützen. Beispiele für automatisierte Antwortaktionen sind das Ändern einer VPC-Sicherheitsgruppe, das Patchen einer Amazon EC2 EC2-Instance oder das Rotieren von Anmeldeinformationen.

Serverseitige Verschlüsselung

Verschlüsselung von Daten am Zielort durch denjenigen AWS-Service , der sie empfängt.

Service-Kontrollrichtlinie (SCP)

Eine Richtlinie, die eine zentrale Kontrolle über die Berechtigungen für alle Konten in einer Organisation in AWS Organizations ermöglicht. SCPs definieren Integritätsschutz oder legen Grenzwerte für Aktionen fest, die ein Administrator an Benutzer oder Rollen delegieren kann. Sie können SCPs als Zulassungs- oder Ablehnungslisten verwenden, um festzulegen, welche Services oder Aktionen zulässig oder verboten sind. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter Richtlinien zur Dienststeuerung.

Service-Endpunkt

Die URL des Einstiegspunkts für einen AWS-Service. Sie können den Endpunkt verwenden, um programmgesteuert eine Verbindung zum Zielservice herzustellen. Weitere Informationen finden Sie unter AWS-Service -Endpunkte in der Allgemeine AWS-Referenz.

Service Level Agreement (SLA)

Eine Vereinbarung, in der klargestellt wird, was ein IT-Team seinen Kunden zu bieten verspricht, z. B. in Bezug auf Verfügbarkeit und Leistung der Services.

Service-Level-Indikator (SLI)

Eine Messung eines Leistungsaspekts eines Dienstes, z. B. seiner Fehlerrate, Verfügbarkeit oder Durchsatz.

Service-Level-Ziel (SLO)

Eine Zielkennzahl, die den Zustand eines Dienstes darstellt, gemessen anhand eines Service-Level-Indikators.

Modell der geteilten Verantwortung

Ein Modell, das die Verantwortung beschreibt, mit der Sie gemeinsam AWS für Cloud-Sicherheit und Compliance verantwortlich sind. AWS ist für die Sicherheit der Cloud verantwortlich, wohingegen Sie für die Sicherheit in der Cloud verantwortlich sind. Weitere Informationen finden Sie unter Modell der geteilten Verantwortung.

SIEM

Siehe Sicherheitsinformations- und Event-Management-System.

Single Point of Failure (SPOF)

Ein Fehler in einer einzelnen, kritischen Komponente einer Anwendung, der das System stören kann.

SLA

Siehe Service Level Agreement.

SLI

Siehe Service-Level-Indikator.

ALSO

Siehe Service-Level-Ziel.

split-and-seed Modell

Ein Muster für die Skalierung und Beschleunigung von Modernisierungsprojekten. Sobald neue Features und Produktversionen definiert werden, teilt sich das Kernteam auf, um neue Produktteams zu bilden. Dies trägt zur Skalierung der Fähigkeiten und Services Ihrer Organisation bei, verbessert die Produktivität der Entwickler und unterstützt schnelle Innovationen. Weitere Informationen finden Sie unter Schrittweiser Ansatz zur Modernisierung von Anwendungen in der AWS Cloud.

SPOTTEN

Siehe Single Point of Failure.

Sternschema

Eine Datenbank-Organisationsstruktur, die eine große Faktentabelle zum Speichern von Transaktions- oder Messdaten und eine oder mehrere kleinere dimensionale Tabellen zum Speichern von Datenattributen verwendet. Diese Struktur ist für die Verwendung in einem Data Warehouse oder für Business Intelligence-Zwecke konzipiert.

Strangler-Fig-Muster

Ein Ansatz zur Modernisierung monolithischer Systeme, bei dem die Systemfunktionen schrittweise umgeschrieben und ersetzt werden, bis das Legacy-System außer Betrieb genommen werden kann. Dieses Muster verwendet die Analogie einer Feigenrebe, die zu einem etablierten Baum heranwächst und schließlich ihren Wirt überwindet und ersetzt. Das Muster wurde eingeführt von Martin Fowler als Möglichkeit, Risiken beim Umschreiben

monolithischer Systeme zu managen. Ein Beispiel für die Anwendung dieses Musters finden Sie unter Schrittweises Modernisieren älterer Microsoft ASP.NET (ASMX)-Webservices mithilfe von Containern und Amazon API Gateway.

Subnetz

Ein Bereich von IP-Adressen in Ihrer VPC. Ein Subnetz muss sich in einer einzigen Availability Zone befinden.

Aufsichtskontrolle und Datenerfassung (SCADA)

In der Fertigung ein System, das Hardware und Software zur Überwachung von Sachanlagen und Produktionsabläufen verwendet.

Symmetrische Verschlüsselung

Ein Verschlüsselungsalgorithmus, der denselben Schlüssel zum Verschlüsseln und Entschlüsseln der Daten verwendet.

synthetisches Testen

Testen eines Systems auf eine Weise, die Benutzerinteraktionen simuliert, um potenzielle Probleme zu erkennen oder die Leistung zu überwachen. Sie können Amazon CloudWatch Synthetics verwenden, um diese Tests zu erstellen.

# T

tags

Schlüssel-Wert-Paare, die als Metadaten für die Organisation Ihrer Ressourcen dienen. AWS Mit Tags können Sie Ressourcen verwalten, identifizieren, organisieren, suchen und filtern. Weitere Informationen finden Sie unter Markieren Ihrer AWS -Ressourcen.

Zielvariable

Der Wert, den Sie in überwachtem ML vorhersagen möchten. Dies wird auch als Ergebnisvariable bezeichnet. In einer Fertigungsumgebung könnte die Zielvariable beispielsweise ein Produktfehler sein.

Aufgabenliste

Ein Tool, das verwendet wird, um den Fortschritt anhand eines Runbooks zu verfolgen. Eine Aufgabenliste enthält eine Übersicht über das Runbook und eine Liste mit allgemeinen Aufgaben,

die erledigt werden müssen. Für jede allgemeine Aufgabe werden der geschätzte Zeitaufwand, der Eigentümer und der Fortschritt angegeben.

Testumgebungen

Siehe Umgebung.

Training

Daten für Ihr ML-Modell bereitstellen, aus denen es lernen kann. Die Trainingsdaten müssen die richtige Antwort enthalten. Der Lernalgorithmus findet Muster in den Trainingsdaten, die die Attribute der Input-Daten dem Ziel (die Antwort, die Sie voraussagen möchten) zuordnen. Es gibt ein ML-Modell aus, das diese Muster erfasst. Sie können dann das ML-Modell verwenden, um Voraussagen für neue Daten zu erhalten, bei denen Sie das Ziel nicht kennen.

Transit-Gateway

Ein Transit-Gateway ist ein Netzwerk-Transit-Hub, mit dem Sie Ihre VPCs und On-Premises-Netzwerke miteinander verbinden können. Weitere Informationen finden Sie in der AWS Transit Gateway Dokumentation unter Was ist ein Transit-Gateway.

Stammbasierter Workflow

Ein Ansatz, bei dem Entwickler Feature lokal in einem Feature-Zweig erstellen und testen und diese Änderungen dann im Hauptzweig zusammenführen. Der Hauptzweig wird dann sequentiell für die Entwicklungs-, Vorproduktions- und Produktionsumgebungen erstellt.

Vertrauenswürdiger Zugriff

Gewährung von Berechtigungen für einen Dienst, den Sie angeben, um Aufgaben in Ihrer Organisation AWS Organizations und in deren Konten in Ihrem Namen auszuführen. Der vertrauenswürdige Service erstellt in jedem Konto eine mit dem Service verknüpfte Rolle, wenn diese Rolle benötigt wird, um Verwaltungsaufgaben für Sie auszuführen. Weitere Informationen finden Sie in der AWS Organizations Dokumentation unter Verwendung AWS Organizations mit anderen AWS Diensten.

Optimieren

Aspekte Ihres Trainingsprozesses ändern, um die Genauigkeit des ML-Modells zu verbessern. Sie können das ML-Modell z. B. trainieren, indem Sie einen Beschriftungssatz generieren, Beschriftungen hinzufügen und diese Schritte dann mehrmals unter verschiedenen Einstellungen wiederholen, um das Modell zu optimieren.

Zwei-Pizzen-Team

Ein kleines DevOps Team, das Sie mit zwei Pizzen ernähren können. Eine Teamgröße von zwei Pizzen gewährleistet die bestmögliche Gelegenheit zur Zusammenarbeit bei der Softwareentwicklung.

# U

Unsicherheit

Ein Konzept, das sich auf ungenaue, unvollständige oder unbekannte Informationen bezieht, die die Zuverlässigkeit von prädiktiven ML-Modellen untergraben können. Es gibt zwei Arten von Unsicherheit: Epistemische Unsicherheit wird durch begrenzte, unvollständige Daten verursacht, wohingegen aleatorische Unsicherheit durch Rauschen und Randomisierung verursacht wird, die in den Daten liegt. Weitere Informationen finden Sie im Leitfaden Quantifizieren der Unsicherheit in Deep-Learning-Systemen.

undifferenzierte Aufgaben

Diese Arbeit wird auch als Schwerstarbeit bezeichnet. Dabei handelt es sich um Arbeiten, die zwar für die Erstellung und den Betrieb einer Anwendung erforderlich sind, aber dem Endbenutzer keinen direkten Mehrwert bieten oder keinen Wettbewerbsvorteil bieten. Beispiele für undifferenzierte Aufgaben sind Beschaffung, Wartung und Kapazitätsplanung.

höhere Umgebungen

Siehe Umgebung.

# V

Vacuuming

Ein Vorgang zur Datenbankwartung, bei dem die Datenbank nach inkrementellen Aktualisierungen bereinigt wird, um Speicherplatz zurückzugewinnen und die Leistung zu verbessern.

Versionskontrolle

Prozesse und Tools zur Nachverfolgung von Änderungen, z. B. Änderungen am Quellcode in einem Repository.

VPC-Peering

Eine Verbindung zwischen zwei VPCs, mit der Sie den Datenverkehr mithilfe von privaten IP-Adressen weiterleiten können. Weitere Informationen finden Sie unter Was ist VPC-Peering? in der Amazon-VPC-Dokumentation.

Schwachstelle

Ein Software- oder Hardwarefehler, der die Sicherheit des Systems gefährdet.

# W

Warmer Cache

Ein Puffer-Cache, der aktuelle, relevante Daten enthält, auf die häufig zugegriffen wird. Die Datenbank-Instance kann aus dem Puffer-Cache lesen, was schneller ist als das Lesen aus dem Hauptspeicher oder von der Festplatte.

warme Daten

Daten, auf die selten zugegriffen wird. Bei der Abfrage dieser Art von Daten sind mäßig langsame Abfragen in der Regel akzeptabel.

Fensterfunktion

Eine SQL-Funktion, die eine Berechnung für eine Gruppe von Zeilen durchführt, die sich in irgendeiner Weise auf den aktuellen Datensatz beziehen. Fensterfunktionen sind nützlich für die Verarbeitung von Aufgaben wie die Berechnung eines gleitenden Durchschnitts oder für den Zugriff auf den Wert von Zeilen auf der Grundlage der relativen Position der aktuellen Zeile.

Workload

Ein Workload ist eine Sammlung von Ressourcen und Code, die einen Unternehmenswert bietet, wie z. B. eine kundenorientierte Anwendung oder ein Backend-Prozess.

Workstream

Funktionsgruppen in einem Migrationsprojekt, die für eine bestimmte Reihe von Aufgaben verantwortlich sind. Jeder Workstream ist unabhängig, unterstützt aber die anderen Workstreams im Projekt. Der Portfolio-Workstream ist beispielsweise für die Priorisierung von Anwendungen, die Wellenplanung und die Erfassung von Migrationsmetadaten verantwortlich. Der Portfolio-Workstream liefert diese Komponenten an den Migrations-Workstream, der dann die Server und Anwendungen migriert.

WURM

Sehen Sie einmal schreiben, viele lesen.

WQF

Weitere Informationen finden Sie unter AWS Workload Qualification Framework.

einmal schreiben, viele lesen (WORM)

Ein Speichermodell, das Daten ein einziges Mal schreibt und verhindert, dass die Daten gelöscht oder geändert werden. Autorisierte Benutzer können die Daten so oft wie nötig lesen, aber sie können sie nicht ändern. Diese Datenspeicherinfrastruktur gilt als unveränderlich.

# Z

Zero-Day-Exploit

Ein Angriff, in der Regel Malware, der eine Zero-Day-Sicherheitslücke ausnutzt.

Zero-Day-Sicherheitslücke

Ein unfehlbarer Fehler oder eine Sicherheitslücke in einem Produktionssystem. Bedrohungsakteure können diese Art von Sicherheitslücke nutzen, um das System anzugreifen. Entwickler werden aufgrund des Angriffs häufig auf die Sicherheitsanfälligkeit aufmerksam.

Zombie-Anwendung

Eine Anwendung, deren durchschnittliche CPU- und Arbeitsspeichernutzung unter 5 Prozent liegt. In einem Migrationsprojekt ist es üblich, diese Anwendungen außer Betrieb zu nehmen.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.