Optimieren der SQL Postgre-Parameter in Amazon RDS und Amazon Aurora

AWS Präskriptive Leitlinien

AWS Präskriptive Leitlinien: Optimieren der SQL Postgre-Parameter in Amazon RDS und Amazon Aurora

Table of Contents

Optimieren von PostgreSQL-Parametern in Amazon RDS und Amazon Aurora

Sumana Yanamandra, Ramu Jagini und Rohit Kapoor, Amazon Web Services ()AWS

Februar 2024 (Geschichte der Dokumente)

Amazon Aurora PostgreSQL-Compatible Edition und Amazon Relational Database Service (Amazon RDS) für PostgreSQL sind hochentwickelte relationale Open-Source-Datenbankdienste, die eine breite Palette von Funktionen bieten. Sie können diese Dienste verwenden, um Ihre PostgreSQL-Datenbank auf einer Vielzahl von Plattformen und Anwendungen einzurichten.

Aurora und Amazon RDS bieten eine vereinfachte Möglichkeit, Ihre PostgreSQL-Datenbanken zu verwalten und zu betreiben. Sie sind für die Verwaltung der Datenbankinfrastruktur konzipiert und bieten hohe Verfügbarkeit, Haltbarkeit und Skalierbarkeit, während Sie sich auf die Anwendungsentwicklung konzentrieren können. Die Standardkonfigurationen dieser Dienste sind jedoch möglicherweise nicht für alle Workloads optimal. Standardmäßig sind diese Dienste so konfiguriert, dass sie überall mit möglichst wenigen Ressourcen und ohne Sicherheitslücken ausgeführt werden können. Durch die Optimierung der Parameter können Sie eine bessere Leistung erzielen, Ausfallzeiten reduzieren und die allgemeine Datenbankeffizienz verbessern. Durch die Optimierung der Parameter für Ihren spezifischen Workload können Sie die Funktionen von Amazon RDS und Aurora voll ausschöpfen und ihre Vorteile maximieren.

Sie können beispielsweise die Leistung verbessern, indem Sie Aurora und Amazon RDS for PostgreSQL optimieren und deren Parameter konfigurieren. Sie sollten auch die Leistung berücksichtigen, wenn Sie Datenbankabfragen erstellen. Selbst wenn Sie Datenbankeinstellungen optimieren, kann das System schlecht abschneiden, wenn Ihre Abfragen vollständige Tabellenscans durchführen, wenn sie einen Index verwenden oder wenn sie teure Join- oder Aggregationsoperationen ausführen.

Dieses Handbuch richtet sich an Datenbankentwickler, Datenbankingenieure und Administratoren, die Speicher-, Autovacuum-, Protokollierungs- und logische Replikationsparameter für ihre PostgreSQL-Datenbanken optimieren möchten. Das Handbuch behandelt auch Parameter, die spezifisch für Amazon RDS for PostgreSQL und Aurora PostgreSQL Compatible sind. Durch die Optimierung dieser Parameter können Sie die Datenbankleistung optimieren und den Ressourcenverbrauch für Ihre spezifische Arbeitslast reduzieren, was zu einer besseren Leistung und Kosteneinsparungen führt.

Verwenden von DB- und DB-Cluster-Parametergruppen

Amazon RDS und Aurora können basierend auf der Größe Ihrer Datenbank-Instance automatisch die am besten geeigneten Parameterwerte für bestimmte Einstellungen ermitteln. Sie unterstützen auch die Anpassung von Parametern zur Leistungsoptimierung mithilfe von Parametergruppen für Datenbank-Instances und Cluster.

Sie können DB- und DB-Cluster-Parametergruppen verwenden, um die Parameter zu ändern, die verschiedene Aspekte des Verhaltens der Datenbank-Engine steuern, z. B. Speichernutzung, Festplatten-I/O, Netzwerke und Sperren. Durch die Anpassung dieser Parameter können Sie die Datenbank-Engine für Ihre spezifische Arbeitslast optimieren und die Leistung verbessern.

Sie können DB- und DB-Cluster-Parametergruppen mithilfe der AWS Management Console, der AWS Command Line Interface (AWS CLI) oder der Amazon RDS-API erstellen und konfigurieren. In diesem Handbuch wird davon ausgegangen, dass Sie die verwenden AWS CLI. Anweisungen zur Konsole und zur API finden Sie unter Arbeiten mit DB-Parametergruppen und Arbeiten mit DB-Cluster-Parametergruppen in der Amazon RDS-Dokumentation.

> **Important**
>
> Um die AWS CLI Befehle in diesem Handbuch verwenden zu können, müssen Sie zuerst den installieren und konfigurieren AWS CLI.

Um eine DB-Parametergruppe zu erstellen und zu konfigurieren:

```
# Create a new DB parameter group
aws rds create-db-parameter-group \
  --db-parameter-group-name mydbparamgroup \
  --db-parameter-group-family postgres13 \
  --description "My DB Parameter Group"

# Modify a parameter on the DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <param group name> \
  --parameters "ParameterName=max_connections<parameter-
name>,ParameterValue=<value>,ApplyMethod=immediate"

# Verify DB parameters
```

```
aws rds describe-db-parameters \
  --db-parameter-group-name aurora-instance-1
```

So erstellen und konfigurieren Sie eine DB-Cluster-Parametergruppe:

```
# Create a new DB cluster parameter group
aws rds create-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --db-parameter-group-family postgres12 \
  --description "My new parameter group"

# Modify a parameter on the DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name aws-guide-cluster \
  --parameters "ParameterName=<parameter-name>,ParameterValue=,ApplyMethod=immediate"

# Allocate the new DB cluster parameter to your cluster
aws rds modify-db-cluster  \
  --db-cluster-identifier  \
  --db-cluster-parameter-group-name=-cluster

# Verify cluster parameters
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name=-cluster
```

> **Note**
>
> Aurora und Amazon RDS bieten eine Standardparametergruppe mit vorkonfigurierten Werten, die nicht geändert werden können.

Parametergruppen können als statisch oder dynamisch festgelegt werden. Dynamische Parameter werden sofort angewendet, unabhängig davon, ob die `ApplyMethod=immediate` Option aktiviert ist. Statische Parameter erfordern einen manuellen Neustart, um wirksam zu werden.

Speicherparameter optimieren

Die Optimierung der Speicherparameter ist eine wichtige Aufgabe zur Optimierung der Leistung von Amazon RDS - und Aurora SQL Postgre-kompatiblen Datenbanken. Die korrekte Speicherzuweisung für verschiedene Datenbankoperationen wie das Ausführen von Abfragen, Sortieren, Indizieren und Zwischenspeichern kann die Datenbankleistung erheblich verbessern. In diesem Abschnitt werden einige der wichtigsten Speicherparameter in Amazon RDS und Aurora SQL Postgre-Compatible behandelt, einschließlich ihrer Standardwerte, Formeln zur Berechnung geeigneter Werte und deren Änderung. Eine vollständige Liste der Parameter finden Sie unter Arbeiten mit Parametern auf Ihrer RDS for SQL Postgre-DB-Instance in der RDS Amazon-Dokumentation und unter Amazon Aurora SQL Postgre-Parametern in der Aurora-Dokumentation.

Die Optimierung dieser Parameter erfordert ein tiefes Verständnis Ihrer Datenbank-Arbeitslast sowie der Ressourcen, die auf Ihrer Aurora- oder RDS Amazon-DB-Instance verfügbar sind. Die Systemleistung wird von zwei großen Kategorien von Parametern beeinflusst: Vitalparameter und Eventualparameter.

Vitalparameter sind unverzichtbare Parameter, die einen erheblichen und direkten Einfluss auf die Leistung des Systems haben und für die Erzielung optimaler Ergebnisse unerlässlich sind.

- shared_buffers
- temp_buffers
- effektive_Cachegröße
- work_mem
- maintenance_work_mem

Kontingente Parameter sind szenario- und geschäftsspezifische Parameter. Sie hängen von anderen Faktoren ab und spielen eine indirekte, aber entscheidende Rolle bei der Unterstützung wichtiger Parameter und der Maximierung der Gesamtsystemleistung.

- random_page_cost
- seq_page_cost
- Größe der Track_Activity_Query_Size
- Zeitlimit für in_transaction_session_idle_in_transaction_session

- Timeout für die Anweisung
- Suchpfad
- max_Verbindungen

Diese Parameter werden in den folgenden Abschnitten ausführlicher behandelt.

shared_buffers

Der `shared_buffers` Parameter steuert die Speichermenge, die Postgre SQL verwendet, um Daten im Speicher zwischenzuspeichern. Wenn Sie diesen Parameter auf einen geeigneten Wert setzen, kann dies zur Verbesserung der Abfrageleistung beitragen.

Für Amazon RDS `shared_buffers` ist der Standardwert für auf `{DBInstanceClassMemory/32768}` Byte festgelegt, basierend auf dem verfügbaren Speicher für die DB-Instance. Für Aurora ist der Standardwert auf festgelegt`{DBInstanceClassMemory/12038,-50003}`, basierend auf dem verfügbaren Speicher für die DB-Instance. Der optimale Wert für diesen Parameter hängt von mehreren Faktoren ab, darunter der Größe der Datenbank, der Anzahl gleichzeitiger Verbindungen und dem verfügbaren Instance-Speicher.

> **Note**
>
> Wenn Sie eine DB-Instance herunterskalieren, stellen Sie sicher, dass sie `shared_buffers` an die neuen Speichergrenzen angepasst wird. Andernfalls kann Postgre SQL möglicherweise nicht gestartet werden oder es könnten Probleme mit der SQL Postgre-Engine auftreten. Dokumentieren Sie alle Änderungen, um future Anpassungen zu erleichtern.

AWS CLI Syntax

Der folgende Befehl ändert sich `shared_buffers` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify shared_buffers on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
```

```
  --parameters
 "ParameterName=shared_buffers,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify shared_buffers on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters "ParameterName=shared_buffers,ParameterValue=<new-
value>,ApplyMethod=immediate"
```

Typ: Statisch (das Anwenden von Änderungen erfordert einen Neustart)

Standardwert: `{DBInstanceClassMemory/32768}` Bytes in Amazon RDS für PostgreSQL, `{DBInstanceClassMemory/12038,-50003}` in Aurora Postgre SQL -Compatible. In den meisten Fällen entspricht diese Gleichung etwa 25 Prozent Ihres Systemspeichers. Gemäß dieser Richtlinie wird die `shared_buffers` Einstellung in der Parametergruppe mithilfe der SQL Postgre-Standardeinheiten von 8K-Puffern anstelle von Byte oder Kilobyte eingerichtet.

Die `shared_buffers` Parametereinstellung kann erhebliche Auswirkungen auf die Leistung haben. Wir empfehlen Ihnen daher, Ihre Änderungen gründlich zu testen, um sicherzustellen, dass der Wert für Ihre Arbeitslast geeignet ist.

Beispiel

Nehmen wir an, Sie haben eine Finanzdienstleistungsanwendung, die eine SQL Postgre-Datenbank auf Amazon RDS oder Aurora ausführt. Diese Datenbank wird zum Speichern von Kundentransaktionsdaten verwendet. Sie hat eine große Anzahl von Tabellen und wird von mehreren Anwendungen auf einer großen Anzahl von Servern aufgerufen. Die Anwendung weist eine langsame Abfrageleistung und eine hohe CPU Auslastung auf. Sie stellen fest, dass die Optimierung des `shared_buffers` Parameters zur Verbesserung der Leistung beitragen kann.

In Amazon RDS for Postgre SQL `shared_buffers` ist der Standardwert auf `{DBInstanceClassMemory/32768}` Byte des verfügbaren Speichers in `db.r5.xlarge` (z. B. 3 GB) festgelegt. Um den geeigneten Wert für zu ermittelns`hared_buffers`, führen Sie eine Reihe von Tests mit unterschiedlichen Werten von durchs`hared_buffers`, wobei Sie mit dem Standardwert für verfügbaren Speicher beginnen und den Wert schrittweise erhöhen. Für jeden Test messen Sie die Abfrageleistung und die CPU Nutzung der Datenbank.

Auf der Grundlage der Testergebnisse stellen Sie fest, dass die Einstellung des Werts `shared_buffers` auf 8 GB die beste allgemeine Abfrageleistung und CPU Nutzung für Ihren

Workload ergibt. Der Wert wird durch eine Kombination aus Tests und Analysen von Workload-Merkmalen bestimmt, darunter die Größe der Datenbank, die Anzahl und Komplexität der Abfragen, die Anzahl der gleichzeitigen Benutzer und die verfügbaren Systemressourcen. Nachdem Sie die Änderung vorgenommen haben, überprüfen Ihre Überwachungssysteme die Leistung der Datenbank, um sicherzustellen, dass der neue Wert für Ihre Arbeitslast geeignet ist. Anschließend können Sie weitere Parameter nach Bedarf anpassen, um die Leistung weiter zu verbessern.

temp_buffers

`temp_buffers`ist ein wichtiger Konfigurationsparameter in Aurora Postgre SQL -Compatible und Amazon RDS for PostgreSQL, der die Leistung von Workloads, die Sortier-, Hashes- und Aggregationsoperationen in temporären Tabellen beinhalten, erheblich beeinträchtigen kann. Dieser Parameter bestimmt die Menge an Speicher, die temporären Puffern zugewiesen wird, was sich wiederum auf die Effizienz und Geschwindigkeit solcher Operationen auswirkt. Wenn nicht genügend Speicher zugewiesen ist, muss das System möglicherweise langsamere`temp_buffers`, weniger effiziente Methoden für Sortier-, Hashes- und Aggregationsoperationen für temporäre Tabellen verwenden, was zu einer suboptimalen Leistung führt.

AWS CLI Syntax

Der folgende Befehl ändert sich `temp_buffers` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify temp_buffers on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=temp_buffers,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify temp_buffers on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=temp_buffers,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 8 MB

Weitere Informationen zu diesem Parameter finden Sie unter Ressourcenverbrauch in der SQL Postgre-Dokumentation.

Beispiel

Wenn Ihre Arbeitslast viele Sortier-, Hashing- und Aggregationsoperationen für temporäre Tabellen umfasst, wird `temp_buffers` möglicherweise nicht genügend Speicher zugewiesen. In diesem Fall muss das System möglicherweise Sortieroperationen an temporären Tabellen durchführen, die zu langsameren, festplattenbasierten Methoden führen, anstatt speicherinterne Sortier-, Hashing- und Aggregationsoperationen durchzuführen. Dies kann zu einer erheblichen Verlangsamung der Abfrageleistung führen, insbesondere bei Abfragen mit großen Datensätzen. Durch eine Erhöhung des Werts von `temp_buffers` kann sichergestellt werden, dass genügend Arbeitsspeicher für die Ausführung solcher Operationen im Arbeitsspeicher verfügbar ist, was zu einer deutlichen Leistungsverbesserung führt.

Um den optimalen Wert von zu ermittelntemp_buffers, überwachen Sie Ihre Systemleistung und identifizieren Sie Bereiche mit suboptimaler Leistung. Wenn Sie langsame Antwortzeiten bei Anfragen oder eine hohe CPU Auslastung feststellen, sollten Sie eine Anpassung `temp_buffers` in Betracht ziehen. Wenn Ihre Arbeitslast beispielsweise viele temporäre Tabellen umfasst, `temp_buffers` kann eine Erhöhung des Werts von dazu beitragen, dass diese Tabellen im Arbeitsspeicher gespeichert werden. Dies kann viel schneller sein als die Verwendung von I/O mit Lese-/Schreibzugriff aus dem Speicher.

Experimentieren Sie mit verschiedenen Werten `temp_buffers` in kleinen Schritten und überwachen Sie die Systemleistung nach jeder Änderung sorgfältig. Analysieren Sie die Auswirkungen verschiedener Werte auf die Leistung und passen Sie die Einstellung anhand der spezifischen Merkmale Ihrer Arbeitslast an.

effective_cache_size

Der `effective_cache_size` Parameter gibt die Speichermenge an, von der Postgre annehmen SQL soll, dass sie für das Zwischenspeichern von Daten verfügbar ist. Durch die korrekte Einstellung dieses Parameters kann die Leistung verbessert werden, da Postgre SQL den verfügbaren Speicher besser nutzen kann.

AWS CLI Syntax

Der folgende Befehl ändert sich `effective_cache_size` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify effective_cache_size on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=effective_cache_size,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify effective_cache_size on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=effective_cache_size,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `SUM(DBInstanceClassMemory/12038,-50003)` KB

Beispiel

Eine Online-Lernplattform verfügt über eine große Datenbank mit Kursmaterialien, Studentendaten und anderen Inhalten, auf die Benutzer häufig zugreifen. Die Anwendung läuft auf einer Amazon RDS for SQL `db.r5.xlarge` Postgre-Instance mit 32 GB Arbeitsspeicher. Die Leistung der Anwendung verlangsamt sich, wenn Benutzer versuchen, häufig aufgerufene Inhalte zu lesen. Nach der Analyse der Ressourcennutzung des Datenbankservers stellen Sie fest, dass Postgre den verfügbaren Speicher SQL nicht optimal nutzt.

Der `effective_cache_size` Parameter in Amazon RDS for Postgre SQL steuert die Speichermenge, die der Server für das Festplatten-Caching verwendet. Der Standardwert ist für die Instance-Klasse auf `SUM({DBInstanceClassMemory/12038},-50003)` KB festgelegt`db.r5.xlarge`, aber dieser Standardwert ist möglicherweise nicht für alle Workloads geeignet. In diesem Beispiel speichert der Datenbankserver möglicherweise große Mengen an Kursmaterialien und Studentendaten, auf die häufig zugegriffen wird. Eine Erhöhung des `effective_cache_size` Parameterwerts kann dazu führen, dass mehr Daten im Arbeitsspeicher zwischengespeichert werden, wodurch die Anzahl der erforderlichen Festplattenlesevorgänge reduziert und die Abfrageleistung verbessert wird.

Wenn Sie eine Abfrage ausführen, prüft Amazon RDS for Postgre SQL zunächst, ob sich die für die Abfrage erforderlichen Daten bereits im Cache befinden. Wenn ja, können die Daten aus dem Speicher gelesen werden, anstatt von der Festplatte gelesen zu werden. Wenn sich die Daten nicht

im Cache befinden, müssen sie von der Festplatte gelesen werden, was ein langsamer Vorgang sein kann.

Für die Online-Lernplattform können Sie sich nach dem Testen und Analysieren für 16 GB (die Hälfte des verfügbaren Speichers) entscheiden. `effective_cache_size` Dieser Wert ermöglicht es PostgreSQL, den verfügbaren Speicher besser zu nutzen, wodurch die Anzahl der erforderlichen Festplattenlesevorgänge reduziert und die Abfrageleistung verbessert wird.

work_mem

Der `work_mem` Parameter steuert die Speichermenge, die von Abfragen für Sortier- und Hashing-Operationen verwendet wird. Der Standardwert ist 4 MB. Wenn eine Abfrage mehrere Operationen umfasst, kann sie für jeden Vorgang bis zu 4 MB verwenden. Eine Erhöhung des Werts von `work_mem` kann die Leistung von Abfragen verbessern, die Sortierung oder Hashing erfordern, da diese Operationen mehr Speicher benötigen. Wenn dieser Parameter jedoch zu hoch eingestellt wird, kann dies zu einer übermäßigen Speicherauslastung und damit zu Leistungseinbußen führen.

Um den optimalen Wert für zu berechnen`work_mem`, können Sie die folgende Formel verwenden:

```
work_mem = (available_memory / (max_connections * work_mem_fraction))
```

wobei `available_memory` die Gesamtmenge des auf dem Server verfügbaren Speichers, max_connections die maximal zulässige Anzahl von Verbindungen und work_mem_fraction ein Bruchteil ist, der bestimmt, wie viel des verfügbaren Speichers jeder Verbindung zugewiesen werden soll.

AWS CLI Syntax

Der folgende Befehl ändert sich `work_mem` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify work_mem on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify work_mem on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
```

```
    --db-cluster-parameter-group-name <parameter_group_name> \
    --parameters
 "ParameterName=work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 4 MB

Beispiel

Ein Analysetool für soziale Medien verarbeitet große Datenmengen, und Abfragen, die komplexe Sortier- und Verbindungsvorgänge beinhalten, führen zu einer hohen Festplatten-E/A-Auslastung und Datenverlust auf die Festplatte. Wenn Sie den Wert `work_mem` von 4 MB auf 16 MB erhöhen, SQL kann Postgre mehr Speicher für diese Operationen verwenden. Dadurch wird der I/O-Aufwand reduziert und die Abfrageleistung verbessert.

maintenance_work_mem

Der `maintenance_work_mem` Parameter steuert die Speichermenge, die für Wartungsvorgänge wie, und die Indexerstellung verwendet wird. `VACUUM` `ANALYZE` Der Standardwert für diesen Parameter in Amazon RDS und Aurora ist 64 MB.

Um den entsprechenden Wert für diesen Parameter zu berechnen, können Sie diese Formel verwenden:

```
maintenance_work_mem = (total_memory - shared_buffers) / (max_connections * 5)
```

Aurora Postgre SQL -Compatible Edition und Amazon RDS for Postgre SQL wenden die folgende Formel an, um den optimalen Wert festzulegen:

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

AWS CLI Syntax

Der folgende Befehl ändert sich `maintenance_work_mem` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify maintenance_work_mem on a DB parameter group
```

```
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=maintenance_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify maintenance_work_mem on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=maintenance_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 64 MB

Beispiel

Ihre umfangreiche Anwendung verwendet eine SQL Postgre-Datenbank, die auf Aurora oder Amazon RDS gehostet wird. Sie stellen fest, dass die Datenbank langsam ist und bei Wartungsarbeiten wie dem Löschen und Indizieren nicht reagiert. Sie können Messwerte wie Speicherverbrauch, Wartungsbetriebszeiten und CPU Auslastung überwachen, um festzustellen, ob der aktuelle Wert von Probleme verursacht. `maintenance_work_mem`

Um den optimalen Wert für zu ermitteln`maintenance_work_mem`, können Sie den Parameter anpassen und seine Auswirkungen überwachen. Wenn der Speicherverbrauch konstant hoch ist oder die Betriebszeiten bei Wartungsarbeiten länger als erwartet sind, kann eine Erhöhung des Speichers `maintenance_work_mem` hilfreich sein. Umgekehrt, wenn die CPU Auslastung während Wartungsarbeiten konstant hoch ist, kann eine Verringerung `maintenance_work_mem` hilfreich sein. Durch iterative Anpassungen und Tests können Sie den optimalen Wert finden, der `maintenance_work_mem` das beste Gleichgewicht zwischen Speichernutzung, Wartungsbetriebszeiten und CPU Auslastung bietet.

Nehmen wir an, Sie stellen während Ihrer Untersuchung fest, dass der Standardwert von 64 MB für zu niedrig für Ihre Datenbankgröße `maintenance_work_mem` ist. Das hat zur Folge, dass Wartungsarbeiten länger dauern, zu übermäßigen Ausfallzeiten führen und die Leistung Ihrer Anwendung beeinträchtigen. Um dieses Problem zu beheben, können Sie den `maintenance_work_mem` Parameter anpassen, indem Sie ihn von 64 MB auf 512 MB erhöhen (was Sie als optimalen Wert angeben). Durch die Änderung können Sie Ihre Wartungszeiten um zwei Drittel verbessern. Beispielsweise dauert ein Vakuumvorgang, dessen Abschluss zuvor 30 Minuten

gedauert hat, jetzt möglicherweise nur noch 10 Minuten. Als Ergebnis dieser Optimierung kann Ihre Datenbank Wartungsaktivitäten jetzt effizienter abwickeln.

random_page_cost

Der `random_page_cost` Parameter hilft bei der Bestimmung der Kosten für den zufälligen Seitenzugriff. Der Abfrageplaner in Amazon RDS und Aurora verwendet diesen Parameter zusammen mit anderen Statistiken über die Tabelle, um den effizientesten Plan für die Ausführung einer Abfrage zu ermitteln.

Die `random_page_cost` Parameter `seq_page_cost` und die Parameter sind eng miteinander verknüpft und werden vom Planer normalerweise zusammen verwendet, um die Kosten verschiedener Zugriffsmethoden zu vergleichen und zu entscheiden, welche die effizienteste ist. Wenn Sie also einen dieser Parameter ändern, sollten Sie auch überlegen, ob der andere Parameter angepasst werden muss.

Im Allgemeinen versucht der Abfrageplaner, die Kosten für die Ausführung einer Abfrage zu minimieren. Die Kosten werden anhand einer Kombination aus der Anzahl der Lesevorgänge auf der Festplatte und dem Wert von ermitteltrandom_page_cost. Ein höherer Wert von `random_page_cost` begünstigt tendenziell sequentielle Scans, wohingegen ein niedrigerer Wert eher Indexscans begünstigt. Ein niedrigerer Wert bevorzugt tendenziell auch Nested-Loop-Joins anstelle von Hash-Joins.

Der `random_page_cost` Parameter verwendet den standardmäßigen SQL Postgre-Engine-Wert (4), sofern in der Parametergruppe oder in der lokalen Sitzung kein Wert festgelegt ist. Sie können diesen Wert je nach den spezifischen Eigenschaften Ihres Servers und Ihrer Arbeitslast anpassen. Wenn die meisten der in Ihrem Workload verwendeten Indizes in den Arbeitsspeicher oder in den mehrstufigen Aurora-Cache passen, `seq_page_cost` ist es angemessen, den Wert von `random_page_cost` auf einen Wert zu ändern, der nahe am

AWS CLI Syntax

Der folgende Befehl ändert sich `random_page_cost` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify random_page_cost on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
```

```
  --parameters
 "ParameterName=random_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify random_page_cost on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=random_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 4

Beispiel

Nehmen wir an, Sie haben eine Datenbank, die eine große Datenmenge in einer Tabelle speichert, die häufig mit Filtern für nicht indizierte Spalten abgefragt wird. Das Abschließen der Abfragen nimmt viel Zeit in Anspruch, und der Abfrageplaner wählt nicht den effizientesten Plan für den Zugriff auf die Daten aus.

Eine Möglichkeit, die Leistung zu verbessern, besteht darin, den `random_page_cost` Parameter zu verringern. Wenn Sie ihn auf 1 setzen, wären die Kosten für den zufälligen Seitenzugriff viermal günstiger als der Standardwert. Wenn Sie den Standardwert 4 beibehalten `random_page_cost` würden, wäre der zufällige Seitenzugriff viermal teurer als der sequentielle Seitenzugriff (wie durch den `seq_page_cost` Parameter bestimmt, der standardmäßig 1,0 ist). In diesem speziellen Fall könnte der zufällige Seitenzugriff jedoch je nach Speichertyp tatsächlich viel teurer sein.

Wenn Sie den Wert des `random_page_cost` Parameters verringern, ist es wahrscheinlicher, dass der Abfrageplaner einen indexbasierten Plan auswählt oder eine andere Zugriffsmethode verwendet, die besser zu den spezifischen Eigenschaften der Tabelle passt.

Es wird empfohlen, die Abfrageleistung zu überwachen, nachdem Sie den Parameter geändert haben, und gegebenenfalls Anpassungen vorzunehmen. Sie sollten auch den Abfrageplaner mit einer `EXPLAIN` Anweisung überprüfen, um zu überprüfen, ob er einen effizienten Plan auswählt.

Dies ist nur ein Beispiel. Die optimale Einstellung hängt von den spezifischen Merkmalen Ihrer Arbeitslast ab. Außerdem ist dies nur ein Aspekt der Leistungsoptimierung. Sie sollten auch andere Parameter und Konfigurationsoptionen berücksichtigen, die sich auf Ihre Abfrageleistung auswirken können.

seq_page_cost

Der `seq_page_cost` Parameter hilft bei der Bestimmung der Kosten für den sequentiellen Seitenzugriff. Der Abfrageplaner in Amazon RDS und Aurora verwendet diesen Parameter zusammen mit anderen Statistiken über die Tabelle, um den effizientesten Plan für die Ausführung einer Abfrage zu ermitteln.

Die `random_page_cost` Parameter `seq_page_cost` und die Parameter sind eng miteinander verknüpft und werden vom Planer normalerweise zusammen verwendet, um die Kosten verschiedener Zugriffsmethoden zu vergleichen und zu entscheiden, welche die effizienteste ist. Wenn Sie also einen dieser Parameter ändern, sollten Sie auch überlegen, ob der andere Parameter angepasst werden muss.

Wenn sequentiell auf eine Tabelle zugegriffen wird, SQL kann Postgre den Dateisystem-Cache des Betriebssystems verwenden, um einen schnelleren Zugriff zu ermöglichen. `seq_page_cost`Ist standardmäßig auf 1.0 gesetzt, was davon ausgeht, dass sequentieller Seitenzugriff so billig ist wie das Lesen eines einzelnen Festplattenblocks. Wenn Sie über eine Tabelle verfügen, auf die primär sequentiell zugegriffen wird, der Festplattenzugriff jedoch langsam ist, weil er durch weniger begrenzt istIOPS, sollten Sie den Wert von erhöhen, um den zusätzlichen Kosten für den Zugriff auf die Festplatte Rechnung `seq_page_cost` zu tragen.

Eine Änderung des Werts dieses Parameters wirkt sich auf alle Abfragen aus, die im System ausgeführt werden. Wir empfehlen daher, Ihre Abfragen mit unterschiedlichen Werten zu testen, um den optimalen Wert für Ihren speziellen Anwendungsfall zu ermitteln.

AWS CLI Syntax

Der folgende Befehl ändert sich `seq_page_cost` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify seq_page_cost on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=seq_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify seq_page_cost on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
```

```
  --parameters
"ParameterName=seq_page_cost,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 1.0

Beispiel

Nehmen wir an, Sie haben eine Datenbank, die eine große Datenmenge in einer Tabelle speichert, auf die primär sequentiell zugegriffen wird. Die Tabelle wird hauptsächlich für Berichte verwendet, die Abfragen werden sehr langsam ausgeführt, und der Abfrageplaner ist nicht in der Lage, den effizientesten Plan für den Zugriff auf die Daten auszuwählen.

Eine Möglichkeit, die Leistung zu verbessern, besteht darin, den `seq_page_cost` Parameter zu verringern. Der Standardwert ist 1,0, was davon ausgeht, dass ein sequentieller Seitenzugriff genauso billig ist wie das Lesen eines einzelnen Festplattenblocks. In diesem speziellen Fall kann der sequentielle Seitenzugriff jedoch aufgrund des Speichertyps (abhängig von I/O) tatsächlich teurer sein. Wenn Sie den Wert `seq_page_cost` auf 0,5 festlegen, sind die Kosten für den sequentiellen Seitenzugriff halb so teuer wie der Standardwert. Durch diese Änderung kann es wahrscheinlicher sein, dass der Abfrageplaner einen Plan auswählt, der eine sequentielle Zugriffsmethode verwendet, die besser für die spezifischen Eigenschaften der Tabelle geeignet ist.

Es wird empfohlen, die Abfrageleistung zu überwachen, nachdem Sie den Parameter geändert haben, und gegebenenfalls Anpassungen vorzunehmen. Sie sollten den Abfrageplan auch anhand einer `EXPLAIN` Anweisung überprüfen, um zu überprüfen, ob er einen effizienten Plan auswählt.

Dies ist nur ein Beispiel. Die optimale Einstellung hängt von den spezifischen Merkmalen Ihrer Arbeitslast ab. Außerdem ist dies nur ein Aspekt der Leistungsoptimierung. Sie sollten auch andere Parameter und Konfigurationsoptionen berücksichtigen, die sich auf Ihre Abfrageleistung auswirken.

track_activity_query_size

Der `track_activity_query_size` Parameter steuert die Größe der Abfragezeichenfolge, die für jede aktive Sitzung in der Ansicht protokolliert wird. `pg_stat_activity` Standardmäßig werden nur die ersten 1.024 Byte der Abfragezeichenfolge in Amazon RDS für Postgre protokolliertSQL, und 4.096 Byte werden in Aurora Postgre -Compatible protokolliert. SQL Wenn Sie längere Abfragen protokollieren möchten, können Sie diesen Parameter auf einen höheren Wert setzen.

AWS CLI Syntax

Der folgende Befehl ändert sich `track_activity_query_size` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify track_activity_query_size on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=track_activity_query_size,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify track_activity_query_size on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=track_activity_query_size,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Statisch (das Anwenden von Änderungen erfordert einen Neustart)

Standardwert: 1.024 Byte (Amazon RDS für PostgreSQL), 4.096 Byte (Aurora Postgre -Compatible) SQL

Beispiel

Ihre Amazon RDS for SQL Postgre-Datenbank weist eine langsame Abfrageleistung auf, und Sie vermuten, dass das Problem mit Abfragen mit langer Laufzeit zusammenhängt. Sie können weitere Untersuchungen durchführen, indem Sie längere Abfragen in der `pg_stat_activity` Ansicht protokollieren.

Wenn Sie den Wert des `track_activity_query_size` Parameters erhöhen, kann dies zu einer erhöhten Protokollierung führen, was sich auf die Leistung der Datenbank auswirken kann. Es wird empfohlen, den Parameter auf den Standardwert 1.024 zurückzusetzen, nachdem das Problem behoben wurde.

idle_in_transaction_session_timeout

Der `idle_in_transaction_session_timeout` Parameter steuert, wie lange eine inaktive Transaktion wartet, bevor sie gestoppt wird.

Der Standardwert für diesen Parameter in Amazon RDS for Postgre SQL und Aurora Postgre SQL - Compatible beträgt 86.400.000 Millisekunden.

AWS CLI Syntax

Der folgende Befehl ändert sich `idle_in_transaction_session_timeout` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify idle_in_transaction_session_timeout on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=idle_in_transaction_session_timeout,ParameterValue=<new_value>,ApplyMethod=imme

# Modify idle_in_transaction_session_timeout on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=idle_in_transaction_session_timeout,ParameterValue=<new_value>,ApplyMethod=imme
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 86.400.000 Millisekunden (Aurora Postgre-Compatible) SQL

Beispiel

Sie haben eine E-Commerce-Anwendung, die Online-Bestellungen verarbeitet. Die Anwendung verwendet eine SQL Postgre-Datenbank, die auf Amazon RDS oder Aurora gehostet wird. Immer wenn ein Kunde eine Bestellung aufgibt, startet die Anwendung eine neue Transaktion, um das Inventar und die Bestelldaten zu aktualisieren.

Wenn eine Transaktion für längere Zeit inaktiv bleibt, kann dies verhindern, dass andere Transaktionen auf dieselben Datensätze zugreifen, was zu Leistungsproblemen und möglicherweise sogar zu Anwendungsausfällen führt. Darüber hinaus können inaktive Transaktionen, die nicht ordnungsgemäß gestoppt wurden, wertvolle Systemressourcen wie Arbeitsspeicher und verbrauchen`CPU`.

Um solche Probleme zu vermeiden, können Sie den `idle_in_transaction_session_timeout` Parameter auf einen Wert setzen, der für Ihre Anwendung sinnvoll ist. Sie können ihn beispielsweise

auf 5 Minuten (300 Sekunden) festlegen, sodass jede Transaktion, die länger als 5 Minuten inaktiv ist, automatisch gestoppt wird. Auf diese Weise kann sichergestellt werden, dass die Systemressourcen effizient genutzt werden und dass die Anwendung eine große Anzahl von Bestellungen ohne Leistungseinbußen verarbeiten kann. Indem Sie den entsprechenden Wert für festlegen`idle_in_transaction_session_timeout`, können Sie sicherstellen, dass Ihre Anwendung auf Amazon RDS oder Aurora optimal funktioniert.

statement_timeout

Der `statement_timeout` Parameter legt fest, wie lange eine Abfrage maximal ausgeführt werden kann, bevor sie gestoppt wird.

AWS CLI Syntax

Der folgende Befehl ändert sich `statement_timeout` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify statement_timeout on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=statement_timeout,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify statement_timeout on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=statement_timeout,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 0 Millisekunden (kein Timeout)

Beispiel

Ihre Webanwendung ermöglicht es Benutzern, eine große Produktdatenbank zu durchsuchen. Die Bearbeitung der Suchanfragen kann manchmal sehr lange dauern, was zu langsamen Antwortzeiten für Benutzer führt. Um dieses Problem zu beheben, könnten Sie den `statement_timeout`

Parameter auf einen niedrigen Wert setzen, z. B. auf 10 Sekunden, wodurch jede Abfrage, die länger als 10 Sekunden dauert, gestoppt wird.

Dies mag wie eine drastische Maßnahme erscheinen, kann aber tatsächlich sehr effektiv zur Leistungssteigerung beitragen. In vielen Fällen werden Abfragen mit langer Laufzeit durch schlecht optimierte SQL Abfragen oder ineffiziente Indizes verursacht. Wenn Sie einen niedrigen `statement_timeout` Wert festlegen, können Sie diese problematischen Abfragen identifizieren und Maßnahmen zu ihrer Optimierung ergreifen.

Nehmen wir beispielsweise an, Sie stellen fest, dass bei einer bestimmten Suchabfrage ständig ein Timeout auftritt. Sie können Tools wie `EXPLAIN` und verwenden, um die Abfrage `EXPLAIN ANALYZE` zu analysieren und etwaige Leistungsengpässe zu identifizieren. Wenn Sie das Problem identifiziert haben, können Sie Maßnahmen zur Optimierung der Abfrage ergreifen, indem Sie neue Indizes hinzufügen, die Abfrage neu schreiben oder einen anderen Suchalgorithmus verwenden. Indem Sie Ihre SQL Abfragen auf diese Weise kontinuierlich analysieren und optimieren, können Sie die Leistung Ihrer Anwendung erheblich verbessern.

search_path

Der `search_path` Parameter bestimmt die Reihenfolge, in der Schemas in Anweisungen nach Objekten durchsucht werden. SQL Der Standardwert ist`$user, public`, was bedeutet, dass Postgre zuerst in dem Schema, das dem Namen des Benutzers entspricht, und dann im öffentlichen Schema nach Objekten SQL sucht.

Wenn Sie über eine große Anzahl von Schemas verfügen oder auf Objekte in einem bestimmten Schema zugreifen müssen, kann eine Änderung des `search_path` Parameters zur Leistungssteigerung beitragen. Wenn Sie `search_path` auf ein bestimmtes Schema einstellen, SQL kann Postgre die Objekte schneller finden, ohne mehrere Schemas durchsuchen zu müssen.

Um den `search_path` Parameter in Amazon RDS und Aurora zu ändern, können Sie den folgenden Befehl für `ROLE` Level verwenden:

```
ALTER ROLE <username> SET search_path = <schema>;
```

AWS CLI Syntax

Der folgende Befehl ändert sich `search_path` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify search_path on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=search_path,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify search_path on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=search_path,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `$user, public`

Beispiel

Sie haben eine Mehrmandantenanwendung mit separaten Schemas für jeden Mandanten, der eine Amazon RDS for Postgre SQL - oder Aurora SQL Postgre-kompatible Datenbank verwendet, und Sie müssen eine Abfrage ausführen, bei der Daten aus mehreren Schemas zusammengeführt werden.

Standardmäßig verwenden Amazon RDS und Aurora den Suchpfad, um zu bestimmen, welches Schema für eine bestimmte Tabelle verwendet werden soll. Der Suchpfad ist eine Liste von Schemanamen, die Postgre der Reihe SQL nach durchsucht, wenn Sie auf eine Tabelle verweisen, ohne den Schemanamen zu qualifizieren. Standardmäßig suchen Amazon RDS und Aurora zuerst nach der Tabelle im Schema, die denselben Namen wie der aktuelle Benutzer hat, und suchen dann im öffentlichen Schema.

Nehmen wir an, Sie möchten eine Abfrage ausführen, bei der Tabellen aus mehreren Schemas mit den Namen `tenant1tenant2`, und `tenant3` verknüpft werden. Um die Mandantenschemas zu verwenden, können Sie die Schemanamen in die Abfrage einbeziehen:

```
SELECT *
FROM tenant1.table1
JOIN tenant2.table2 ON tenant1.table1.id = tenant2.table2.id
JOIN tenant3.table3 ON tenant2.table2.id = tenant3.table3.id;
```

Eine effizientere Methode besteht jedoch darin, den `search_path` Parameter so zu ändern, dass er die Mandantenschemas einschließt, indem Sie die Befehle im AWS CLI Syntaxabschnitt verwenden. Sie können den `SET` Befehl auch in einer Postgre-Sitzung verwenden: SQL

```
SET search_path = tenant1, tenant2, tenant3, public;
```

Sie können dann die Abfrage schreiben, ohne die Schemanamen zu qualifizieren:

```
SELECT *
FROM table1
JOIN table2 ON table1.id = table2.id
JOIN table3 ON table2.id = table3.id;
```

Dadurch kann Ihre Abfrage übersichtlicher und leichter lesbar werden, und es kann auch Ihren Anwendungscode vereinfachen, wenn Sie viele Abfragen haben, die das Verbinden von Tabellen aus mehreren Schemas beinhalten.

max_connections

Der `max_connections` Parameter legt die maximale Anzahl gleichzeitiger Verbindungen für Ihre Postgre-Datenbank fest. SQL

> **Note**
>
> Wenn Sie eine DB-Instance herunterskalieren, stellen Sie sicher, dass sie `max_connections` an die neuen Speichergrenzen angepasst wird. Andernfalls kann Postgre SQL möglicherweise nicht gestartet werden oder es könnten Probleme mit der SQL Postgre-Engine auftreten. Dokumentieren Sie alle Änderungen, um future Anpassungen zu erleichtern.

AWS CLI Syntax

Der folgende Befehl ändert sich `max_connections` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify max_connections on a DB parameter group
aws rds modify-db-parameter-group \
```

```
--db-parameter-group-name <parameter_group_name> \
--parameters
 "ParameterName=max_connections,ParameterValue=<new_value>,ApplyMethod=pending-reboot"

# Modify max_connections on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name <parameter_group_name> \
--parameters
 "ParameterName=max_connections,ParameterValue=<new_value>,ApplyMethod=pending-reboot"
```

Typ: Statisch (das Anwenden von Änderungen erfordert einen Neustart)

Standardwert: `LEAST(DBInstanceClassMemory/9531392,5000)` Verbindungen

Beachten Sie die folgenden bewährten Methoden, um die Verwendung von `max_connections` in Amazon RDS oder Aurora zu optimieren und die Auswirkungen auf die Leistung zu minimieren:

- Legen Sie den Parameterwert auf der Grundlage der verfügbaren Systemressourcen fest.
- Überwachen Sie die Verbindungsnutzung, um zu verhindern, dass das Limit schnell erreicht wird.
- Verwenden Sie Verbindungspooling, um die Anzahl der erforderlichen Verbindungen zu reduzieren.
- Verwenden Sie Amazon RDS Proxy für das Verbindungspooling.

Wenn Sie Amazon RDS for Postgre SQL oder Amazon Aurora Postgre SQL -Compatible aktivieren, sollten Sie die verfügbaren Instance-Typen und die ihnen zugewiesenen Ressourcen berücksichtigen und sich auf Arbeitsspeicher und Kapazität konzentrieren. `max_connections` CPU Speicher- und I/O-Details werden von verwaltet AWS, sodass Sie allgemeine Workload-Merkmale und Systemmetriken überwachen können, z. B. `FreeableMemory` über Amazon CloudWatch oder die RDS Amazon-Konsole, um sicherzustellen, dass genügend Speicher für Verbindungen vorhanden ist. Überwachen Sie hohe `CPUUtilization` Werte, was darauf hindeuten könnte, dass Anpassungen erforderlich sind. Vermeiden Sie `max_connections` zu hohe Werte, da sich dies auf die Speicherauslastung und möglicherweise indirekt auf I/O auswirken kann. Denken Sie daran, dass jede Verbindung Speicherplatz verbraucht. Um das richtige Gleichgewicht zu finden, erhöhen Sie den Wert langsam `max_connections` und sehen Sie, wie sich das auf Ihr System auswirkt. Achten Sie auf Anzeichen für eine langsamere Leistung oder eine stärkere CPU Nutzung. Prüfen Sie, ob Ihre Anwendung immer noch einwandfrei funktioniert. Verwenden Sie Funktionen wie Read Replicas in Aurora, um den Lesetraffic zu verteilen und die Belastung der primären Instance zu reduzieren. Überprüfen und passen Sie diese `max_connections` regelmäßig auf der Grundlage der

beobachteten Nutzungsmuster an, um eine optimale Datenbankleistung innerhalb der gegebenen Ressourcenbeschränkungen sicherzustellen.

Einstellung der Autovakuum-Parameter

Amazon RDS for SQL Postgre-Datenbanken und Aurora SQL Postgre-Compatible erfordern eine regelmäßige Wartung, die als Vakuumierung bezeichnet wird. Autovacuum ist ein integriertes SQL Postgre-Hilfsprogramm, das veraltete oder unnötige Daten entfernt, um Speicherplatz in der Datenbank freizugeben. Der Autovacuum-Prozess führt den `VACUUM` Befehl in regelmäßigen Abständen im Hintergrund aus.

Die Optimierung der Autovacuum-Einstellungen ist ein entscheidender Schritt zur Aufrechterhaltung der Leistung, Stabilität und Verfügbarkeit Ihres Amazon RDS for Postgre- SQL oder Aurora SQL Postgre-kompatiblen Datenbanksystems. Indem Sie die Autovacuum-Parameter an Ihre Arbeitslast und Datenbankgröße anpassen, können Sie die Leistung des Autovacuum-Prozesses optimieren und dessen Auswirkungen auf die Systemressourcen reduzieren, wodurch der allgemeine Zustand Ihrer Datenbank verbessert wird.

Neben der Anpassung der Autovacuum-Einstellungen ist es wichtig, die Leistung Ihrer Datenbank und ihrer Komponenten mithilfe der in Amazon RDS und Aurora verfügbaren Tools und Metriken zu überwachen. Durch die Überwachung von Leistungskennzahlen wie Aufblähung, freiem Speicherplatz und Abfragelaufzeiten können Sie potenzielle Probleme erkennen, bevor sie zu schwerwiegenden Problemen werden, und geeignete Maßnahmen ergreifen, um sie zu lösen.

In diesem Abschnitt werden die folgenden Themen und Parameter des Autovakuums behandelt:

- VACUUMund Befehle ANALYZE
- Auf Blähungen prüfen
- Autovakuum
- autovacuum_work_mem
- autovacuum_naptime
- autovacuum_max_workers
- Autovakuum_Vakuum-Skalierungsfaktor
- autovacuum_vacuum_threshold
- autovacuum_analyze_scale_factor
- autovacuum_analyze_threshold
- autovacuum_vacuum_cost_limit

Weitere Informationen zu Autovacuum finden Sie unter den folgenden Links:

- Grundlegendes zum Autovakuum in Amazon RDS für SQL Postgre-Umgebungen (Blogbeitrag)
- Arbeiten mit dem SQL Postgre-Autovakuum auf Amazon RDS for Postgre SQL (Amazon-Dokumentation) RDS
- Paralleles Staubsaugen in Amazon RDS für Postgre SQL und Amazon Aurora SQL Postgre (Blogbeitrag)

VACUUMund Befehle ANALYZE

`VACUUM`sammelt Müll und analysiert optional eine Datenbank. Für die meisten Anwendungen reicht es aus, den Autovacuum-Daemon das Staubsaugen durchführen zu lassen. Einige Administratoren möchten jedoch möglicherweise Datenbankparameter für Autovacuum ändern oder die Aktivitäten des Daemons ergänzen oder ersetzen, indem sie manuell verwaltete `VACUUM` Befehle verwenden, die gemäß einem Scheduler ausgeführt werden können.

`VACUUM`fordert Speicher zurück, der von toten Tupeln belegt ist. Bei SQL Standard-Postgre-Operationen werden Tupel, die gelöscht oder durch ein Update überflüssig gemacht werden, erst dann physisch aus den Tabellen entfernt, wenn eine Operation ausgeführt wird. `VACUUM` Daher empfehlen wir, die Ausführung in `VACUUM` regelmäßigen Abständen durchzuführen, insbesondere bei Tabellen, die häufig aktualisiert werden.

Die Optimierung der `VACUUM` Parameter ist in Amazon RDS for Postgre SQL und Aurora Postgre SQL -Compatible besonders wichtig, da diese verwalteten Datenbankdienste im Vergleich zu selbstverwalteten Postgre-Datenbanken andere Eigenschaften aufweisen. SQL Diese Unterschiede können sich auf die Leistung von Vakuumvorgängen auswirken. Die Optimierung der `VACUUM` Parameter ist wichtig, um die Nutzung von Ressourcen zu optimieren und sicherzustellen, dass Vakuumvorgänge die Leistung und Verfügbarkeit Ihres Datenbanksystems nicht negativ beeinflussen.

Hier sind einige der Parameter, die Sie mit dem `VACUUM` Befehl in Aurora Postgre SQL -Compatible und Amazon RDS for Postgre verwenden können: SQL

- `FULL`
- `FREEZE`
- `VERBOSE`

- ANALYZE
- DISABLE_PAGE_SKIPPING
- table_name
- column_name

VACUUM ANALYZEführt für jede VACUUM ausgewählte Tabelle eine Operation aus, gefolgt von einer ANALYZE Operation. Es bietet eine effiziente Möglichkeit, routinemäßige Wartungsarbeiten durchzuführen.

Wenn Sie den VACUUM Befehl ohne die FULL Option verwenden, wird Speicherplatz für die Wiederverwendung zurückgewonnen. Es ist keine exklusive Sperre für die Tabelle erforderlich, sodass Sie diesen Befehl während der üblichen Lese- und Schreibvorgänge ausführen können. In den meisten Fällen gibt der Befehl dem Betriebssystem jedoch keinen zusätzlichen Speicherplatz zurück, sondern hält ihn für die Wiederverwendung in derselben Tabelle verfügbar. VACUUM FULLSchreibt den gesamten Inhalt der Tabelle in eine neue Festplattendatei ohne zusätzlichen Speicherplatz um und ermöglicht die Rückgabe von ungenutztem Speicherplatz an das Betriebssystem. Dieses Formular ist viel langsamer und erfordert eine ACCESS EXCLUSIVE Sperre für jede Tabelle.

Vollständige Informationen zu diesen Parametern finden Sie in der SQLPostgre-Dokumentation.

In Aurora und Amazon ist RDS Autovacuum ein Daemon-Prozess (Background Utility), der die ANALYZE Befehle VACUUM und regelmäßig ausführt, um redundante Daten in der Datenbank und auf dem Server zu bereinigen. Auch wenn Sie auf das automatische Absaugen angewiesen sind, empfehlen wir Ihnen, die in den folgenden Abschnitten beschriebenen Autovakuum-Einstellungen zu überprüfen und anzupassen, um eine optimale Leistung zu gewährleisten.

Auf Blähungen prüfen

Die folgende SQL Abfrage untersucht jede Tabelle im XML Schema und identifiziert tote Zeilen (Tupel), die Festplattenspeicher verschwenden:

```
SELECT schemaname || '.' || relname as tuplename,
    n_dead_tup,
    (n_dead_tup::float / n_live_tup::float) * 100 as pfrag
FROM pg_stat_user_tables
WHERE schemaname = 'xml' and n_dead_tup > 0 and n_live_tup > 0 order by pfrag desc;
```

Wenn diese Abfrage einen hohen Prozentsatz (pfrag) toter Tupel zurückgibt, können Sie den `VACUUM` Befehl verwenden, um Speicherplatz zurückzugewinnen.

Um die Datengrößen vor und nach Transaktionen zu überwachen, führen Sie die folgende Abfrage in der Shell aus, nachdem Sie eine Verbindung zu einer bestimmten Datenbank hergestellt haben:

```
SELECT pg_size_pretty(pg_relation_size('table_name'));
```

autovacuum

Sie können Autovacuum global mithilfe des `autovacuum` Konfigurationsparameters festlegen, oder Sie können ihn für jede Tabelle einzeln ändern, indem Sie die `autovacuum_enabled` Spalte der `pg_class` Tabelle auf `true` oder `false` für eine bestimmte Tabelle festlegen.

Wenn Sie Autovacuum für eine Tabelle aktivieren, scannt der Datenbankserver die Tabelle regelmäßig auf tote Zeilen und Tupel und entfernt sie im Hintergrund, ohne dass der Datenbankadministrator eingreifen muss. Dies trägt dazu bei, die Tabelle klein zu halten, die Abfrageleistung zu verbessern und die Größe der Backups zu reduzieren.

AWS CLI Syntax

Der folgende Befehl aktiviert `autovacuum` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters "ParameterName=autovacuum,ParameterValue=true,ApplyMethod=immediate"

# Modify autovacuum on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters "ParameterName=autovacuum,ParameterValue=true,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: Aktiviert

Sie können Autovacuum auch für eine bestimmte Tabelle deaktivieren oder aktivieren, indem Sie psql verwenden:

```
ALTER TABLE <table_name> SET (autovacuum_enabled = true);
```

Zu viel Staubsaugen kann die Leistung beeinträchtigen. Daher ist es wichtig, die Leistung des Autovakuierungsprozesses sowie die Leistung Ihrer Datenbank zu überwachen und die Einstellungen nach Bedarf anzupassen.

Beispiel

Ihre SQL Postgre-Datenbank enthält eine Tabelle, die eine große Anzahl von Schreib- und Löschvorgängen empfängt. Ohne Autovacuum würde diese Tabelle irgendwann mit toten Zeilen gefüllt werden (d. h. Zeilen, die zum Löschen markiert, aber noch nicht physisch aus der Tabelle entfernt wurden). Diese toten Zeilen würden Speicherplatz auf der Festplatte beanspruchen, Abfragen verlangsamen und die Größe der Backups erhöhen. Sie können Autovacuum für die Tabelle aktivieren, um automatisch nach toten Zeilen zu suchen und diese zu entfernen, um diese Probleme zu beheben.

autovacuum_work_mem

`autovacuum_work_mem`ist ein SQL Postgre-Konfigurationsparameter, der die Speichermenge steuert, die vom Autovacuum-Prozess verwendet wird, wenn er Tabellenverwaltungsaufgaben wie das Absaugen oder Analysieren ausführt.

In Aurora und Amazon können Sie den Wert von anpassenRDS, um die Leistung `autovacuum_work_mem` zu optimieren.

AWS CLI Syntax

Der folgende Befehl aktiviert `autovacuum_work_mem` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_work_mem on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

```
# Modify autovacuum_work_mem on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_work_mem,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `GREATEST({DBInstanceClassMemory/32768},131072)` KB in Aurora SQL Postgre-Compatible, 64 MB in Amazon RDS für Postgre. SQL Der Standardwert kann jedoch je nach der spezifischen Version von Amazon RDS oder Aurora, die Sie verwenden, variieren.

Beispiel

Ihre Amazon RDS for SQL Postgre-Datenbank enthält eine große Tabelle, die häufig aktualisiert wird. Mit der Zeit stellen Sie fest, dass die Datenbank langsamer wird, und Sie vermuten, dass die automatische Bereinigung zu lange dauert.

Im Rahmen Ihrer Untersuchung überprüfen Sie die Systemprotokolle, verwenden die `pg_stat_activity` Ansicht, um zu sehen, welche Abfragen und Prozesse gerade ausgeführt werden, überprüfen die `pg_stat_user_tables` Ansicht, um Statistiken für jede Tabelle zu sehen, verwenden die `pg_settings` Ansicht, um den Wert von mit dem verfügbaren Speicher auf dem System `autovacuum_work_mem` zu vergleichen, und überwachen die Speicherauslastung auf Spitzen. Nachdem Sie diese Informationen gesammelt haben, können Sie sie `autovacuum_work_mem` auf den optimalen Wert einstellen, den Ihre Arbeitslast benötigt. Um das richtige Gleichgewicht zwischen Speichernutzung und Leistung zu finden, können Sie den Wert auf ein Viertel des verfügbaren Speichers auf dem System festlegen. Nachdem Sie den Wert geändert haben, überwachen Sie die Leistung der Datenbank und stellen möglicherweise fest, dass das Autovakuieren viel schneller als zuvor abgeschlossen wird und Ihre Datenbank insgesamt schneller arbeitet.

autovacuum_naptime

Der `autovacuum_naptime` Parameter steuert das Zeitintervall zwischen aufeinanderfolgenden Durchläufen des Autovakuum-Prozesses. Der Standardwert ist 15 Sekunden für Amazon RDS für Postgre SQL und 5 Sekunden für Aurora Postgre SQL -Compatible.

Nehmen wir zum Beispiel an, dass Ihre Amazon RDS for SQL Postgre-Datenbank über eine Tabelle verfügt, die eine große Menge an Schreib- und Löschvorgängen empfängt. Wenn Sie die Standardeinstellung beibehalten, werden die häufigen Autovacuum-Scans diese hochtransaktive Tabelle stören. Wenn Sie diesen Parameter auf einen hohen Wert setzen, dauert das Intervall zwischen aufeinanderfolgenden Scans länger, und tote Zeilen werden seltener entfernt.

Sie können `autovacuum_naptime` damit die durch den Vakuumprozess verursachte Last verwalten, insbesondere wenn Sie einen ausgelasteten Server haben, der bereits eine hohe CPU I/O-Last hat. Je länger Sie die Zeit für das Nickerchen einstellen, desto seltener wird das automatische Absaugen ausgeführt, wodurch die Serverlast reduziert wird. Eine Einstellung `autovacuum_naptime` auf einen sehr hohen Wert kann jedoch dazu führen, dass Ihre SQL Postgre-Tabellen wachsen und sich tote Zeilen ansammeln, was zu Leistungseinbußen führt. Wir empfehlen, die Leistung des Autovakuumprozesses zu überwachen und die `autovacuum_naptime` Einstellung nach Bedarf anzupassen.

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_naptime` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_naptime on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_naptime,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_naptime on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_naptime,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 15 Sekunden (Amazon RDS für PostgreSQL), 5 Sekunden (Aurora Postgre SQL - Compatible)

autovacuum_max_workers

Der `autovacuum_max_workers` Parameter steuert die maximale Anzahl von Worker-Prozessen, die der Autovacuum-Prozess erzeugen kann. Jeder Worker-Prozess ist für das Staubsaugen oder Analysieren einer einzelnen Tabelle verantwortlich.

Nehmen wir zum Beispiel an, Sie haben eine große Datenbank mit vielen Tabellen, die häufig aktualisiert und gelöscht werden. Wenn Sie für einen niedrigen Wert, z. B. 1, festlegen, kann jeweils nur eine Tabelle gelöscht werden, und es dauert länger, bis alle Tabellen bereinigt sind. `autovacuum_max_workers` Wenn Sie einen hohen Wert wie 8 festlegen`autovacuum_max_workers`, können bis zu acht Tabellen gleichzeitig gesaugt werden. Dadurch kann der Säuberungsprozess für Datenbanken, die viele Tabellen enthalten, beschleunigt werden.

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_max_workers` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_max_workers on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_max_workers,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_max_workers on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_max_workers,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Statisch (das Anwenden von Änderungen erfordert einen Neustart)

Standardwert: `GREATEST(DBInstanceClassMemory/64371566592,3)` Arbeiter

Eine Erhöhung der `autovacuum_max_workers` Einstellung kann die Auslastung des Servers erhöhen, was sich negativ auf die Leistung auswirken kann, wenn Sie nicht über genügend Ressourcen verfügen. Die optimale Einstellung hängt von den spezifischen Anforderungen Ihrer Datenbank, ihrer Größe und der Anzahl der darin enthaltenen Tabellen ab. Wir empfehlen, mit

verschiedenen Werten zu experimentieren und die Leistung zu überwachen, um die optimale Einstellung für Ihren Anwendungsfall zu finden.

autovacuum_vacuum_scale_factor

Der `autovacuum_vacuum_scale_factor` Konfigurationsparameter steuert, wie aggressiv der Autovakuumprozess beim Vakuumieren einer Tabelle sein sollte.

Der Vakuum-Skalierungsfaktor ist ein Bruchteil der Gesamtzahl der Tupel in einer Tabelle, der geändert werden muss, bevor die Tabelle mit Autovakuum gereinigt wird. Der Standardwert ist 0,1 (d. h. 10 Prozent der Tupel müssen geändert werden). Wenn eine Tabelle beispielsweise 1.000.000 Tupel enthält und 100.000 dieser Tupel als tot oder gelöscht markiert sind, wird die Tabelle beim Autovakuumverfahren abgesaugt, abhängig vom Wert von als steuerndem Faktor. `autovacuum_vacuum_threshold`

Mithilfe des `autovacuum_vacuum_scale_factor` Parameters können Sie steuern, wie häufig der Vakuumvorgang ausgeführt wird. Wenn in einer Tabelle viele Schreiboperationen ausgeführt werden, sollten Sie den Vakuum-Skalierungsfaktor verringern, sodass das Autovakuieren häufiger ausgeführt wird, und die Tabelle kleiner halten. Umgekehrt sollten Sie, wenn in einer Tabelle nur wenige Schreibvorgänge ausgeführt werden, den Vakuum-Skalierungsfaktor erhöhen, sodass das Autovakuieren seltener ausgeführt wird, und damit Ressourcen gespart werden.

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_vacuum_scale_factor` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_vacuum_scale_factor on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediate

# Modify autovacuum_vacuum_scale_factor on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediate
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 0.1

Der `autovacuum_vacuum_scale_factor` Parameter funktioniert in Verbindung mit `autovacuum_vacuum_threshold` den `autovacuum_vacuum_cost_limit, and autovacuum_naptime` Parametern. Weitere Informationen zu diesem Parameter finden Sie im AWS Blogbeitrag Understanding autovacuum in Amazon RDS for Postgre-Umgebungen. SQL

autovacuum_vacuum_threshold

Der `autovacuum_vacuum_threshold` Parameter steuert die Mindestanzahl von Tupel-Aktualisierungs- oder Löschvorgängen, die in einer Tabelle ausgeführt werden müssen, bevor Autovacuum sie vakuumiert. Diese Einstellung kann nützlich sein, um unnötiges Löschen von Tabellen zu verhindern, die nicht über eine hohe Rate dieser Operationen verfügen. Der Standardwert ist 50, was der Standardeinstellung der SQL Postgre-Engine für Amazon RDS for Postgre SQL und Aurora SQL Postgre -Compatible entspricht.

Nehmen wir zum Beispiel an, Sie haben eine Tabelle mit 100.000 Zeilen, die auf 50 festgelegt `autovacuum_vacuum_threshold` ist. Wenn die Tabelle nur 49 Aktualisierungen oder Löschungen erhält, wird sie mit Autovacuum nicht vakuiert. Wenn die Tabelle 50 oder mehr Aktualisierungen oder Löschungen erhält, saugt Autovacuum sie ab. Dies hängt vom Wert von `autovacuum_vacuum_scale_factor` multipliziert mit der Anzahl der Tabellenzeilen als steuerndem Faktor ab.

Ein zu hoher Wert für diesen Parameter kann dazu führen, dass die Tabelle wächst und sich tote Zeilen ansammeln, was sich negativ auf die Leistung auswirken kann.

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_vacuum_threshold` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_vacuum_threshold on a DB parameter group
aws rds modify-db-parameter-group \
   --db-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=autovacuum_vacuum_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"
```

```
# Modify autovacuum_vacuum_threshold on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
   --db-cluster-parameter-group-name <parameter_group_name> \
   --parameters
 "ParameterName=autovacuum_vacuum_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 50 Operationen

Der `autovacuum_vacuum_threshold` Parameter funktioniert in Verbindung mit `autovacuum_vacuum_scale_factor` den `autovacuum_vacuum_cost_limit, and autovacuum_naptime` Parametern. Die optimalen Einstellungen hängen von den spezifischen Anforderungen Ihrer Datenbank und der Tabellengröße ab.

Weitere Informationen zu diesem Parameter finden Sie im AWS Blogbeitrag Understanding autovacuum in Amazon RDS for Postgre-Umgebungen. SQL

autovacuum_analyze_scale_factor

Der `autovacuum_analyze_scale_factor` Parameter steuert, wie aggressiv der Autovacuum-Prozess bei der Analyse (Erfassung von Statistiken über die Verteilung der Daten in einer Tabelle) sein sollte.

Der Autovakuum-Prozess verwendet diesen Parameter, um einen Schwellenwert zu berechnen, der auf der Anzahl der Tupel in einer Tabelle basiert. Wenn die Anzahl der Tupeleinfügungen, Aktualisierungen oder Löschungen diesen Schwellenwert überschreitet, analysiert Autovacuum die Tabelle. Der Standardwert ist 0,05 (d. h. 5 Prozent der Tupel müssen geändert werden) für Amazon RDS for Postgre SQL und Aurora Postgre -Compatible. SQL

Nehmen wir zum Beispiel an, Ihre Tabelle hat 1.000.000 Tupel und Sie behalten den Standardwert bei 0,05. `autovacuum_analyze_scale_factor` Wenn die Tabelle 50.000 oder mehr Aktualisierungen oder Löschungen erhält, wird sie je nach `autovacuum_analyze_threshold` Wert automatisch entfernt und die Anzahl der Tabellenzeilen wird als ausschlaggebender Faktor hinzugefügt.

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_analyze_scale_factor` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_analyze_scale_factor on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediat

# Modify autovacuum_analyze_scale_factor on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_scale_factor,ParameterValue=<new_value>,ApplyMethod=immediat
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 0,05 (5 Prozent)

Es ist wichtig, dass der Abfrageplaner Statistiken sammelt, um fundierte Entscheidungen treffen zu können, z. B. darüber, wie auf die Daten zugegriffen und wie sie organisiert werden sollen. Wir empfehlen daher, die Leistung des Autovakuum-Prozesses zu überwachen und die Einstellungen nach Bedarf anzupassen, um sicherzustellen, dass die Statistiken auf dem neuesten Stand sind.

Der `autovacuum_analyze_scale_factor` Parameter funktioniert in Verbindung mit den `autovacuum_analyze_threshold` `autovacuum_analyze_cost_limit, and` `autovacuum_naptime` Parametern. Die optimale Einstellung hängt von den spezifischen Anforderungen Ihrer Datenbank- und Tabellengröße sowie der Häufigkeit der Aktualisierungen ab. Weitere Informationen zu diesem Parameter finden Sie im AWS Blogbeitrag Understanding autovacuum in Amazon RDS for Postgre-Umgebungen. SQL

autovacuum_analyze_threshold

`autovacuum_vacuum_threshold`Der Parameter ist ähnlich wie. `autovacuum_analyze_threshold` Er steuert die Mindestanzahl von Tupeleinfügungen, -aktualisierungen oder -löschungen, die in einer Tabelle vorgenommen werden müssen, bevor sie von Autovacuum analysiert wird. Diese Einstellung kann nützlich sein, um unnötiges Löschen von Daten

in Tabellen zu verhindern, in denen diese Operationen nicht häufig vorkommen. Der Standardwert ist 50, was der Standardeinstellung der SQL Postgre-Engine für Amazon RDS for Postgre SQL und Aurora SQL Postgre -Compatible entspricht.

Nehmen wir zum Beispiel an, Sie haben eine Tabelle mit 100.000 Zeilen und behalten den `autovacuum_analyze_threshold` Standardwert bei 50. Wenn die Tabelle nur 49 Einfügungen, Aktualisierungen oder Löschungen erhält, analysiert Autovacuum sie nicht. Wenn die Tabelle 50 oder mehr Einfügungen, Aktualisierungen oder Löschungen erhält, analysiert Autovacuum sie, wobei der Wert von `autovacuum_analyze_scale_factor` multipliziert mit der Anzahl der Tabellenzeilen als ausschlaggebender Faktor beibehalten wird.

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_analyze_threshold` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_analyze_threshold on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_analyze_threshold on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_analyze_threshold,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 50 Operationen

Dieser Parameter funktioniert in Verbindung mit dem `autovacuum_analyze_scale_factor` Parameter. Berücksichtigen Sie daher beide Einstellungen, wenn Sie Autovacuum konfigurieren.

Es ist wichtig, dass der Abfrageplaner Statistiken sammelt, um fundierte Entscheidungen treffen zu können, z. B. wie auf die Daten zugegriffen und wie sie organisiert werden sollen. Ein `autovacuum_analyze_threshold` zu hoher Wert kann dazu führen, dass die Statistiken veralten

und die Leistung beeinträchtigt wird. Wir empfehlen, die Leistung des Autovakuumprozesses zu überwachen und die Einstellungen nach Bedarf anzupassen.

Weitere Informationen zu diesem Parameter finden Sie im AWS Blogbeitrag Understanding autovacuum in Amazon RDS for Postgre-Umgebungen. SQL

autovacuum_vacuum_cost_limit

Der `autovacuum_vacuum_cost_limit` Parameter steuert die Menge und die I/O-Ressourcen, die ein Autovacuum-Worker verbrauchen kann. CPU

Durch die Begrenzung der Ressourcennutzung von Autovacuum-Prozessen kann verhindert werden, dass diese zu viel CPU oder Festplatten-E/A verbrauchen, was sich auf die Leistung anderer Abfragen auswirken könnte, die auf demselben System ausgeführt werden. Der Parameter gibt ein Kostenlimit an. Dabei handelt es sich um eine Arbeitseinheit, die der Mitarbeiter ausführen darf, bevor er eine Pause einlegen und prüfen muss, ob der Grenzwert noch nicht erreicht ist. Wenn der Parameter beispielsweise auf 2.000 festgelegt ist, darf ein Mitarbeiter 2.000 Arbeitseinheiten bearbeiten, bevor er eine Pause einlegt.

Sie können den `autovacuum_vacuum_cost_limit` Parameter festlegen, indem Sie den `SET` Befehl in einer SQL Postgre-Sitzung verwenden oder einen Befehl verwenden. AWS CLI

AWS CLI Syntax

Der folgende Befehl ändert sich `autovacuum_vacuum_cost_limit` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify autovacuum_vacuum_cost_limit on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_cost_limit,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify autovacuum_vacuum_cost_limit on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=autovacuum_vacuum_cost_limit,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `GREATEST({log(DBInstanceClassMemory/21474836480)*600},200)` Arbeitseinheiten

Wenn Sie den Wert auf `autovacuum_vacuum_cost_limit` zu hoch festlegen, verbraucht der Autovakuum-Prozess möglicherweise zu viele Ressourcen und verlangsamt andere Abfragen. Wenn Sie ihn zu niedrig einstellen, nimmt der Autovakuum-Prozess möglicherweise nicht genügend Speicherplatz zurück, wodurch die Tabelle mit der Zeit größer wird. Es ist wichtig, die richtige Balance zu finden, die für Ihr System funktioniert.

Dieser Parameter wirkt sich nur auf den Autovakuum-Prozess aus, nicht auf die manuellen `VACUUM` Befehle. Außerdem gilt er nur für Autovakuum-Prozesse für, `VACUUM` aber nicht für. `ANALYZE`

Optimieren der Protokollierungsparameter

Durch die Optimierung der Protokollierungsparameter in PostgreSQL können Sie sicherstellen, dass Sie die richtigen Informationen sammeln, ohne große Protokolle zu erstellen, die Ihr System überfordern.

Die Optimierung der Protokollierungsparameter ist entscheidend, um die Protokolldetails mit der Systemleistung und der Festplattennutzung in Einklang zu bringen. Sie können die folgenden Protokollierungsparameter anpassen, um den entsprechenden Detaillierungsgrad in den Protokollen zu erfassen, Probleme zu diagnostizieren und Vorfälle effektiv zu untersuchen und gleichzeitig die Auswirkungen auf die Systemleistung und die Festplattennutzung zu minimieren.

- rds.force_autovacuum_logging
- rds.force_admin_logging_level
- Protokolldauer
- Angabe von log_min_duration_
- log_error_verbosity
- log_Aussage
- log_statement_stats
- log_min_error_statement
- log_min_nachrichten
- log_temp_files
- log_verbindungen
- log_Verbindungsabbrüche

Diese Parameter werden in den folgenden Abschnitten ausführlicher behandelt.

> **Warning**
>
> Die besten Einstellungen für diese Parameter hängen von den Richtlinien und Compliance-Anforderungen Ihrer Organisation ab. Die Aktivierung von Protokollierungsparametern kann jedoch zu einer großen Anzahl von Protokollen und Meldungen führen, was Speicherplatz beanspruchen und die Leistung beeinträchtigen kann, insbesondere bei einer stark ausgelasteten Datenbank. Es wird empfohlen, diese Parameter sorgfältig zu verwenden.

Sie könnten sich beispielsweise dafür entscheiden, sie vorübergehend zu aktivieren, um ein Problem mit einer langsamen SQL-Anweisung einzugrenzen, und sie nach Ablauf des Überwachungszeitraums auszuschalten.

rds.force_autovacuum_logging

Der `rds.force_autovacuum_logging` Parameter (nur in Amazon RDS for PostgreSQL verfügbar) steuert, ob Autovacuum-Aktionen im Serverprotokoll protokolliert werden. Seine Werte sind`disabled`,,`debug5`,`debug4`,`debug3`,`debug2`,`debug1`,`info`,,`notice`, `warning``error`,`log`. `fatal` `panic` Der Standardwert ist `warning`.

Wenn Sie die Option aktivieren`rds.force_autovacuum_logging`, werden alle Aktionen des Autovacuum-Prozesses protokolliert, z. B. wann der Prozess startet, wann er endet und wie viele Zeilen er vakuiert. Dies ist hilfreich beim Debuggen oder bei der Behebung von Leistungsproblemen beim Autovakuieren.

AWS CLI Syntax

Der folgende Befehl ändert sich `rds.force_autovacuum_logging` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify rds.force_autovacuum_logging on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_autovacuum_logging,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify rds.force_autovacuum_logging on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_autovacuum_logging,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `warning`

Beispiel

Sie können den `rds.force_autovacuum_logging` Parameter verwenden, um die Leistung von Autovacuum auf einer Tabelle mit einer sehr hohen Schreibrate zu analysieren. Wenn Ihre Tabelle beispielsweise eine große Anzahl von Schreib- und Löschvorgängen pro Sekunde empfängt und die Leistung langsam ist, können Sie den Parameter aktivieren, um die Start- und Endzeiten jedes Autovakuumlaufs zu protokollieren und zu ermitteln, wie viele Zeilen vakuumiert wurden. Dies kann wertvolle Informationen darüber liefern, wie oft Autovacuum ausgeführt wird, wie lange es dauert und wie viele Zeilen es vakuiert. Sie können diese Informationen dann verwenden, um Autovakuum-Einstellungen wie`autovacuum_vacuum_scale_factor`,`autovacuum_vacuum_threshold`, zu optimieren und die Leistung zu optimieren. `autovacuum_naptime`

rds.force_admin_logging_level

Der `rds.force_admin_logging_level` Parameter (nur in Amazon RDS for PostgreSQL verfügbar) steuert den Detaillierungsgrad der Protokolle, die durch Verwaltungsvorgänge wie Löschen, Analysieren und Neuindizieren erstellt werden. Er akzeptiert die Werte`debug5`,,,`debug4`,`debug3`,,`debug2`,`debug1`,`log`, `info` `notice` `warning`, `error`, `log` und (Standard). `fatal` `off` Die optimale Einstellung hängt von Ihrem Anwendungsfall ab. Wenn Sie beispielsweise ein Problem beheben, möchten Sie den Parameter möglicherweise auf eine Debug-Ebene setzen. Andernfalls können Sie die `warning` Einstellung `log`, oder verwenden.

Wenn Sie diese Einstellung `rds.force_admin_logging_level` auf festlegen`debug1`, können Sie detaillierte Informationen für einen Neuindizierungsvorgang protokollieren, z. B. die Start- und Endzeit, die Anzahl der verarbeiteten Zeilen und alle Fehler oder Warnungen, die während des Vorgangs auftreten. Dies kann wertvolle Informationen darüber liefern, wie der Neuindizierungsprozess abläuft, und Sie bei der Behebung eventuell auftretender Probleme unterstützen.

AWS CLI Syntax

Der folgende Befehl ändert sich `rds.force_admin_logging_level` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify rds.force_admin_logging_level on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
```

```
  --parameters
 "ParameterName=rds.force_admin_logging_level,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify rds.force_admin_logging_level on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=rds.force_admin_logging_level,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `off`

Beispiel

Sie können `rds.force_admin_logging_level` es verwenden, um die Leistung von Verwaltungsvorgängen in mehreren Tabellen in einer großen Datenbank zu überwachen und zu analysieren. Nehmen wir beispielsweise an, Sie haben eine große Datenbank mit vielen Tabellen und möchten die Leistung dieser Tabellen optimieren, indem Sie sie regelmäßig leeren und analysieren. Wenn Sie den `rds.force_admin_logging_level` Parameter auf `info` oder setzen`log`, können Sie die Start- und Endzeiten der einzelnen Operationen sowie die betroffenen Tabellen protokollieren. Sie können diese Informationen verwenden, um die Leistung von Verwaltungsvorgängen in verschiedenen Tabellen nachzuverfolgen und die Tabellen zu identifizieren, für die möglicherweise eine häufigere oder intensivere Wartung erforderlich ist.

Einige der Protokollierungsebenen generieren eine große Anzahl von Protokolldateien und Meldungen, die schnell Speicherplatz belegen können, insbesondere wenn Ihre Datenbank stark ausgelastet ist. Es wird empfohlen, diesen Parameter vorsichtig zu verwenden und ihn nach Ablauf des Überwachungszeitraums auszuschalten.

log_duration

Der `log_duration` Parameter steuert, ob die Dauer jeder Abfrage (d. h. die Zeit, die für die Ausführung benötigt wird) zusammen mit der Abfrage protokolliert wird. Wenn Sie diesen Parameter auf festlegen`on`, wird die Zeit, die für die Ausführung der einzelnen Abfragen benötigt wird, zusammen mit dem Abfragetext in die Protokollausgabe aufgenommen. Die Zeit wird in Millisekunden gemessen.

Der `log_duration` Parameter wird hauptsächlich zur Leistungsoptimierung und Fehlerbehebung verwendet. Indem Sie die Dauer jeder Abfrage protokollieren, können Sie Abfragen identifizieren, deren Ausführung am längsten dauert, und sich dann auf die Optimierung dieser Abfragen konzentrieren. Dies kann Ihnen helfen, Leistungsengpässe zu identifizieren und zu beheben und die Gesamtleistung Ihrer Datenbank zu verbessern.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_duration` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_duration on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_duration,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_duration on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_duration,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `off`

Beispiel

Sie können diesen Parameter verwenden, wenn Sie vermuten, dass eine bestimmte Abfrage oder eine Gruppe von Abfragen Leistungsprobleme verursacht. Wenn Sie den `log_duration` Parameter aktivieren und die Protokollausgabe überprüfen, können Sie feststellen, welche Abfragen am längsten ausgeführt werden, und dann entsprechende Maßnahmen ergreifen, z. B. Indizes optimieren, neue Indizes hinzufügen oder die Abfrage neu schreiben.

Durch die Aktivierung `log_duration` kann das Volumen der Protokollausgabe erhöht werden. Wir empfehlen, sie nur bei Bedarf zu verwenden und sie während des Standardbetriebs auszuschalten, um zu vermeiden, dass der Speicherplatz voll wird oder die Protokolle schwer lesbar sind.

log_min_duration_statement

Der `log_min_duration_statement` Parameter steuert die Mindestdauer in Millisekunden, für die eine SQL-Anweisung ausgeführt wird, bevor sie protokolliert wird.

Dieser Parameter hilft Ihnen dabei, Abfragen mit langer Laufzeit zu identifizieren, die zu Leistungsproblemen führen können. Sie können ihn auf einen Schwellenwert festlegen (eine Laufzeit, die für einen bestimmten Workload als zu lang angesehen wird), um Abfragen zu erfassen, die diesen Schwellenwert überschreiten, und potenzielle Leistungsengpässe zu identifizieren. Ein Beispiel für einen Anwendungsfall finden Sie weiter unten in diesem Handbuch unter Verwenden von Protokollierungsparametern zur Erfassung von Bind-Variablen.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_min_duration_statement` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_min_duration_statement on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_duration_statement,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_ min_duration_statement on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_duration_statement,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: 1 (deaktiviert, was die Standardeinstellung der PostgreSQL-Engine ist)

Beispiel

Mit dem folgenden Befehl werden alle Anweisungen protokolliert, deren Ausführung länger als 100 Millisekunden dauert:

```
aws rds modify-db-parameter-group \
```

```
    --db-parameter-group-name <parameter_group_name> \
    --parameters
 "ParameterName=log_min_duration_statement,ParameterValue=100,ApplyMethod=immediate"
```

log_error_verbosity

Der `log_error_verbosity` Parameter steuert den Detaillierungsgrad, der in der Protokollausgabe für Fehler und Meldungen enthalten ist, die auf der Fehlerebene oder höher protokolliert werden. Dieser Parameter kann einen von drei Werten annehmen: `terse`, `default`oder`verbose`.

- `terse`enthält nur den Meldungstext, die Fehlerstufe sowie die Datei- und Zeilennummer, in der der Fehler aufgetreten ist.
- `default`enthält den Meldungstext, die Fehlerstufe, die Datei- und Zeilennummer sowie den Fehlerkontext.
- `verbose`enthält den Meldungstext, die Fehlerstufe, die Datei- und Zeilennummer, den Fehlerkontext und die vollständige Fehlermeldung.

Stellen Sie den Parameter auf ein`verbose`, um die detailliertesten Informationen zur Problembehandlung und zum Debuggen in einer Produktionsumgebung zu erhalten. In einer Produktionsumgebung sollten Sie ihn auf `default` oder einstellen, `terse` sodass er nur wichtige Informationen bereitstellt und den Protokollspeicher nicht mit zu vielen Details füllt.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_error_verbosity` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_error_verbosity on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_error_verbosity,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_error_verbosity on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_error_verbosity,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `default`

log_statement

Der `log_statement` Parameter steuert, welche SQL-Anweisungen im Serverprotokoll protokolliert werden. Der Parameter kann einen der folgenden Werte annehmen:

- `none`(Standard) protokolliert keine Anweisungen
- `ddl`protokolliert nur DDL-Anweisungen (Data Definition Language) wie und `CREATE TABLE` `ALTER TABLE`
- `mod`protokolliert nur datenmodifizierende Anweisungen wie`INSERT`, und `UPDATE` `DELETE`
- `all`protokolliert alle SQL-Anweisungen

Sie können den `log_statement` Parameter verwenden, um die Menge der in das Protokoll geschriebenen Informationen zu steuern, indem Sie nur die spezifischen Anweisungstypen dokumentieren, die für Ihren Anwendungsfall relevant sind.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_statement` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_statement on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_statement on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: none

Beispiel

In einer Produktionsumgebung möchten Sie möglicherweise festlegen`log_statement`, dass nur DDL-Anweisungen protokolliert und alle am Datenbankschema vorgenommenen Änderungen nachverfolgt werden. `ddl` In einer Entwicklungsumgebung sollten Sie den Parameter auf so einstellen, dass alle Anweisungen protokolliert werden`all`, um das Debuggen und die Problembehandlung zu erleichtern. Ein weiteres Anwendungsbeispiel finden Sie weiter unten in diesem Handbuch unter Verwenden von Protokollierungsparametern zur Erfassung von Bind-Variablen.

Durch die Aktivierung `log_statement` kann das Volumen der Protokollausgabe erhöht werden. Verwenden Sie sie daher nur bei Bedarf und schalten Sie sie aus, um zu vermeiden, dass der Speicherplatz voll wird oder die Protokolle schwer lesbar sind.

Wir empfehlen, dass Sie Ihr System überwachen und den Wert dieses Parameters anpassen, um ein angemessenes Gleichgewicht zwischen der Menge der protokollierten Informationen und dem Speicherplatz und der Leistung des Systems zu erreichen.

log_statement_stats

Der `log_statement_stats` Parameter steuert, ob die Statistiken, die mit der Ausführung einer SQL-Anweisung verknüpft sind, zusammen mit der Anweisung protokolliert werden. Wenn Sie diesen Parameter aktivieren, werden Statistiken wie die Anzahl der betroffenen Zeilen, die Anzahl der gelesenen und geschriebenen Festplattenblöcke und die Zeit, die für die Ausführung der Anweisung benötigt wird, in die Protokollausgabe aufgenommen.

Sie können den `log_statement_stats` Parameter verwenden, um zusätzliche Informationen über die Leistung einzelner Anweisungen und die Gesamtauslastung zu sammeln. Durch die Protokollierung von Statement-Statistiken können Sie Muster bei der Abfrageleistung und der Ressourcennutzung identifizieren und diese Informationen verwenden, um Ihre Datenbank zu optimieren und die Gesamtleistung zu verbessern.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_statement_stats` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_statement_stats on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement_stats,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_statement_stas on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_statement_stats,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `off` (PostgreSQL-Engine-Standard); verwenden Sie 0 oder 1 (Boolean), um Parametergruppen festzulegen

Beispiel

Sie können `log_statement_stats` damit das Verhalten einer bestimmten Abfrage analysieren, herausfinden, wie sie Ressourcen wie CPU, Arbeitsspeicher und Festplatten-I/O nutzt, und ermitteln, ob die Abfrage optimiert werden kann. Sie können diesen Parameter auch verwenden, um festzustellen, ob eine bestimmte Tabelle häufig gelesen wird (was darauf hindeutet, dass ein Index für eine bestimmte Spalte erstellt werden muss) oder ob eine Tabelle zu oft gescannt wird.

Durch die Aktivierung `log_statement_stats` kann das Volumen der Protokollausgabe erhöht werden. Verwenden Sie sie daher nur bei Bedarf und schalten Sie sie aus, um zu vermeiden, dass der Speicherplatz voll wird oder die Protokolle schwer lesbar sind.

log_min_error_statement

Der `log_min_error_statement` Parameter steuert, welche SQL-Anweisungen, die zu einem Fehler führen, protokolliert werden. Seine Werte sind `debug5``debug4`,`debug3`,`debug2`,`debug1`,`info`,`notice`, `warning``error`,`log`,`fatal`, und`panic`. Diese Einstellungen steuern die Menge der Informationen, die in das Protokoll geschrieben werden, sodass Sie Nachrichten mit geringerem Schweregrad herausfiltern können. Sie können für diesen Parameter einen höheren Schweregrad festlegen, um den Umfang der Protokollausgabe zu reduzieren und wichtige Meldungen leichter auffinden zu können.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_min_error_statement` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_min_error_statement on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_error_statement,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_min_error_statement on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_error_statement,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `error` (Standard der PostgreSQL-Engine)

Beispiel

Sie könnten die Verwendung in Betracht ziehen, `log_min_error_statement` wenn Sie ein bestimmtes Problem beheben und Fehlermeldungen von SQL-Anweisungen sehen möchten, die Fehler verursachen.

log_min_messages

Der `log_min_messages` Parameter steuert den Schweregrad, der in das Protokoll geschrieben wird. Sie können den Parameter auf`debug5`,,`debug4`, `debug3``debug2`,`debug1`,`info`,`notice`,`warning`,`error`, `logfatal`, oder setzen`panic`. Diese Einstellungen steuern die Menge der Informationen, die in das Protokoll geschrieben werden, sodass Sie Nachrichten mit geringerem Schweregrad herausfiltern können. Sie können für diesen Parameter einen höheren Schweregrad festlegen, um den Umfang der Protokollausgabe zu reduzieren und wichtige Meldungen leichter auffinden zu können.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_min_messages` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_min_messages on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_messages,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_min_messages on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_min_messages,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `notice`

Beispiel

Wenn Sie ein bestimmtes Problem beheben und alle Fehlermeldungen sehen möchten, können Sie diesen Parameter so einstellen, dass nur Fehler und Probleme mit höherem Schweregrad protokolliert werden. `error` Wenn Sie die Leistung des Systems überwachen möchten, können Sie diesen Parameter auf einstellen, `info` um detailliertere Informationen wie die Dauer und Statistiken der einzelnen Anweisungen anzuzeigen.

Wenn `log_min_messages` Sie einen höheren Schweregrad wählen, verringert sich die Anzahl der Protokolle. Es wird empfohlen, diesen Parameter an Ihren speziellen Anwendungsfall, die Größe des Protokolls, das Sie überprüfen möchten, und den verfügbaren Festplattenspeicher anzupassen.

log_temp_files

Der `log_temp_files` Parameter steuert die Protokollierung von Namen und Größen temporärer Dateien. Er gilt für temporäre Dateien, die für Zwecke wie Sortierungen, Hashes und temporäre Abfrageergebnisse erstellt wurden. Wenn dieser Parameter aktiviert ist, wird für jede temporäre Datei beim Löschen ein Protokolleintrag generiert, einschließlich ihrer Dateigröße in Byte. Sie können diesen Parameter auf 0 (Null) für eine umfassende Protokollierung aller temporären Dateiinformationen oder auf einen positiven Wert für Protokolldateien setzen, die diese Größe

überschreiten (in Kilobyte, wenn Einheiten nicht angegeben sind). Dies kann nützlich sein, um Leistungsengpässe oder andere Probleme im Zusammenhang mit temporärem Speicher zu identifizieren und zu beheben. Standardmäßig ist die Protokollierung temporärer Dateien deaktiviert.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_temp_files` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_temp_files on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_temp_files,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_temp_files on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_temp_files,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: -1 (Standard der PostgreSQL-Engine)

Beispiel

Sie können diesen Parameter aktivieren, wenn Sie vermuten, dass das System zu viel temporären Speicher verwendet oder dass temporäre Dateien nicht ordnungsgemäß gelöscht werden. Wenn Sie die Protokollausgabe untersuchen, können Sie sehen, welche Abfragen oder Vorgänge temporäre Dateien generieren und wie diese Dateien verwendet werden.

Bei einigen Abfragen oder Vorgängen wird eine große Anzahl temporärer Dateien erstellt, sodass sich die Aktivierung auf die Gesamtleistung Ihres Systems `log_temp_files` auswirken kann.

log_connections

Der `log_connections` Parameter steuert, ob Verbindungen zur Datenbank protokolliert werden. Wenn Sie diesen Parameter auf setzen`on`, enthält das Protokoll Informationen über jede

erfolgreiche Verbindung zur Datenbank, z. B. die IP-Adresse des Clients, den Benutzernamen, den Datenbanknamen sowie Datum und Uhrzeit der Verbindung.

Sie können den `log_connections` Parameter verwenden, um Verbindungen zur Datenbank zu überwachen und Fehler zu beheben. Sie können die Benutzer, Anwendungen, Terminals und Bots, die eine Verbindung zur Datenbank herstellen, sehen, von wo aus und wie oft sie eine Verbindung herstellen. Diese Informationen können nützlich sein, um Verbindungsprobleme zu identifizieren und zu lösen oder Nutzungsmuster zu verfolgen.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_connections` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_connections on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_connections,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_connections on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_connections,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `off` (Standard der PostgreSQL-Engine)

Beispiel

Sie können diesen Parameter verwenden, wenn Sie vermuten, dass zu viele Verbindungen zur Datenbank oder ein bestimmter Benutzer oder eine IP-Adresse, die zu häufig eine Verbindung herstellt, die Leistung beeinträchtigen. Wenn Sie den `log_connections` Parameter aktivieren und die Protokollausgabe überprüfen, können Sie die Anzahl und Details aller Verbindungen sehen.

Bevor Sie diesen Parameter aktivieren, sollten Sie die Richtlinien Ihrer Organisation überprüfen und die Sicherheitsauswirkungen der Protokollierung von IP-Adressen und Benutzernamen berücksichtigen.

log_disconnections

Der `log_disconnections` Parameter steuert die Protokollierung von Verbindungsabbrüchen mit der Datenbank. Wenn Sie diesen Parameter auf setzen`on`, werden Informationen über das Ende jeder Sitzung protokolliert, z. B. die IP-Adresse des Clients, der Benutzername, der Datenbankname sowie Datum und Uhrzeit der Verbindungsunterbrechung.

Sie können den `log_disconnections` Parameter verwenden, um die Beendigung von Datenbanksitzungen zu überwachen und Fehler zu beheben. Sie können die Benutzer, Anwendungen, Terminals und Bots sehen, die die Verbindung zur Datenbank trennen, wann und warum. Sie können beispielsweise unerwartete Abbrüche wie einen Absturz oder vom Administrator veranlasste Verbindungsabbrüche überprüfen. Diese Informationen können nützlich sein, um Probleme im Zusammenhang mit Verbindungsabbrüchen oder zur Nachverfolgung von Nutzungsmustern zu identifizieren und zu lösen.

AWS CLI Syntax

Der folgende Befehl ändert sich `log_disconnections` für eine bestimmte DB-Parametergruppe. Diese Änderung gilt für alle Instances oder Cluster, die die Parametergruppe verwenden.

```
# Modify log_disconnections on a DB parameter group
aws rds modify-db-parameter-group \
  --db-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_disconnections,ParameterValue=<new_value>,ApplyMethod=immediate"

# Modify log_disconnections on a DB cluster parameter group
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name <parameter_group_name> \
  --parameters
 "ParameterName=log_disconnections,ParameterValue=<new_value>,ApplyMethod=immediate"
```

Typ: Dynamisch (Änderungen werden sofort angewendet, wenn Sie sie festlegen`ApplyMethod=immediate`)

Standardwert: `off` (Standard der PostgreSQL-Engine)

Beispiel

Sie können dies verwenden, `log_disconnections` wenn Sie vermuten, dass zu viele Benutzer die Verbindung zur Datenbank trennen oder dass ein bestimmter Benutzer oder eine bestimmte

IP-Adresse zu häufig die Verbindung trennt. Wenn Sie den `log_disconnections` Parameter aktivieren und die Protokollausgabe überprüfen, können Sie die Anzahl und die Details aller Verbindungsabbrüche sehen, einschließlich wer, wann und ob vor der Trennung Fehler aufgetreten sind.

Bevor Sie diesen Parameter aktivieren, sollten Sie die Richtlinien Ihrer Organisation überprüfen und die Sicherheitsauswirkungen der Protokollierung von IP-Adressen und Benutzernamen berücksichtigen.

Verwendung von Protokollierungsparametern zur Erfassung von Bind-Variablen

Ein typischer Anwendungsfall für die Erfassung von Bind-Variablen in PostgreSQL ist das Debuggen und die Leistungsoptimierung von SQL-Abfragen. Mit einer Bind-Variablen können Sie Daten an eine Abfrage übergeben, wenn Sie sie ausführen. Durch das Erfassen der Bindungsvariablen können Sie die Eingabedaten sehen, die an eine Abfrage übergeben wurden. Dies kann Ihnen helfen, Probleme mit den Daten oder der Abfrageleistung zu identifizieren. Das Erfassen der Bindungsvariablen kann Ihnen auch dabei helfen, die Eingabedaten zu überprüfen und potenzielle Sicherheitsrisiken oder böswillige Aktivitäten zu erkennen.

Es gibt verschiedene Möglichkeiten, Bind-Variablen für PostgreSQL zu erfassen. Eine Methode besteht darin, die Parameter `debug_print_parse` und `debug_print_rewritten` zu aktivieren. Dies veranlasst PostgreSQL, die geparsten und neu geschriebenen Versionen der SQL-Anweisungen zusammen mit den gebundenen Variablen an das Serverprotokoll zu senden.

- `debug_print_parse`: Wenn Sie diesen Parameter aktivieren, wird der Analysebaum der eingehenden Abfragen in das Serverprotokoll gedruckt. Dies kann nützlich sein, um die Struktur einer Abfrage und die Werte aller gebundenen Parameter zu verstehen.
- `debug_print_rewritten`: Wenn Sie diesen Parameter aktivieren, werden die neu geschriebenen Formen eingehender Abfragen in das Serverprotokoll gedruckt. Dies kann nützlich sein, um zu verstehen, wie der Abfrageplaner eine Abfrage und die Werte aller gebundenen Parameter interpretiert.

Sie können zwei zusätzliche Parameter in Amazon RDS und Aurora verwenden, um Bind-Variablen in Ihren PostgreSQL-Datenbanken zu erfassen:

- `log_min_duration_statement`: Dieser Parameter legt die Mindestdauer einer Anweisung in Millisekunden fest, bevor sie protokolliert wird. Wenn eine Anweisung länger als die angegebene Dauer dauert, werden ihre Bindungswerte in die Protokollausgabe aufgenommen.
- `log_statement`: Dieser Parameter steuert, welche SQL-Anweisungen protokolliert werden. Setzen Sie diesen Parameter auf `all` oder binden Sie, um die gebundenen Werte in das Protokoll aufzunehmen. Eine Erhöhung der Protokollierungsebene wirkt sich auf die Leistung aus. Wir empfehlen daher, die Änderungen nach der Problembehandlung rückgängig zu machen.

Sie können auch die `pg_stat_statements` Erweiterung verwenden, die Leistungsstatistiken für alle von einem Server ausgeführten SQL-Anweisungen bereitstellt, einschließlich des Abfragetextes und der gebundenen Werte. Mit dieser Erweiterung können Sie pgAdmin oder ähnliche Tools verwenden, um die Abfrageleistung zu überwachen und zu analysieren.

Eine andere Möglichkeit besteht darin, die `pg_bind_parameter_status()` Funktion zu verwenden, um die Werte gebundener Parameter aus einer vorbereiteten Anweisung abzurufen oder die `pg_get_parameter_status (paramname)` Funktion zu verwenden, um den Status oder Wert eines bestimmten Laufzeitparameters abzurufen.

Darüber hinaus können Sie Tools von Drittanbietern wie PGBadger verwenden, um die PostgreSQL-Protokolle zu analysieren und die Bind-Variablen und andere Informationen zur weiteren Analyse zu extrahieren.

Optimieren von Replikationsparametern

In PostgreSQL können Sie Datenänderungen von einer PostgreSQL-Datenbank in eine andere replizieren, indem Sie logische Replikation anstelle einer physischen, dateibasierten Replikation verwenden. Die logische Replikation verwendet das Write-Ahead-Protokoll (WAL) zur Erfassung von Änderungen und unterstützt die Replikation ausgewählter Tabellen oder ganzer Datenbanken.

Amazon RDS for PostgreSQL und Aurora PostgreSQL-Compatible unterstützen beide logische Replikation, sodass Sie eine hochverfügbare und skalierbare Datenbankarchitektur einrichten können, die Lese- und Schreibdatenverkehr aus mehreren Quellen verarbeiten kann. Diese Dienste verwenden pglogical, eine Open-Source-PostgreSQL-Erweiterung, um die logische Replikation zu implementieren.

Die Optimierung der logischen Replikation in Aurora und Amazon RDS ist wichtig, um optimale Leistung, Skalierbarkeit und Verfügbarkeit zu erzielen. Sie können die Parameter in der pglogischen Erweiterung anpassen, um die Leistung der logischen Replikation zu verwalten. Beispielsweise ist Folgendes möglich:

- Verbessern Sie die Leistung der Replikation, indem Sie die Anzahl der Arbeitsprozesse erhöhen oder deren Speicherzuweisung anpassen.
- Reduzieren Sie das Risiko von Verzögerungen bei der Replikation, indem Sie die Häufigkeit der Synchronisation zwischen der Quell- und der Replikatdatenbank anpassen.
- Optimieren Sie die Nutzung von Ressourcen, indem Sie die Speicher- und CPU-Zuweisung der Arbeitsprozesse anpassen.
- Stellen Sie sicher, dass der Replikationsprozess die Leistung der Quelldatenbank nicht übermäßig beeinträchtigt.

Sie können die folgenden Parameter in Aurora und Amazon RDS verwenden, um die logische Replikation zu steuern und zu konfigurieren:

- `max_replication_slots`legt die maximale Anzahl von Replikationsslots fest, die auf dem Server erstellt werden können. Ein Replikationsslot ist eine benannte, persistente Reservierung für eine Replikationsverbindung, um WAL-Daten an ein Replikat zu senden.
- `max_wal_senders`legt die maximale Anzahl gleichzeitig verbundener WAL-Senderprozesse fest. WAL-Senderprozesse werden verwendet, um die WAL vom Primärserver zum Replikat zu streamen.

- `wal_sender_timeout`legt die maximale Zeit in Millisekunden fest, die ein WAL-Sender auf eine Antwort vom Replikat wartet, bevor er aufgibt und die Verbindung wieder herstellt.
- `wal_receiver_timeout`legt die maximale Zeit in Millisekunden fest, die ein Replikat auf WAL-Daten aus der Primärdatenbank wartet, bevor es zu einem Timeout kommt.
- `log_replication_commands`, wenn auf gesetzt, werden die replikationsbezogenen on SQL-Anweisungen ausgeführt.

Wenn Sie den `rds.logical_replication` Parameter aktivieren (indem Sie ihn auf 1 setzen), wird der `wal_level` Parameter auf gesetzt`logical`, was bedeutet, dass alle an der Datenbank vorgenommenen Änderungen in einem Format in die WAL geschrieben werden, das gelesen und auf ein Replikat angewendet werden kann. Diese Einstellung ist erforderlich, um die logische Replikation zu aktivieren. Diese Einstellung ermöglicht auch die Replikation von `SELECT` Anweisungen.

Die Einstellung `wal_level` auf `logical` kann die Datenmenge erhöhen, die auf die WAL und damit auf die Festplatte geschrieben wird, was sich auf die Systemleistung auswirken kann. Es wird empfohlen, bei der Aktivierung der logischen Replikation den verfügbaren Festplattenspeicher und die Systemleistung zu berücksichtigen.

Beispiel

Sie möchten Daten aus Ihrer Primärdatenbank zu Sicherungs- und Notfallwiederherstellungszwecken in eine sekundäre Datenbank replizieren. Da die sekundäre Datenbank jedoch ein hohes Volumen an Lesevorgängen aufweist, sollten Sie sicherstellen, dass der Replikationsprozess so schnell und effizient wie möglich abläuft, ohne die Datenintegrität zu gefährden.

Die Standardwerte für die logische Replikation in Amazon RDS und Aurora priorisieren Konsistenz vor Leistung, sodass sie für diesen Anwendungsfall möglicherweise nicht optimal sind. Um Ihre Einstellungen für die logische Replikation im Hinblick auf Geschwindigkeit und Effizienz zu optimieren, können Sie die Parameter wie folgt anpassen:

- Erhöhen Sie den Wert `max_replication_slots` von 10 (Standard für Amazon RDS) oder 20 (Standard für Aurora) auf 30, um potenziellen future Wachstums- und Replikationsanforderungen gerecht zu werden.
- Erhöhen Sie den Wert `max_wal_senders` von 10 (Standard) auf 20, um sicherzustellen, dass genügend WAL-Senderprozesse vorhanden sind, um mit dem Replikationsbedarf Schritt zu halten.

- Verringern Sie den Wert `wal_sender_timeout` von 30 Sekunden (Standard) auf 15 Sekunden, um sicherzustellen, dass inaktive WAL-Senderprozesse schneller beendet werden, wodurch Ressourcen für die aktive Replikation freigesetzt werden.
- Verringern Sie den Wert `wal_receiver_timeout` von 30 Sekunden (Standard) auf 15 Sekunden, um sicherzustellen, dass inaktive WAL-Empfängerprozesse schneller beendet werden, wodurch Ressourcen für die aktive Replikation freigesetzt werden.
- Erhöhen Sie den Wert `max_logical_replication_workers` von 4 (Standard) auf 8, um sicherzustellen, dass genügend logische Replikationsarbeitsprozesse vorhanden sind, um mit dem Replikationsbedarf Schritt zu halten.

Diese Optimierungen ermöglichen eine schnellere und effizientere Datenreplikation bei gleichzeitiger Wahrung der Datenintegrität und -sicherheit.

Wenn beispielsweise ein Notfall eintritt und die Primärdatenbank nicht mehr verfügbar ist, verfügt die sekundäre Datenbank aufgrund des optimierten Replikationsprozesses bereits über die neuesten Daten. Dies würde es Ihrem Geschäftsbetrieb ermöglichen, wichtige Dienste weiterhin ohne Unterbrechung bereitzustellen.

Bewährte Methoden

Die Optimierung der logischen Replikation bei großen Workloads kann eine komplexe Aufgabe sein, die von einer Vielzahl von Faktoren abhängt, darunter der Größe des Datensatzes, der Anzahl der zu replizierenden Tabellen, der Anzahl der Replikate und den verfügbaren Ressourcen. Im Folgenden finden Sie einige allgemeine Tipps zur Optimierung der logischen Replikation bei großen Workloads:

- Überwachen Sie die Replikationsverzögerung. Die Replikationsverzögerung ist der Zeitunterschied zwischen dem Primärserver und den Standby-Servern. Durch die Überwachung der Replikationsverzögerung können Sie potenzielle Engpässe erkennen und Maßnahmen zur Verbesserung der Replikationsleistung ergreifen. Sie können die `pg_current_wal_lsn()` Funktion verwenden, um die aktuelle Replikationsverzögerung zu überprüfen.
- Passen Sie die WAL-Einstellungen an. Die `pg_logical` Erweiterung verwendet WAL, um Änderungen vom Primärserver an den Standby-Server zu übertragen. Wenn die WAL-Einstellungen nicht richtig eingestellt sind, kann die Replikation langsam und unzuverlässig werden. Stellen Sie sicher, dass Sie die `max_replication_slots` Parameter `max_wal_senders` und je nach Ihren Workloads auf angemessene Werte einstellen.

- Entwickeln Sie eine Indexierungsstrategie. Richtige Indizes auf dem Primärserver können dazu beitragen, die Leistung der logischen Replikation zu verbessern, die I/O auf dem Primärserver zu reduzieren und die Systemlast zu reduzieren.
- Verwenden Sie parallel Replikation. Die Verwendung parallel Replikation kann dazu beitragen, die Replikationsgeschwindigkeit zu erhöhen, da mehrere parallel Arbeitsprozesse Daten replizieren können. Diese Funktion ist in PostgreSQL 12 und höher verfügbar.

Nächste Schritte

Nachdem Sie die Speicher-, Replikations-, Autovacuum- und Protokollierungsparameter für Ihre Amazon RDS for PostgreSQL- oder Aurora PostgreSQL-kompatible Datenbank optimiert haben, sollten Sie die folgenden Schritte in Betracht ziehen, um die Leistung Ihrer Datenbank weiter zu verbessern:

- Überwachen Sie Ihre Datenbank. Behalten Sie den Überblick über die Leistung Ihrer Datenbank im Laufe der Zeit, indem Sie integrierte Überwachungstools oder Lösungen von Drittanbietern verwenden. Überwachen Sie wichtige Leistungskennzahlen wie CPU-Auslastung, Festplatten-I/O, Speicherauslastung und Abfragelaufzeiten, um potenzielle Engpässe und Verbesserungsmöglichkeiten zu identifizieren.
- Passen Sie die Parameter kontinuierlich an. Wenn sich Ihre Arbeitslast weiterentwickelt, sollten Sie Ihre Datenbankparameter weiterhin überwachen und anpassen, um eine optimale Leistung sicherzustellen. Überprüfen Sie regelmäßig Systemprotokolle, Fehlermeldungen und Leistungskennzahlen, um neue Optimierungsmöglichkeiten zu identifizieren.
- Implementieren Sie Caching. Verwenden Sie Caching, um die Anzahl der Abfragen zu reduzieren, die die Datenbank betreffen. Sie können Caching auf Anwendungsebene implementieren, indem Sie Tools wie Memcached oder Redis verwenden, oder Sie können Amazon verwenden, ElastiCache um einen In-Memory-Cache für Ihre Datenbank bereitzustellen.
- Optimieren Sie Ihre Abfragen. Schlecht entworfene Abfragen können die Datenbankleistung erheblich beeinträchtigen. Verwenden Sie `EXPLAIN` und andere Tools zur Abfrageoptimierung, um langsame Abfragen zu identifizieren, zu optimieren und unnötige Abfragen zu vermeiden.

Wenn Sie diese Richtlinien befolgen, können Sie die Leistung Ihrer Aurora- oder Amazon RDS for PostgreSQL PostgreSQL-Datenbank optimieren und sicherstellen, dass sie den Anforderungen Ihrer Anwendung durch verbesserte Datenbankleistung, höhere Zuverlässigkeit, geringere Ausfallzeiten, bessere Sicherheit und Kosteneinsparungen entspricht. Indem Sie die Konfigurationsparameter an Ihre Arbeitslast anpassen, können Sie sicherstellen, dass Ihre Datenbank effizient läuft und Ressourcen effektiv nutzt, was zu einer besseren Leistung und einer reaktionsschnelleren Anwendung führt. Darüber hinaus können richtig konfigurierte Parameter die Wahrscheinlichkeit von Fehlern und Sicherheitslücken verringern, was zu einer erhöhten Zuverlässigkeit und besseren Sicherheit führt. Dies kann zu Kosteneinsparungen in Form von geringeren Wartungs- und Ausfallzeiten sowie zu einer insgesamt besseren Benutzererfahrung und -zufriedenheit führen.

Ressourcen

- Amazon Aurora PostgreSQL-Parameter, Teil 1: Speicher- und Abfrageplanverwaltung (AWS Blogbeitrag)
- Amazon Aurora PostgreSQL-Parameter, Teil 2: Replikation, Sicherheit und Protokollierung (AWS Blogbeitrag)
- Amazon Aurora PostgreSQL-Parameter, Teil 3: Optimizer-Parameter (AWS Blogbeitrag)
- Amazon Aurora PostgreSQL-Parameter, Teil 4: ANSI-Kompatibilitätsoptionen (AWS Blogbeitrag)
- Arbeiten mit Amazon Aurora PostgreSQL (Dokumentation)AWS
- Arbeiten mit Amazon RDS for PostgreSQL (Dokumentation)AWS
- Überwachen der Datenbanklast mit Performance Insights auf Amazon RDS (AWS Dokumentation)
- Verwenden von CloudWatch Amazon-Metriken (AWS Dokumentation)
- pg_stats_statements (PostgreSQL-Dokumentation)

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
Die Informationen zu Speicher- und Autovacuum-Parametern wurden aktualisiert	Die Beschreibung des Parameters random_page_cost wurde aktualisiert, fehlende Einheiten zu den Standardwerten für Speicher- und Autovacuum-Parameter hinzugefügt und die Syntax für den AWS CLI Parameter max_connections aktualisiert.	27. Februar 2024
Aktualisierte Informationen über `autovacuum`	Die Autovakuum-Standardeinstellung wurde korrigiert (aktiviert).	27. Dezember 2023
Aktualisierte Informationen über `max_connections`	Der Abschnitt max_connections wurde mit neuen Anleitungen zur Optimierung dieses Parameters aktualisiert.	15. November 2023
Erste Veröffentlichung	—	31. Oktober 2023

AWS Glossar zu präskriptiven Leitlinien

Die folgenden Begriffe werden häufig in Strategien, Leitfäden und Mustern verwendet, die von AWS Prescriptive Guidance bereitgestellt werden. 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 SQL Postgre-Compatible 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 (AmazonRDS) 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 Kundenbeziehungsmanagementsystem (CRM) 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 ein 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 im Bereich künstliche 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 in der AWS Migrationsstrategie finden Sie im Operations Integration Guide. AIOps

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.

Atomarität, 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 ABACfür 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 Einführung der Cloud ()AWS CAF

Ein Framework mit Richtlinien und bewährten Verfahren AWS , das Unternehmen bei der Entwicklung eines effizienten und effektiven Plans für die erfolgreiche Umstellung auf die Cloud unterstützt. AWS CAFgliedert 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 AWS CAF bietet es Anleitungen zur Personalentwicklung, Schulung und Kommunikation, um das Unternehmen auf eine erfolgreiche

Cloud-Einführung vorzubereiten. Weitere Informationen finden Sie AWS CAFauf der Website und im AWS CAFWhitepaper.

AWS Rahmen für die Qualifizierung der Arbeitslast ()AWS WQF

Ein Tool, das Workloads bei der Datenbankmigration bewertet, Migrationsstrategien empfiehlt und Arbeitsschätzungen bereitstellt. AWS WQFist in AWS Schema Conversion Tool ()AWS SCT enthalten. 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 Anrufe und ähnliche Aktionen 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, sogenannte bösartige Bots, 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 in der Regel 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 Änderungsdaten (CDC)

Der Prozess der Nachverfolgung von Änderungen an einer Datenquelle, z. B. einer Datenbanktabelle, und der Aufzeichnung von Metadaten zu der Änderung. Sie können es CDC für verschiedene Zwecke verwenden, z. B. zur 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-Exzellenzzentrum (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 CCoEBeiträgen 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 — Tätigen Sie grundlegende Investitionen, um Ihre Cloud-Einführung zu skalieren (z. B. Einrichtung einer landing zone, Definition einerCCoE, 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 or Bitbucket Cloud. Jede Version des Codes wird als Zweig bezeichnet. 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.

Datenbank für das Konfigurationsmanagement () 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 Phase der Migration, die sich CMDB in der Phase der Portfolioerkennung und -analyse befindet.

Konformitätspaket

Eine Sammlung von AWS Config Regeln und Abhilfemaßnahmen, die Sie zusammenstellen können, um Ihre Compliance- und Sicherheitsprüfungen individuell anzupassen. Mithilfe einer Vorlage können Sie ein Conformance Pack als einzelne Einheit in einer AWS-Konto Region oder in einer Organisation bereitstellen. YAML 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 is commonly described as a pipeline. CI/CDkann 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 Variation 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 historischer 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.

Sprache zur Datenbankmanipulation (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. DVSMerweitert 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, wie z. B. unbeabsichtigte Fehlkonfigurationen oder ein Malware-Angriff.

Notfallwiederherstellung (DR)

Die Strategie und der Prozess, mit denen Sie Ausfallzeiten und Datenverluste aufgrund einer Katastrophe minimieren. 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 dazu, wie Sie domänengesteuertes Design mit dem Strangler-Fig-Muster verwenden können, finden Sie unter Modernizing legacy Microsoft. ASP NET(ASMX) schrittweise 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 der Wertströme in der Entwicklung.

E

EDA

Siehe explorative Datenanalyse.

EDI

Siehe elektronischer Datenaustausch.

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.

elektronischer Datenaustausch (EDI)

Der automatisierte Austausch von Geschäftsdokumenten zwischen Organisationen. Weitere Informationen finden Sie unter Was ist elektronischer Datenaustausch.

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 Dienst, den Sie in einer virtuellen privaten 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) Prinzipalen erstellen AWS PrivateLink und diesen Berechtigungen gewähren. Diese Konten oder Prinzipale können sich privat mit Ihrem Endpunktdienst verbinden, indem sie VPC Schnittstellenendpunkte erstellen. Weitere Informationen finden Sie unter Create an Endpoint Service in der Dokumentation zu Amazon Virtual Private Cloud (AmazonVPC).

Unternehmensressourcenplanung (ERP)

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

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 AWS CAF Sicherheitsepen 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. EDAwird 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.

Eingabeaufforderung mit wenigen Klicks

Geben Sie LLMeinem eine kleine Anzahl von Beispielen, die die Aufgabe und das gewünschte Ergebnis veranschaulichen, bevor Sie ihn bitten, eine ähnliche Aufgabe auszuführen. Bei dieser Technik handelt es sich um eine Anwendung des kontextbezogenen Lernens, bei der Modelle anhand von Beispielen (Aufnahmen) lernen, die in Eingabeaufforderungen eingebettet sind. Bei Aufgaben, die spezifische Formatierungs-, Argumentations- oder Fachkenntnisse erfordern, kann die Eingabeaufforderung mit wenigen Handgriffen effektiv sein. Siehe auch Zero-Shot Prompting.

FGAC

Siehe detaillierte 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.

FM

Siehe Fundamentmodell.

Fundamentmodell (FM)

Ein großes neuronales Deep-Learning-Netzwerk, das mit riesigen Datensätzen generalisierter und unbeschrifteter Daten trainiert wurde. FMssind in der Lage, eine Vielzahl allgemeiner Aufgaben zu erfüllen, z. B. Sprache zu verstehen, Text und Bilder zu generieren und Konversationen in natürlicher Sprache zu führen. Weitere Informationen finden Sie unter Was sind Foundation-Modelle.

G

generative KI

Eine Untergruppe von KI-Modellen, die mit großen Datenmengen trainiert wurden und mit einer einfachen Textaufforderung neue Inhalte und Artefakte wie Bilder, Videos, Text und Audio erstellen können. Weitere Informationen finden Sie unter Was ist Generative KI.

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.

goldenes Bild

Ein Snapshot eines Systems oder einer Software, der als Vorlage für die Bereitstellung neuer Instanzen dieses Systems oder dieser Software verwendet wird. In der Fertigung kann ein Golden Image beispielsweise zur Bereitstellung von Software auf mehreren Geräten verwendet werden und trägt so zur Verbesserung der Geschwindigkeit, Skalierbarkeit und Produktivität bei der Geräteherstellung bei.

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 dazu beiträgt, Ressourcen, Richtlinien und die Einhaltung von Vorschriften in allen Unternehmenseinheiten zu regeln (OUs). Präventiver Integritätsschutz setzt Richtlinien durch, um die Einhaltung von Standards zu gewährleisten. Sie werden mithilfe von Dienststeuerungsrichtlinien 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, eine 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.

Holdout-Daten

Ein Teil historischer, beschrifteter Daten, der aus einem Datensatz zurückgehalten wird, der zum Trainieren eines Modells für maschinelles Lernen verwendet wird. Sie können Holdout-Daten verwenden, um die Modellleistung zu bewerten, indem Sie die Modellvorhersagen mit den Holdout-Daten vergleichen.

Homogene Datenbankmigration

Migrieren Sie Ihre Quelldatenbank in eine Zieldatenbank, die dieselbe Datenbank-Engine verwendet (z. B. Microsoft SQL Server zu Amazon RDS for 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 Principals zugeordnet ist und deren Berechtigungen innerhalb der AWS Cloud Umgebung definiert.

Leerlaufanwendung

Eine Anwendung mit einer durchschnittlichen CPU Speicherauslastung 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.

eingehend (Eingang) VPC

In einer Architektur AWS mit mehreren Konten, VPC die Netzwerkverbindungen von außerhalb einer Anwendung akzeptiert, überprüft und weiterleitet. In der AWS Sicherheitsreferenzarchitektur wird empfohlen, Ihr Netzwerkkonto mit eingehenden und ausgehenden Daten sowie Inspektionen einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet im weiteren Sinne 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. Weitere Informationen finden Sie unter Aufbau einer digitalen Transformationsstrategie für das industrielle Internet der Dinge (IIoT).

Inspektion VPC

In einer Architektur AWS mit mehreren Konten, eine zentrale Architektur, VPC die Inspektionen des Netzwerkverkehrs zwischen VPCs (in demselben oder unterschiedlichen AWS-Regionen), dem Internet und lokalen Netzwerken verwaltet. In der AWS Security Reference Architecture wird empfohlen, Ihr Netzwerkkonto mit eingehendem und ausgehendem Datenverkehr sowie Inspektionen einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet im weiteren Sinne 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 maschinelles Lernen mit. AWS

IoT

Siehe Internet der Dinge.

IT-Informationsbibliothek (ITIL)

Eine Reihe von bewährten Methoden für die Bereitstellung von IT-Services und die Abstimmung dieser Services auf die Geschäftsanforderungen. ITILbietet die Grundlage fürITSM.

IT-Servicemanagement (ITSM)

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 Operations Integration Guide.

ITIL

Weitere Informationen finden Sie in der IT-Informationsbibliothek.

ITSM

Siehe IT-Servicemanagement.

L

Labelbasierte Zugriffskontrolle () LBAC

Eine Implementierung der obligatorischen Zugriffskontrolle (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..

großes Sprachmodell () LLM

Ein KI-Modell für Deep Learning, das anhand einer riesigen Datenmenge vorab trainiert wurde. An LLM kann mehrere Aufgaben ausführen, z. B. Fragen beantworten, Dokumente zusammenfassen, Text in andere Sprachen übersetzen und Sätze vervollständigen. Weitere Informationen finden Sie unter Was sind. LLMs

Große Migration

Eine Migration von 300 oder mehr Servern.

LBAC

Siehe Labelbasierte Zugriffskontrolle.

Geringste Berechtigung

Die bewährte Sicherheitsmethode, bei der nur die für die Durchführung einer Aufgabe erforderlichen Mindestberechtigungen erteilt werden. Weitere Informationen finden Sie in der Dokumentation unter Anwenden von Berechtigungen mit den geringsten Rechten. IAM

Lift and Shift

Siehe 7 Rs.

Little-Endian-System

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

LLM

Siehe großes Sprachmodell.

Niedrigere Umgebungen

Siehe Umgebung.

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.

Fertigungsleitsystem () 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.

MES

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 Dienst, der über genau definierte Kanäle kommuniziert APIs 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 mithilfe von Lightweight über eine klar definierte Schnittstelle. 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. MAPumfasst 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 von Migration Factory gehören in der Regel Betriebsanalysten 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.

Bewertung des Migrationsportfolios () MPA

Ein Online-Tool, das Informationen zur Validierung des Geschäftsszenarios für die Migration auf das bereitstellt. AWS Cloud MPAbietet eine detaillierte Portfoliobewertung (richtige Servergröße, Preisgestaltung, TCO Vergleiche, Analyse der Migrationskosten) sowie Migrationsplanung (Analyse und Datenerfassung von Anwendungen, Gruppierung von Anwendungen, Priorisierung

der Migration und Wellenplanung). Das MPATool (Anmeldung erforderlich) steht allen AWS Beratern und APN Partnerberatern kostenlos zur Verfügung.

Bewertung der Eignung für die Migration (MRA)

Der Prozess der Gewinnung von Erkenntnissen über den Cloud-Bereitschaftsstatus eines Unternehmens, der Identifizierung von Stärken und Schwächen und der Erstellung eines Aktionsplans zur Schließung festgestellter Lücken unter Verwendung von AWS CAF. Weitere Informationen finden Sie im Benutzerhandbuch für Migration Readiness. MRAist die erste Phase der AWS Migrationsstrategie.

Migrationsstrategie

Der Ansatz, der verwendet wird, um einen Workload auf den zu migrieren AWS Cloud. 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

Siehe Origin Access Control.

OAI

Siehe Zugriffsidentität von Origin.

OCM

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 betrieblicher 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

Weitere Informationen finden Sie unter 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 betrieblicher Ebene () OLA

Eine Vereinbarung, in der klargestellt wird, welche funktionalen IT-Gruppen sich gegenseitig versprechen, um eine Vereinbarung auf Serviceniveau zu unterstützen (). SLA

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 Einen Trail für eine Organisation erstellen.

Organisatorisches Änderungsmanagement (OCM)

Ein Framework für das Management wichtiger, disruptiver Geschäftstransformationen aus Sicht der Mitarbeiter, der Kultur und der Führung. OCMunterstützt Unternehmen bei der Vorbereitung und Umstellung auf neue Systeme und Strategien, indem die Einführung von Veränderungen beschleunigt, Übergangsprobleme angegangen und kulturelle und organisatorische Veränderungen vorangetrieben werden. In der AWS Migrationsstrategie wird dieses Framework als Mitarbeiterbeschleunigung bezeichnet, da bei Projekten zur Cloud-Einführung die Geschwindigkeit des Wandels erforderlich ist. Weitere Informationen finden Sie im OCMLeitfaden.

ursprüngliche Zugriffskontrolle (OAC)

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

ursprüngliche Zugriffsidentität () OAI

In CloudFront, eine Option zur Zugriffsbeschränkung, um Ihre Amazon S3 S3-Inhalte zu sichern. Wenn Sie es verwendenOAI, 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, welche eine detailliertere und erweiterte Zugriffskontrolle bietet.

ORR

Siehe Überprüfung der Betriebsbereitschaft.

NICHT

Siehe Betriebstechnologie.

ausgehend (Ausgang) VPC

In einer Architektur AWS mit mehreren Konten eine, VPC die Netzwerkverbindungen verarbeitet, die von einer Anwendung aus initiiert werden. In der AWS Security Reference Architecture wird empfohlen, Ihr Netzwerkkonto mit eingehenden und ausgehenden Daten und Inspektionen einzurichten, VPCs um die bidirektionale Schnittstelle zwischen Ihrer Anwendung und dem Internet im weiteren Sinne zu schützen.

P

Berechtigungsgrenze

Eine IAM Verwaltungsrichtlinie, die den IAM Prinzipalen zugewiesen wird, um die maximalen Berechtigungen festzulegen, die der Benutzer oder die Rolle haben kann. Weitere Informationen finden Sie in der IAM Dokumentation unter Grenzen von Berechtigungen.

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 hierfür PII sind Namen, Adressen und Kontaktinformationen.

PII

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. Diese Entität ist in der Regel ein Root-Benutzer für eine AWS-Konto, eine IAM Rolle oder einen Benutzer.

Weitere Informationen finden Sie in der IAM Dokumentation unter Principal in Roles (Begriffe und Konzepte).

Datenschutz von Haus aus

Ein systemtechnischer Ansatz, der den Datenschutz während des gesamten Entwicklungsprozesses berücksichtigt.

Privat gehostete Zonen

Ein Container, der Informationen darüber enthält, wie Amazon Route 53 auf DNS Anfragen für eine Domain und deren Subdomains innerhalb einer oder mehrerer VPCs Domains 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. Diese Steuerelemente scannen Ressourcen, 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, von der Konstruktion, Entwicklung und Markteinführung über Wachstum und Reife bis hin zu Verkauf und Verkauf.

Produktionsumgebung

Siehe Umgebung.

programmierbare Logiksteuerung (PLC)

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

schnelle Verkettung

Verwenden Sie die Ausgabe einer LLMAufforderung als Eingabe für die nächste Aufforderung, um bessere Antworten zu erzielen. Diese Technik wird verwendet, um eine komplexe Aufgabe in Unteraufgaben zu unterteilen oder um eine vorläufige Antwort iterativ zu verfeinern oder zu

erweitern. Sie trägt dazu bei, die Genauigkeit und Relevanz der Antworten eines Modells zu verbessern und ermöglicht detailliertere, personalisierte Ergebnisse.

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.

publish/subscribe (pub/sub)

Ein Muster, das asynchrone Kommunikation zwischen Microservices ermöglicht, um die Skalierbarkeit und Reaktionsfähigkeit zu verbessern. In einem Microservice-basierten System kann ein Microservice beispielsweise Ereignismeldungen in einem Kanal veröffentlichen MES, 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 SQL relationalen 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

RACIMatrix

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

RAG

Siehe Abruf, erweiterte Generierung.

Ransomware

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

RASCIMatrix

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.

Ziel des Wiederherstellungspunkts (RPO)

Die maximal zulässige Zeitspanne seit dem letzten Datenwiederherstellungspunkt. Damit wird festgelegt, was als akzeptabler Datenverlust zwischen dem letzten Wiederherstellungspunkt und der Serviceunterbrechung gilt.

Ziel für die Wiederherstellungszeit (RTO)

Die maximal zulässige Verzögerung zwischen der Unterbrechung des Dienstes 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.

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

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 RASCIMatrix bezeichnet, und wenn Sie sie ausschließen, wird sie als RACIMatrix 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.

Abruf Erweiterte Generierung () RAG

Eine generative KI-Technologie, bei der ein Benutzer LLMauf eine verlässliche Datenquelle verweist, die sich außerhalb seiner Trainingsdatenquellen befindet, bevor eine Antwort generiert wird. Ein RAG Modell könnte beispielsweise eine semantische Suche in der Wissensdatenbank oder in benutzerdefinierten Daten einer Organisation durchführen. Weitere Informationen finden Sie unter Was ist RAG.

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, die über definierte Zugriffsregeln verfügen. RCACbesteht 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

SAML2.0

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

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 Benutzeranmeldedaten, 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.

Sicherheit durch Design

Ein systemtechnischer Ansatz, der die Sicherheit während des gesamten Entwicklungsprozesses berücksichtigt.

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 (SIEM)

Tools und Dienste, die Systeme zur Verwaltung von Sicherheitsinformationen (SIM) und zur Verwaltung 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 EC2 Amazon-Instance oder das Rotieren von Anmeldeinformationen.

Serverseitige Verschlüsselung

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

Richtlinie zur Dienststeuerung (SCP)

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

Service-Endpunkt

Der 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.

Vereinbarung zum Servicelevel () 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.

Indikator für das Serviceniveau () SLI

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

Ziel auf Serviceniveau () 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.

zentraler Fehlerpunkt (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.

SLO

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.

SPOF

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 Modernizing legacy MicrosoftASP. NET(ASMX) schrittweise Webservices mithilfe von Containern und Amazon API Gateway.

Subnetz

Ein Bereich von IP-Adressen in IhremVPC. 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.

Systemaufforderung

Eine Technik zur Bereitstellung von Kontext, Anweisungen oder Richtlinien für das Verhalten und LLMdie Steuerung des Verhaltens. Systemaufforderungen helfen dabei, den Kontext festzulegen und Regeln für Interaktionen mit Benutzern festzulegen.

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 Netzwerk-Transit-Hub, über den Sie Ihre Netzwerke VPCs und Ihre lokalen Netzwerke miteinander verbinden können. Weitere Informationen finden Sie in der Dokumentation unter Was ist ein Transit-Gateway. AWS 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.

VPCPeering

Eine Verbindung zwischen zweiVPCs, die es Ihnen ermöglicht, den Verkehr mithilfe privater IP-Adressen weiterzuleiten. Weitere Informationen finden Sie in der VPC Amazon-Dokumentation unter Was ist VPC Peering.

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.

WORM

Sehen, einmal schreiben, viele lesen.

WQF

Siehe AWSWorkload-Qualifizierungsrahmen.

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.

Eingabeaufforderung ohne Zwischenfälle

Bereitstellung von Anweisungen zur Ausführung LLMeiner Aufgabe, jedoch ohne Beispiele (Schnappschüsse), die als Orientierungshilfe dienen könnten. Er LLM muss sein vortrainiertes Wissen einsetzen, um die Aufgabe zu lösen. Die Effektivität von Zero-Shot-Prompting hängt von der Komplexität der Aufgabe und der Qualität der Aufforderung ab. Siehe auch Few-Shot-Prompting.

Zombie-Anwendung

Eine Anwendung mit einer durchschnittlichen CPU Speicherauslastung von unter 5 Prozent. 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.