Migration großer MySQL- oder MariaDB-Datenbanken mit mehreren Terabyte zu AWS

AWS Prescriptive Guidance

AWS Prescriptive Guidance: Migration großer MySQL- oder MariaDB-Datenbanken mit mehreren Terabyte zu AWS

Table of Contents

Migration großer MySQL- oder MariaDB-Datenbanken mit mehreren Terabyte zu AWS

Babaiah Valluru und Ankur Bhanawat, Amazon Web Services ()AWS

November 2024 (Geschichte der Dokumente)

Viele Organisationen mit lokalen MySQL- und MariaDB-Datenbankservern sind daran interessiert, ihre Datenbank-Workloads auf die zu migrieren. AWS Cloud Viele entscheiden sich für Amazon Relational Database Service (Amazon RDS) für MariaDB, Amazon RDS for MySQL oder Amazon Aurora MySQL-Compatible Edition. Amazon RDS wurde entwickelt, um die Einrichtung, den Betrieb und die Skalierung relationaler Datenbanken in der Cloud zu vereinfachen. Amazon Aurora ist Teil von Amazon RDS und bietet integrierte Sicherheit, kontinuierliche Backups, serverloses Computing, bis zu 15 Read Replicas, automatisierte Replikation in mehreren Regionen und Integration mit anderen. AWS-Services

Obwohl die Migration zu einem dieser Systeme viele Vorteile bieten AWS-Services kann, ist die Datenbankmigration eine der zeitaufwändigsten und kritischsten Aufgaben, die Datenbankadministratoren ausführen müssen. Die Migration großer Datenbanken erfordert eine genaue Planung und Implementierung und die Sicherstellung, dass die Leistung der migrierten Workloads gleichwertig ist oder sogar verbessert wird. In diesem Handbuch können sich große Datenbanken auf eine einzelne Datenbank mit mehreren Terabyte oder auf viele große Datenbanken beziehen, die zusammen mehrere Terabyte an Daten ergeben. Die Auswahl der richtigen Migrationsdienste und -tools ist der Schlüssel zum Erfolg der Migration. Es gibt zwei gängige Ansätze für die Migration einer Datenbank: logische und physische. Weitere Informationen zu diesen Ansätzen finden Sie in der MySQL - und MariaDB-Dokumentation.

In diesem Handbuch werden verschiedene Open-Source-Tools oder Tools von Drittanbietern beschrieben, mit denen Sie große, lokale MySQL- und MariaDB-Datenbanken mit mehreren Terabyte zu Amazon RDS for MariaDB, Amazon RDS for MySQL oder Amazon Aurora MySQL-Compatible Edition migrieren können. Die in diesem Handbuch beschriebenen Optionen verwenden logische oder physische Migrationsansätze, und jede Option umfasst mehrere Ansätze für die Übertragung der großen Datenbanksicherungsdateien vom lokalen Rechenzentrum in die Cloud, wo Sie die Datenbank aus der Sicherungsdatei wiederherstellen können.

Zielgruppe

Dieses Handbuch richtet sich an Programmdatenbankadministratoren, Datenbankingenieure, Migrationsingenieure, Projektmanager und Betriebs- oder Infrastrukturmanager, die planen, ihre MySQL- oder MariaDB-Datenbanken auf die zu migrieren. AWS Cloud

Gezielte Geschäftsergebnisse

Das Ziel dieses Leitfadens ist es, Ihnen zu helfen:

- Wählen Sie einen Migrationsansatz für eine große Datenbank, der am besten zu Ihrem Anwendungsfall und Ihrer Umgebung passt.
- Vermeiden Sie Verzögerungen und finanzielle Verluste, die entstehen können, wenn die Migrationsstrategie fehlerhaft ist.
- Erfahren Sie mehr über die Vor- und Nachteile der einzelnen Migrationsoptionen.
- Erfahren Sie mehr über verschiedene Ansätze, mit denen Sie große Datenbanksicherungsdateien von Ihrem lokalen Rechenzentrum in das AWS Cloudübertragen können.
- Informieren Sie sich über allgemeine bewährte Methoden für die Migration großer Datenbanken sowie über bewährte Methoden für jedes Tool, mit denen Sie die Datenbank effizienter migrieren können.

Migrationsoptionen für große MySQL- und MariaDB-Datenbanken

Sie können aus einer Vielzahl von Optionen wählen, um von lokalen MySQL- oder MariaDB-Datenbanken zu Amazon Relational Database Service (Amazon RDS) - oder Amazon Aurora MySQL-Compatible Edition-Datenbank-Instances zu migrieren. Die Wahl des richtigen Migrationsansatzes und Tools ist für eine erfolgreiche Migration von entscheidender Bedeutung. In diesem Leitfaden bewerten Sie die Optionen anhand Ihrer Benutzerfreundlichkeit, Datengröße und Ausfallzeiten.

Im Folgenden sind die gängigen Migrationstools und -ansätze aufgeführt, mit denen selbstverwaltete MySQL-Datenbanken mit mehreren Terabyte effizient zu Amazon RDS-, Aurora- oder Amazon Elastic Compute Cloud (Amazon EC2) -Datenbank-Instances migriert werden können:

- Percona XtraBackup(Physisch)
- MyDumper(Logisch)
- mysqldump und mysqlpump(Logisch)
- Geteiltes Backup(Physisch, logisch oder beides)

Im Folgenden sind die gängigen Migrationstools und -ansätze aufgeführt, die verfügbar sind, um Datenbanken mit mehreren Terabyte MySQL-compatible (wie MariaDB) effizient zu Amazon RDS-, Aurora- oder Amazon EC2 EC2-Datenbank-Instances zu migrieren:

- MyDumper(Logisch)
- mysqldump und mysqlpump(Logisch)
- Geteiltes Backup(Physisch, logisch oder beides)

Für jedes Migrationstool gibt es mehrere Methoden, mit denen Sie die große Datenbank-Backup-Datei auf die übertragen können AWS Cloud. Für jedes Tool stehen Optionen zur Verfügung, und Sie können auch Amazon S3 File Gateway verwenden. Weitere Informationen finden Sie unter Verwenden von Amazon S3 File Gateway zum Übertragen von Backup-Dateien in diesem Handbuch.

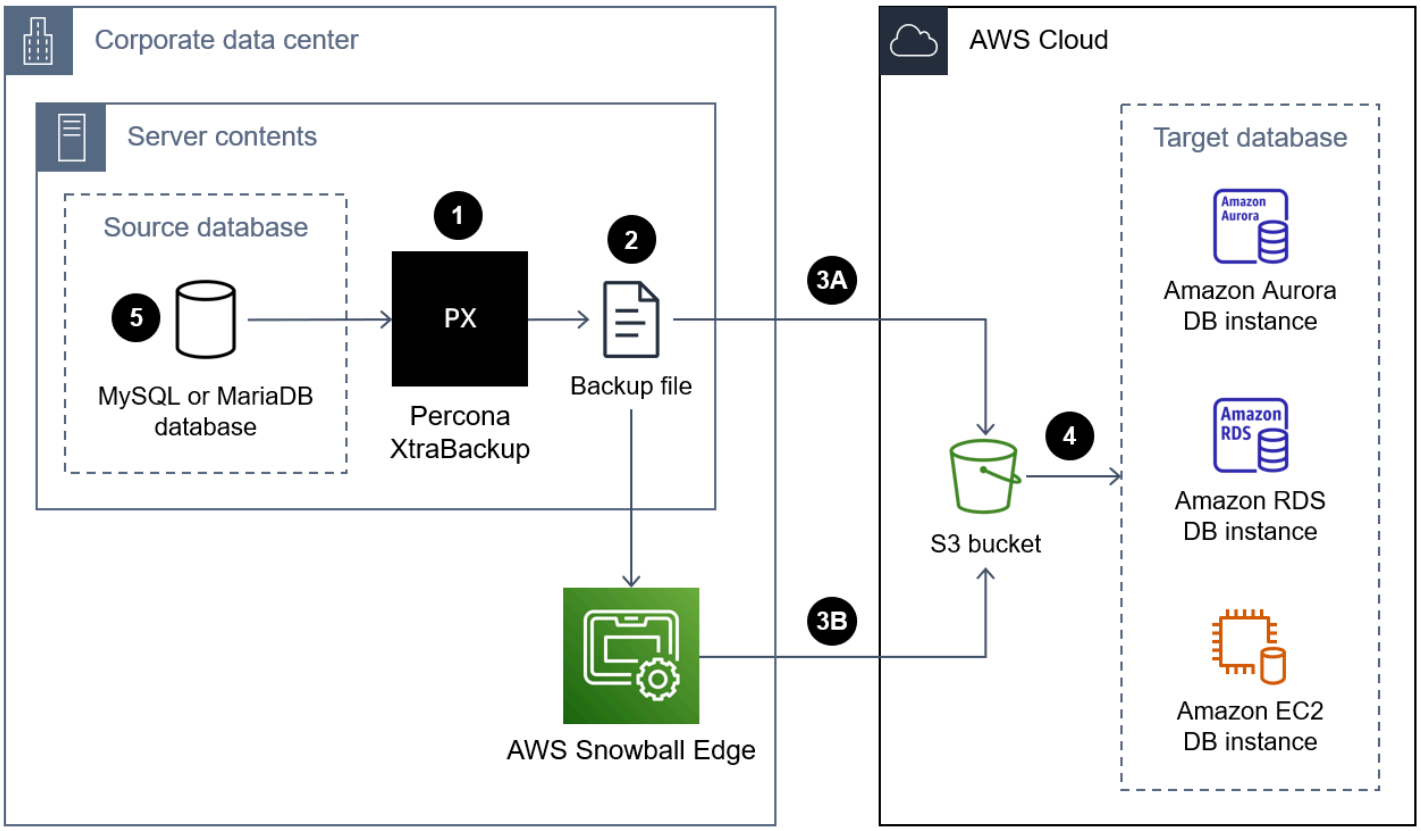
Percona XtraBackup

> **Important**
>
> Percona XtraBackup wird für die MariaDB-Versionen 10.3 oder höher nicht unterstützt und wird für die Versionen 10.1 und 10.2 nur teilweise unterstützt.

Percona XtraBackup ist eine gängige Open-Source-Warm-Backup-Software für MySQL und MariaDB, die blockierungsfreie Backups für InnoDB- und XtraDB-Speicher-Engines erstellt. Es funktioniert mit MySQL- oder MariaDB-Servern. Weitere Informationen über das Tool und einige seiner Funktionen und Vorteile finden Sie XtraBackup in der Percona-Dokumentation unter Über Percona. XtraBackup

Dieses Tool verwendet den Ansatz der physischen Migration. Es kopiert direkt das MySQL- oder MariaDB-Datenverzeichnis und die darin enthaltenen Dateien. Bei großen Datenbanken, z. B. solchen, die größer als 100 GB sind, kann dies zu einer deutlich besseren Wiederherstellungszeit führen als bei einigen anderen Tools. Sie erstellen eine Sicherungskopie der lokalen Quelldatenbank, migrieren die Sicherungsdateien in die Cloud und stellen die Sicherung dann auf der neuen Zieldatenbank-Instance wieder her.

Das folgende Diagramm zeigt die allgemeinen Schritte, die bei der Migration einer Datenbank mithilfe einer XtraBackup Percona-Backup-Datei erforderlich sind. Abhängig von der Größe der Sicherungsdatei stehen zwei Optionen für die Übertragung der Sicherung in einen Amazon Simple Storage Service (Amazon S3) -Bucket im zur Verfügung AWS Cloud.



Im Folgenden sind die Schritte aufgeführt, um mit Percona eine Datenbank XtraBackup zu migrieren: AWS Cloud

1. Installieren Sie Percona XtraBackup auf dem lokalen Server. Wenn Sie Amazon Aurora MySQL Version 2 oder Amazon RDS verwenden, finden Sie weitere Informationen unter Percona XtraBackup 2.4 installieren. Wenn Sie Amazon Aurora MySQL Version 3 verwenden, finden Sie weitere Informationen unter Percona XtraBackup 8.0 installieren in der Percona-Dokumentation XtraBackup.
2. Erstellen Sie eine vollständige Sicherung der MySQL- oder MariaDB-Quelldatenbank. Anweisungen für Percona XtraBackup 2.4 finden Sie unter Vollständige Sicherung. Anweisungen für Percona XtraBackup 8.0 finden Sie unter Erstellen Sie ein vollständiges Backup.
3. Übertragen Sie die Sicherungsdateien mithilfe eines zugelassenen Dienstes oder Tools in Ihrer Organisation über das Internet, z. B. den folgenden:
   - AWS Site-to-Site VPN
   - AWS Client VPN
   - AWS Direct Connect

- Amazon S3 File Gateway (Weitere Informationen finden Sie Verwenden von Amazon S3 File Gateway zum Übertragen von Backup-Dateien in diesem Handbuch.)
- AWS Command Line Interface (AWS CLI)

4. Stellen Sie die Sicherungsdateien aus dem Amazon S3 S3-Bucket auf der Zieldatenbank-Instance wieder her. Detaillierte Informationen finden Sie hier:
   - Informationen zur Aurora MySQL-Compatible Edition finden Sie unter Migrieren von Daten aus MySQL mithilfe eines Amazon S3 S3-Buckets in der Amazon RDS-Dokumentation.
   - Informationen zu Amazon RDS for MySQL oder Amazon EC2 finden Sie unter Daten in eine MySQL-DB-Instance importieren.
   - Informationen zu Amazon RDS for MariaDB oder Amazon EC2 finden Sie unter Daten in eine MariaDB-DB-Instance importieren.
5. (Optional) Sie können die Replikation zwischen der Quelldatenbank und der Zieldatenbank-Instance einrichten. Sie können die Replikation von Binärprotokollen (Binlog) verwenden, um Ausfallzeiten zu reduzieren. Weitere Informationen finden Sie hier:
   - Einstellung der Konfiguration der Replikationsquelle in der MySQL-Dokumentation
   - Für Amazon Aurora finden Sie Folgendes:
     - Synchronisieren des Amazon Aurora MySQL-DB-Clusters mit der MySQL-Datenbank mithilfe der Replikation in der Aurora-Dokumentation
     - Verwendung der Binlog-Replikation in Amazon Aurora in der Aurora-Dokumentation
   - Informationen zu Amazon RDS finden Sie im Folgenden:
     - Arbeiten mit der MySQL-Replikation in der Amazon RDS-Dokumentation
     - Arbeiten mit der MariaDB-Replikation in der Amazon RDS-Dokumentation
   - Informationen zu Amazon EC2 finden Sie im Folgenden:
     - Einrichtung der positionsbasierten Replikation von binären Logdateien in der MySQL-Dokumentation
     - Repliken einrichten in der MySQL-Dokumentation
     - Einrichtung der Replikation in der MariaDB-Dokumentation

Vorteile

- Da Percona einen physischen Migrationsansatz XtraBackup verwendet, ist der Wiederherstellungsprozess in der Regel schneller als bei Tools, die einen logischen

Migrationsansatz verwenden. Dies liegt daran, dass die Leistung eher durch den Festplatten- oder Netzwerkdurchsatz als durch die für die Datenverarbeitung erforderlichen Rechenressourcen begrenzt wird.

- Da der Wiederherstellungsprozess eine direkte Kopie der Dateien aus dem S3-Bucket zur Zieldatenbankinstanz ist, werden XtraBackup Percona-Dateien in der Regel schneller wiederhergestellt als Sicherungsdateien, die mit anderen Tools erstellt wurden.
- Percona XtraBackup ist anpassungsfähig. Es unterstützt beispielsweise mehrere Threads, damit Sie Dateien schneller kopieren können, und unterstützt die Komprimierung, um die Größe des Backups zu reduzieren.

Einschränkungen

- Eine Offline-Sicherung ist nicht möglich, da Percona Zugriff auf den Quelldatenbankserver haben XtraBackup muss.
- Percona XtraBackup kann nur auf Systemen mit identischen Systemarchitekturen verwendet werden. Es ist beispielsweise nicht möglich, ein Backup einer Quelldatenbank, die auf Intel für Windows Server läuft, auf einem ARM für Linux-Zielserver wiederherzustellen.
- Percona wird für MariaDB Version 10.3 oder höher XtraBackup nicht unterstützt, und es wird nur teilweise für MariaDB Version 10.2 und Version 10.1 unterstützt. Weitere Informationen finden Sie unter Percona XtraBackup Overview: Compatibility with MariaDB in der MariaDB-Wissensdatenbank.
- Sie können Percona nicht verwenden XtraBackup , um eine MariaDB-Quelldatenbank auf einer MySQL-Zieldatenbank-Instance wie Amazon RDS for MySQL oder Aurora wiederherzustellen. MySQL-Compatible
- Das Gesamtdatenvolumen und die Anzahl der Objekte, die Sie in einem S3-Bucket speichern können, sind unbegrenzt, die maximale Dateigröße beträgt jedoch 5 TB. Wenn Ihre Backup-Datei 5 TB überschreitet, können Sie sie in mehrere kleinere Dateien aufteilen.
- Wenn die `innodb_file_per_table` Einstellung deaktiviert ist, unterstützt Percona XtraBackup keine Teilsicherungen, die`--tables`,`--tables-exclude`, `--tables-file` `--databases--databases-exclude`, oder `--databases-file` verwenden. Weitere Informationen zu Percona XtraBackup Version 2.4 finden Sie unter Partielle Backups. Weitere Informationen für Percona XtraBackup Version 8.0 finden Sie unter Erstellen einer teilweisen Sicherung.

Bewährte Methoden

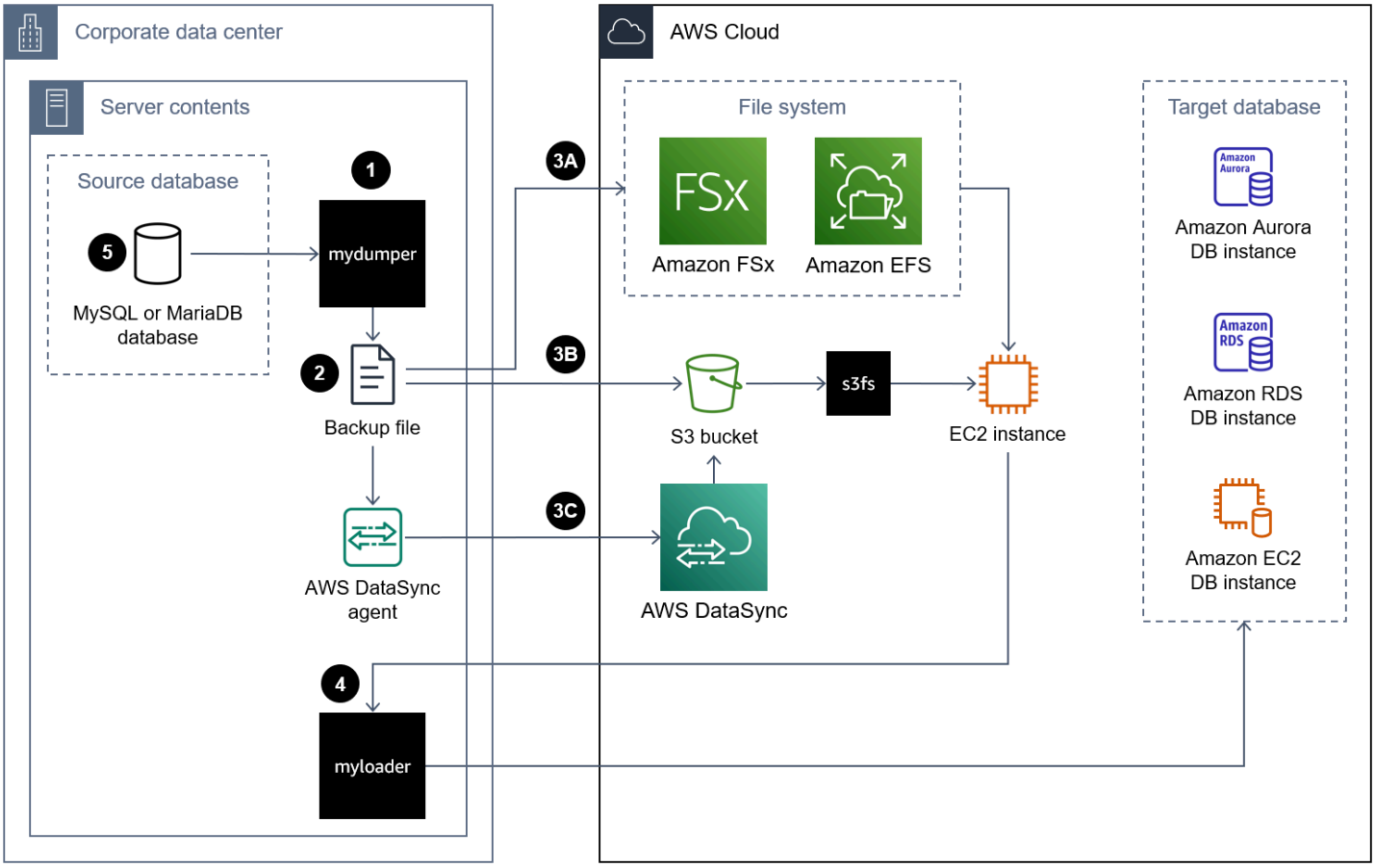
- Gehen Sie wie folgt vor, um die Leistung des Backup-Vorgangs zu verbessern:
  - Kopieren Sie mehrere Dateien parallel mit --parallel= <threads>
  - Komprimieren Sie mehrere Dateien parallel mit --compress-threads= <threads>
  - Erhöhen Sie den Speicher mit --use-memory= <size>
  - Verschlüsseln Sie mehrere Dateien parallel mit --encrypt-threads= <threads>
- Stellen Sie sicher, dass auf dem Quellserver ausreichend Speicherplatz für die Datenbanksicherungsdateien vorhanden ist.
- Generieren Sie die Datenbanksicherung mit der Percona-Datei im xbstream-Format (.xbstream). Weitere Informationen finden Sie in der Percona-Dokumentation unter Die xbstream-Binärdatei im Überblick. XtraBackup

MyDumper

MyDumper(GitHub) ist ein logisches Open-Source-Migrationstool, das aus zwei Dienstprogrammen besteht:

- MyDumper exportiert ein konsistentes Backup von MySQL-Datenbanken. Es unterstützt das Sichern der Datenbank mithilfe mehrerer parallel Threads, bis zu einem Thread pro verfügbarem CPU-Kern.
- myloader liest die von erstellten Sicherungsdateien MyDumper, stellt eine Verbindung zur Zieldatenbankinstanz her und stellt dann die Datenbank wieder her.

Das folgende Diagramm zeigt die wichtigsten Schritte, die bei der Migration einer Datenbank mithilfe einer MyDumper Sicherungsdatei erforderlich sind. Dieses Architekturdiagramm enthält drei Optionen für die Migration der Sicherungsdatei vom lokalen Rechenzentrum zu einer EC2-Instanz in der. AWS Cloud



Im Folgenden finden Sie die Schritte MyDumper zur Migration einer Datenbank auf die: AWS Cloud

1. Install MyDumper und Myloader. Anweisungen finden Sie unter So installieren Sie mydumper/myloader (GitHub).
2. Wird verwendet MyDumper , um eine Sicherungskopie der MySQL- oder MariaDB-Quelldatenbank zu erstellen. Anweisungen finden Sie unter Wie benutzt man. MyDumper
3. Verschieben Sie die Sicherungsdatei auf eine EC2-Instanz in der, AWS Cloud indem Sie einen der folgenden Ansätze verwenden:

   Ansatz 3A — Mounten Sie ein Amazon FSx - oder Amazon Elastic File System (Amazon EFS) -Dateisystem auf dem lokalen Server, auf dem Ihre Datenbank-Instance ausgeführt wird. Sie können AWS Direct Connect oder verwenden Site-to-Site VPN , um die Verbindung herzustellen. Sie können die Datenbank direkt auf der bereitgestellten Dateifreigabe sichern, oder Sie können die Sicherung in zwei Schritten durchführen, indem Sie die Datenbank in einem lokalen Dateisystem sichern und sie dann auf das gemountete FSx- oder EFS-Volume hochladen.

Mounten Sie als Nächstes das Amazon FSx- oder Amazon EFS-Dateisystem, das ebenfalls auf dem lokalen Server bereitgestellt ist, auf einer EC2-Instance.

Ansatz 3B — Verwenden Sie das AWS CLI AWS SDK oder die Amazon S3 S3-REST-API, um die Sicherungsdatei direkt vom lokalen Server in einen S3-Bucket zu verschieben. Befindet sich der Ziel-S3-Bucket in einem AWS-Region , der weit vom Rechenzentrum entfernt ist, können Sie Amazon S3 Transfer Acceleration verwenden, um die Datei schneller zu übertragen. Verwenden Sie das s3fs-fuse-Dateisystem, um den S3-Bucket auf der EC2-Instance zu mounten.

Methode 3C — Installieren Sie den AWS DataSync Agenten im lokalen Rechenzentrum und verwenden Sie ihn dann, AWS DataSyncum die Sicherungsdatei in einen Amazon S3 S3-Bucket zu verschieben. Verwenden Sie das s3fs-fuse-Dateisystem, um den S3-Bucket auf der EC2-Instance zu mounten.

> **Note**
>
> Sie können Amazon S3 File Gateway auch verwenden, um die großen Datenbank-Backup-Dateien in einen S3-Bucket im zu übertragen AWS Cloud. Weitere Informationen finden Sie unter Verwenden von Amazon S3 File Gateway zum Übertragen von Backup-Dateien in diesem Handbuch.

4. Verwenden Sie myloader, um das Backup auf der Zieldatenbank-Instance wiederherzustellen. Anweisungen finden Sie unter Verwendung von myloader ()GitHub.
5. (Optional) Sie können die Replikation zwischen der Quelldatenbank und der Zieldatenbankinstanz einrichten. Sie können die Replikation von Binärprotokollen (Binlog) verwenden, um Ausfallzeiten zu reduzieren. Weitere Informationen finden Sie hier:
   - Einstellung der Konfiguration der Replikationsquelle in der MySQL-Dokumentation
   - Für Amazon Aurora finden Sie Folgendes:
     - Synchronisieren des Amazon Aurora MySQL-DB-Clusters mit der MySQL-Datenbank mithilfe der Replikation in der Aurora-Dokumentation
     - Verwendung der Binlog-Replikation in Amazon Aurora in der Aurora-Dokumentation
   - Informationen zu Amazon RDS finden Sie im Folgenden:
     - Arbeiten mit der MySQL-Replikation in der Amazon RDS-Dokumentation
     - Arbeiten mit der MariaDB-Replikation in der Amazon RDS-Dokumentation
   - Informationen zu Amazon EC2 finden Sie im Folgenden:

- Einrichtung der positionsbasierten Replikation von binären Logdateien in der MySQL-Dokumentation
- Repliken einrichten in der MySQL-Dokumentation
- Einrichtung der Replikation in der MariaDB-Dokumentation

Vorteile

- MyDumper unterstützt Parallelität mithilfe von Multithreading, wodurch die Geschwindigkeit von Sicherungs- und Wiederherstellungsvorgängen verbessert wird.
- MyDumper vermeidet teure Routinen zur Zeichensatzkonvertierung, wodurch sichergestellt wird, dass der Code hocheffizient ist.
- MyDumper vereinfacht das Anzeigen und Analysieren von Daten, indem separate Dumping-Dateien für Tabellen und Metadaten verwendet werden.
- MyDumper verwaltet Schnappschüsse für alle Threads und bietet genaue Positionen der primären und sekundären Protokolle.
- Sie können Perl Compatible Regular Expressions (PCRE) verwenden, um anzugeben, ob Tabellen oder Datenbanken ein- oder ausgeschlossen werden sollen.

Einschränkungen

- Sie können ein anderes Tool wählen, wenn Ihre Datentransformationsprozesse Zwischenspeicherdateien im Flatformat statt im SQL-Format erfordern.
- myloader importiert Datenbankbenutzerkonten nicht automatisch. Wenn Sie das Backup auf Amazon RDS oder Aurora wiederherstellen, erstellen Sie die Benutzer mit den erforderlichen Berechtigungen neu. Weitere Informationen finden Sie unter Rechte für Master-Benutzerkonten in der Amazon RDS-Dokumentation. Wenn Sie das Backup auf einer Amazon EC2 EC2-Datenbank-Instance wiederherstellen, können Sie die Benutzerkonten der Quelldatenbank manuell exportieren und in die EC2-Instance importieren.

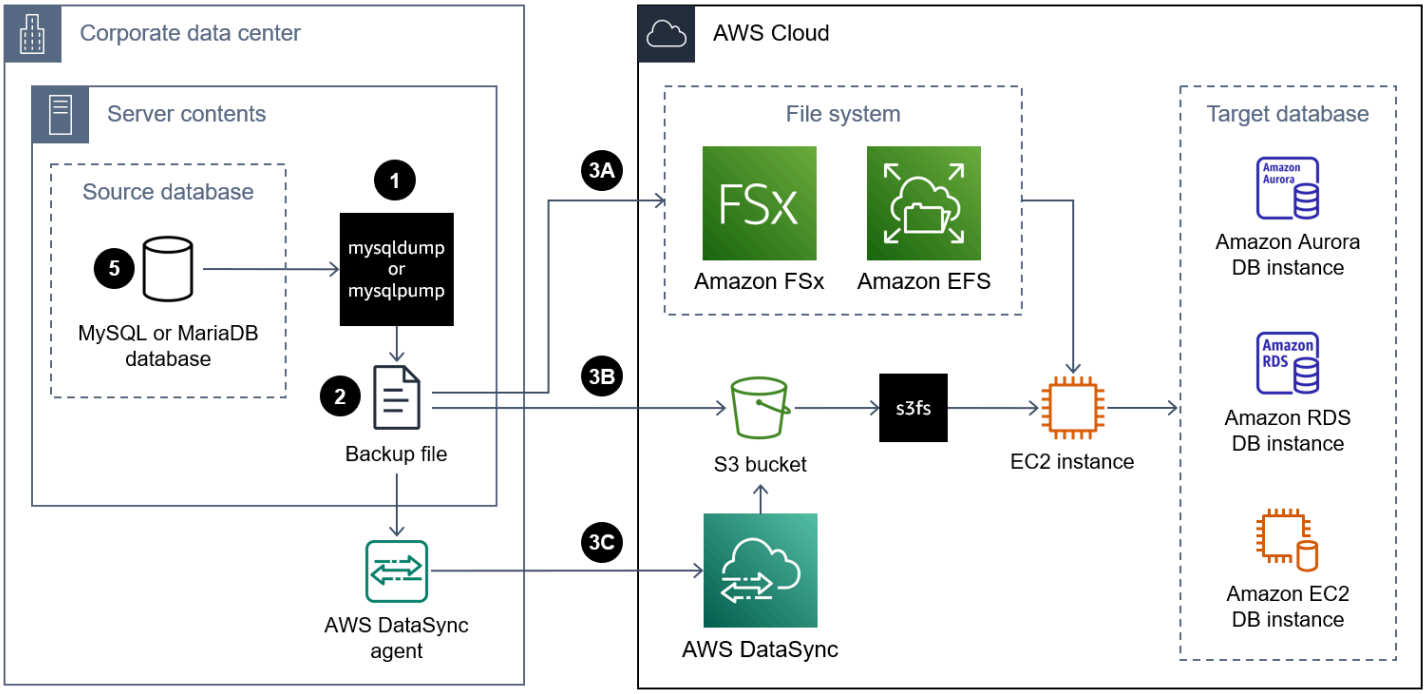
Bewährte Methoden

- Konfigurieren MyDumper Sie es so, dass jede Tabelle in Segmente unterteilt wird, z. B. 10.000 Zeilen in jedem Segment, und jedes Segment in eine separate Datei geschrieben wird. Dadurch ist es möglich, die Daten später parallel zu importieren.
- Wenn Sie die InnoDB-Engine verwenden, verwenden Sie die `--trx-consistency-only` Option, um das Sperren zu minimieren.
- Die Verwendung MyDumper zum Exportieren der Datenbank kann leseintensiv werden, und der Vorgang kann sich auf die Gesamtleistung der Produktionsdatenbank auswirken. Wenn Sie über eine Replikatdatenbankinstanz verfügen, führen Sie den Exportvorgang vom Replikat aus aus. Bevor Sie den Export aus dem Replikat ausführen, beenden Sie den Replikations-SQL-Thread. Dadurch kann der Exportvorgang schneller ausgeführt werden.
- Exportieren Sie die Datenbank nicht während der Hauptgeschäftszeiten. Durch die Vermeidung von Spitzenzeiten kann die Leistung Ihrer primären Produktionsdatenbank während des Datenbankexports stabilisiert werden.
- Amazon RDS for MySQL unterstützt das `keyring_aws` Plugin nicht. Weitere Informationen finden Sie unter Bekannte Probleme und Einschränkungen. Um die lokalen verschlüsselten Tabellen zur Amazon RDS-Instance zu migrieren, müssen Sie in den Backup-Skripten `ENCRYPTION` oder `DEFAULT ENCRYPTION` aus der `CREATE TABLE` Syntax entfernen. Für die Verschlüsselung im Ruhezustand können Sie einen AWS Key Management Service (AWS KMS) -Schlüssel verwenden. Weitere Informationen finden Sie unter Verschlüsseln von Amazon-RDS-Ressourcen.

mysqldump und mysqlpump

mysqldump und mysqlpump sind native Datenbank-Backup-Tools für MySQL. MariaDB unterstützt mysqldump, aber nicht mysqlpump. Beide Tools erstellen logische Backups und sind Teil der MySQL-Client-Programme. mysqldump unterstützt Single-Thread-Verarbeitung. mysqlpump unterstützt die parallel Verarbeitung von Datenbanken und Objekten innerhalb von Datenbanken, um den Dump-Prozess zu beschleunigen. Es wurde in MySQL Version 5.7.8 eingeführt. mysqlpump wurde in MySQL Version 8.4 entfernt.

Das folgende Diagramm zeigt die wichtigsten Schritte bei der Migration einer Datenbank mithilfe einer mysqldump- oder mysqlpump-Backup-Datei.



Im Folgenden finden Sie die Schritte zur Verwendung von mysqldump oder mysqlpump zur Migration einer Datenbank auf: AWS Cloud

1. Installieren Sie MySQL Shell auf dem lokalen Server. Anweisungen finden Sie in der MySQL-Dokumentation unter Installation von MySQL Shell. Dadurch werden sowohl mysqldump als auch mysqlpump installiert.
2. Erstellen Sie mit mysqldump oder mysqlpump eine Sicherungskopie der lokalen Quelldatenbank. Anweisungen finden Sie unter mysqldump und mysqlpump in der MySQL-Dokumentation oder unter Backups mit mysqldump erstellen in der MariaDB-Dokumentation. Weitere Hinweise zum Aufrufen von MySQL-Programmen und zum Angeben von Optionen finden Sie unter MySQL-Programme verwenden.
3. Verschieben Sie die Sicherungsdatei auf eine EC2-Instanz in der, AWS Cloud indem Sie einen der folgenden Ansätze verwenden:

   Ansatz 3A — Mounten Sie ein Amazon FSx - oder Amazon Elastic File System (Amazon EFS) -Dateisystem auf dem lokalen Server, auf dem Ihre Datenbank-Instance ausgeführt wird. Sie können AWS Direct Connect oder verwenden Site-to-Site VPN , um die Verbindung herzustellen. Sie können die Datenbank direkt auf der bereitgestellten Dateifreigabe sichern, oder Sie können die Sicherung in zwei Schritten durchführen, indem Sie die Datenbank in einem lokalen Dateisystem sichern und sie dann auf das gemountete FSx- oder EFS-Volume hochladen.

Mounten Sie als Nächstes das Amazon FSx- oder Amazon EFS-Dateisystem, das ebenfalls auf dem lokalen Server bereitgestellt ist, auf einer EC2-Instance.

Ansatz 3B — Verwenden Sie das AWS CLI AWS SDK oder die Amazon S3 S3-REST-API, um die Sicherungsdatei direkt vom lokalen Server in einen S3-Bucket zu verschieben. Befindet sich der Ziel-S3-Bucket in einem AWS-Region , der weit vom Rechenzentrum entfernt ist, können Sie Amazon S3 Transfer Acceleration verwenden, um die Datei schneller zu übertragen. Verwenden Sie das s3fs-fuse-Dateisystem, um den S3-Bucket auf der EC2-Instance zu mounten.

Methode 3C — Installieren Sie den AWS DataSync Agenten im lokalen Rechenzentrum und verwenden Sie ihn dann, AWS DataSyncum die Sicherungsdatei in einen Amazon S3 S3-Bucket zu verschieben. Verwenden Sie das s3fs-fuse-Dateisystem, um den S3-Bucket auf der EC2-Instance zu mounten.

> **Note**
>
> Sie können Amazon S3 File Gateway auch verwenden, um die großen Datenbank-Backup-Dateien in einen S3-Bucket im zu übertragen AWS Cloud. Weitere Informationen finden Sie unter Verwenden von Amazon S3 File Gateway zum Übertragen von Backup-Dateien in diesem Handbuch.

4. Verwenden Sie die native Wiederherstellungsmethode, um das Backup in der Zieldatenbank wiederherzustellen. Anweisungen finden Sie unter Reloading SQL-Format Backups in der MySQL-Dokumentation oder unter Daten aus Dump-Dateien wiederherstellen in der MariaDB-Dokumentation.
5. (Optional) Sie können die Replikation zwischen der Quelldatenbank und der Zieldatenbank-Instance einrichten. Sie können die Replikation von Binärprotokollen (Binlog) verwenden, um Ausfallzeiten zu reduzieren. Weitere Informationen finden Sie hier:
   - Einstellung der Konfiguration der Replikationsquelle in der MySQL-Dokumentation
   - Für Amazon Aurora finden Sie Folgendes:
     - Synchronisieren des Amazon Aurora MySQL-DB-Clusters mit der MySQL-Datenbank mithilfe der Replikation in der Aurora-Dokumentation
     - Verwendung der Binlog-Replikation in Amazon Aurora in der Aurora-Dokumentation
   - Informationen zu Amazon RDS finden Sie im Folgenden:
     - Arbeiten mit der MySQL-Replikation in der Amazon RDS-Dokumentation
     - Arbeiten mit der MariaDB-Replikation in der Amazon RDS-Dokumentation

- Informationen zu Amazon EC2 finden Sie im Folgenden:
  - Einrichtung der positionsbasierten Replikation von binären Logdateien in der MySQL-Dokumentation
  - Repliken einrichten in der MySQL-Dokumentation
  - Einrichtung der Replikation in der MariaDB-Dokumentation

Vorteile

- mysqldump und mysqlpump sind in der MySQL Server-Installation enthalten
- Die von diesen Tools generierten Sicherungsdateien haben ein besser lesbares Format.
- Bevor Sie die Sicherungsdatei wiederherstellen, können Sie die resultierende .sql-Datei mit einem Standard-Texteditor ändern.
- Sie können eine bestimmte Tabelle, Datenbank oder sogar eine bestimmte Datenauswahl sichern.
- mysqldump und mysqlpump sind unabhängig von der Maschinenarchitektur.

Einschränkungen

- mysqldump ist ein Backup-Prozess mit einem einzigen Thread. Die Leistung bei der Erstellung eines Backups ist für kleine Datenbanken gut, kann jedoch ineffizient werden, wenn die Backup-Größe größer als 10 GB ist.
- Sicherungsdateien im logischen Format sind umfangreich, insbesondere wenn sie als Text gespeichert werden, und lassen sich oft nur langsam erstellen und wiederherstellen.
- Die Datenwiederherstellung kann langsam sein, da das erneute Anwenden von SQL-Anweisungen in der Ziel-DB-Instance eine intensive Festplatten I/O - und CPU-Verarbeitung für das Einfügen, die Indexerstellung und die Durchsetzung von Einschränkungen der referentiellen Integrität erfordert.
- Das Hilfsprogramm mysqlpump wird für MySQL-Versionen vor 5.7.8 oder für Versionen 8.4 und höher nicht unterstützt.
- Standardmäßig erstellt mysqlpump keine Sicherungskopie der Systemdatenbanken, wie z. B. oder. `performance_schema` `sys` Um einen Teil der Systemdatenbank zu sichern, geben Sie ihm in der Befehlszeile einen expliziten Namen.
- mysqldump sichert keine InnoDB-Anweisungen. `CREATE TABLESPACE`

> **Note**
>
> Backups von CREATE TABLESPACE-Anweisungen und Systemdatenbanken sind nur nützlich, wenn Sie MySQL- oder MariaDB-Datenbanksicherungen auf einer EC2-Instance wiederherstellen. Diese Backups werden nicht für Amazon RDS oder Aurora verwendet.

Bewährte Methoden

- Wenn Sie die Datenbanksicherung wiederherstellen, deaktivieren Sie die SchlüsselprüfungenFOREIGN_KEY_CHECKS, z. B. auf Sitzungsebene in der Zieldatenbank. Dadurch wird die Wiederherstellungsgeschwindigkeit erhöht.
- Stellen Sie sicher, dass der Datenbankbenutzer über ausreichende Rechte verfügt, um das Backup zu erstellen und wiederherzustellen.

Geteiltes Backup

Bei einer Split-Backup-Strategie migrieren Sie einen großen Datenbankserver, indem Sie das Backup in mehrere Teile aufteilen. Sie können unterschiedliche Ansätze verwenden, um jeden Teil des Backups zu migrieren. Dies kann die beste Option für die folgenden Anwendungsfälle sein:

- Großer Datenbankserver, aber kleine Einzeldatenbanken — Dies ist ein guter Ansatz, wenn die Größe des gesamten Datenbankservers mehrere TB beträgt, die Größe jeder einzelnen, unabhängigen Benutzerdatenbank jedoch weniger als 1 TB beträgt. Um den Gesamtmigrationszeitraum zu verkürzen, können Sie einzelne Datenbanken separat und parallel migrieren.

  Lassen Sie uns ein Beispiel für einen lokalen Datenbankserver mit 2 TB verwenden. Dieser Server besteht aus vier Datenbanken, die jeweils 0,5 TB groß sind. Sie können Backups jeder einzelnen Datenbank separat erstellen. Beim Wiederherstellen der Sicherung können Sie entweder alle Datenbanken auf einer Instanz parallel wiederherstellen, oder wenn die Datenbanken unabhängig sind, können Sie jede Sicherung auf einer separaten Instanz wiederherstellen. Es hat sich bewährt, unabhängige Datenbanken auf separaten Instanzen wiederherzustellen, anstatt sie auf derselben Instanz wiederherzustellen. Weitere Informationen finden Sie unter Bewährte Methoden in diesem Handbuch.

- Großer Datenbankserver, aber kleine einzelne Datenbanktabellen — Dies ist ein guter Ansatz, wenn die Größe des gesamten Datenbankservers mehrere TB beträgt, die Größe jeder unabhängigen Datenbanktabelle jedoch weniger als 1 TB beträgt. Um den Gesamtmigrationszeitraum zu verkürzen, können Sie unabhängige Tabellen einzeln migrieren.

  Lassen Sie uns ein Beispiel für eine Einzelbenutzerdatenbank verwenden, die 1 TB groß ist und die einzige Datenbank auf einem lokalen Datenbankserver ist. Es gibt 10 Tabellen in der Datenbank, von denen jede 100 GB groß ist. Sie können Backups jeder einzelnen Tabelle separat erstellen. Beim Wiederherstellen des Backups können Sie alle Tabellen auf einer Instanz parallel wiederherstellen.

- Eine Datenbank enthält sowohl transaktionale als auch nicht-transaktionale Workload-Tabellen. Ähnlich wie im vorherigen Anwendungsfall können Sie einen Split-Backup-Ansatz verwenden, wenn Sie sowohl transaktionale als auch nicht-transaktionale Workload-Tabellen in derselben Datenbank haben.

  Lassen Sie uns ein Beispiel für eine 2-TB-Datenbank verwenden, die aus 0,5 TB an Tabellen für kritische Workloads besteht, die für die Online-Transaktionsverarbeitung (OLTP) verwendet werden, und einer einzelnen 1,5-TB-Tabelle, die für die Archivierung alter Daten verwendet wird. Sie können die Sicherung aller Datenbankobjekte mit Ausnahme der Archivtabelle als konsistente Sicherung mit einer einzigen Transaktion durchführen. Dann erstellen Sie eine weitere, separate Sicherung nur der Archivtabelle. Für die Sicherung der Archivtabelle können Sie auch erwägen, mehrere parallel Sicherungen zu erstellen, indem Sie Bedingungen verwenden, um die Anzahl der Zeilen in der Sicherungsdatei aufzuteilen. Im Folgenden wird ein Beispiel gezeigt:

```
mysqldump -p your_db1 --tables your_table1 --where="column1 between 1 and 1000000 " >
 your_table1_part1.sql
mysqldump -p your_db1 --tables your_table1 --where="column1 between 1000001 and
 2000000 " > your_table1_part2.sql
mysqldump -p your_db1 --tables your_table1 --where="column1 > 2000000 " >
 your_table1_part3.sql
```

  Beim Wiederherstellen der Sicherungsdateien können Sie die transaktionale Workload-Backup und die Sicherung der Archivtabelle parallel wiederherstellen.

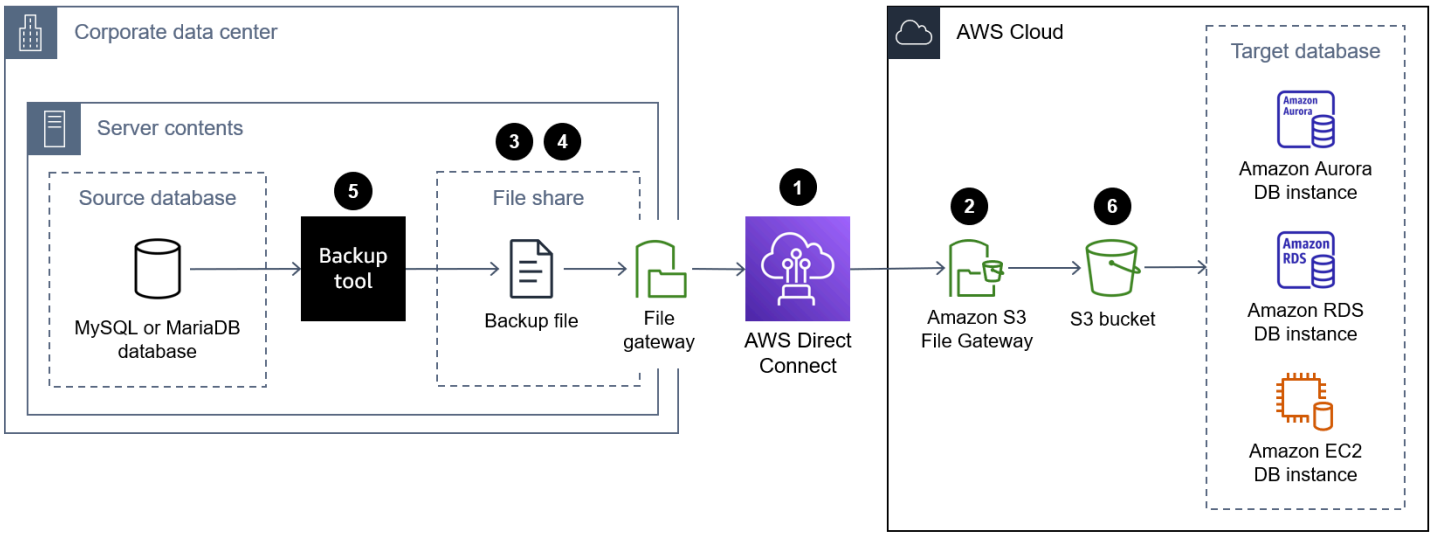
- Einschränkungen der Rechenressourcen — Wenn Sie auf dem lokalen Server über begrenzte Rechenressourcen wie CPU, Arbeitsspeicher oder Festplatte I/O verfügen, kann dies die Stabilität und Leistung bei der Erstellung des Backups beeinträchtigen. Anstatt ein vollständiges Backup zu erstellen, können Sie es in Teile aufteilen.

Beispielsweise kann ein lokaler Produktionsserver stark mit Workloads belastet sein und über begrenzte CPU-Ressourcen verfügen. Wenn Sie ein Backup einer Datenbank mit mehreren Terabyte auf diesem Server in einem einzigen Durchlauf erstellen, kann dies zusätzliche CPU-Ressourcen verbrauchen und sich negativ auf den Produktionsserver auswirken. Anstatt die gesamte Datenbanksicherung zu erstellen, teilen Sie die Sicherung in mehrere Teile auf, z. B. jeweils 2—3 Tabellen.

Verwenden von Amazon S3 File Gateway zum Übertragen von Backup-Dateien

Amazon S3 File Gateway verbindet Ihre lokale Umgebung über eine Dateischnittstelle mit Amazon Simple Storage Service (Amazon S3), sodass Sie Amazon S3-Objekte mithilfe branchenüblicher Dateiprotokolle wie Network File System (NFS) und Server Message Block (SMB) speichern und abrufen können. Es wurde als kostengünstige, skalierbare Lösung für die Speicherung von Daten in der Cloud konzipiert. Da Sie ihn zum Speichern von Datenbanksicherungsdateien verwenden können, kann Ihnen dieser Service bei der Migration großer, lokaler Datenbanken auf die AWS Cloud helfen. Sie könnten beispielsweise Amazon S3 File Gateway und Ihr bevorzugtes Datenbank-Backup-Tool verwenden, um die große MySQL- oder MariaDB-Datenbank direkt in einem Amazon S3 S3-Bucket zu sichern. Anschließend können Sie den S3-Bucket auf der Ziel-Instance bereitstellen und das Backup wiederherstellen.

Das folgende Diagramm zeigt die wichtigsten Schritte, die erforderlich sind, wenn Sie Amazon S3 File Gateway verwenden, um die Sicherungsdatei für eine lokale Datenbank in einen S3-Bucket in der AWS Cloud zu übertragen.



Im Folgenden finden Sie die Schritte zur Verwendung von Amazon S3 File Gateway zur Übertragung einer Datenbank-Backup-Datei von einem lokalen Rechenzentrum in einen S3-Bucket im AWS Cloud:

1. Connect das lokale Rechenzentrum mit dem, AWS Cloud indem Sie einen Dienst wie AWS Direct Connect oder AWS Site-to-Site VPN oder über eine öffentliche Internetverbindung verwenden.

2. Erstellen Sie ein S3 File Gateway. Anweisungen finden Sie unter Ihr Gateway erstellen.
3. Erstellen Sie eine NFS- oder SMB-Dateifreigabe, die vom S3 File Gateway gehostet wird. Anweisungen finden Sie unter Erstellen einer Dateifreigabe.
4. Mounten Sie die NFS- oder SMB-Dateifreigabe auf dem lokalen Server, der Ihre MySQL- oder MariaDB-Datenbank hostet. Anweisungen finden Sie unter Bereitstellen und Verwenden Ihrer Dateifreigabe.
5. Sichern Sie die lokale MySQL- oder MariaDB-Datenbank in dem Verzeichnis, in dem die NFS-Dateifreigabe bereitgestellt ist. Sie können jedes der in diesem Handbuch beschriebenen Backup-Tools verwenden.
6. Stellen Sie die Datenbanksicherung auf der Zieldatenbankinstanz wieder her, indem Sie einen der in diesem Handbuch beschriebenen Ansätze verwenden.

Vorteile

- Indem Sie Datenbank-Backups direkt im S3-Bucket erstellen und das Backup auf der Ziel-DB-Instance direkt aus demselben S3-Bucket wiederherstellen, können Sie den End-to-End-Migrationsprozess erheblich beschleunigen.
- Datenbank-Backup-Dateien werden dauerhaft in Amazon S3 gespeichert, und Sie wählen die Lifecycle-Management-Richtlinie und die S3-Speicherklasse.

Einschränkungen

Bei der Verwendung von Amazon S3 File Gateway-Dateifreigaben gelten folgende Einschränkungen:

- Die maximale Anzahl von Dateifreigaben pro Gateway beträgt 50.
- Um Lese- und Schreibkonflikte zu vermeiden, wenn mehrere Dateifreigaben denselben S3-Bucket verwenden, müssen Sie jede Dateifreigabe so konfigurieren, dass sie einen eindeutigen Präfixnamen verwendet.
- Die maximale Größe einer einzelnen Datei beträgt 5 TB, was der maximalen Größe jedes einzelnen Objekts in Amazon S3 entspricht.
- Die maximale Pfadlänge beträgt 1024 Zeichen.
- Windows-ACLs werden nur für Dateifreigaben unterstützt, die für Active Directory aktiviert sind, wenn Sie Windows SMB-Clients für den Zugriff auf die Dateifreigaben verwenden.

- Amazon S3 File Gateway unterstützt maximal 10 ACL-Einträge für jede Datei und jedes Verzeichnis.
- Die Root-ACL-Einstellungen von SMB-Dateifreigaben befinden sich nur auf dem Gateway. Diese Einstellungen bleiben bei Gateway-Updates und Neustarts erhalten.

> **Note**
>
> Wenn Sie die ACLs im Stammverzeichnis statt im übergeordneten Ordner unter dem Stammverzeichnis konfigurieren, sind die ACL-Berechtigungen in Amazon S3 nicht dauerhaft.

Bewährte Methoden

Weitere Informationen zu den bewährten Methoden für Amazon S3 File Gateway finden Sie unter Bewährte Methoden in der S3 File Gateway-Dokumentation.

Bewährte Methoden für die Migration großer MySQL- und MariaDB-Datenbanken

Lesen Sie zusätzlich zu den toolspezifischen Best Practices, die für jede Migrationsoption aufgeführt sind, auch die folgenden allgemeinen bewährten Methoden. Diese bewährten Methoden gelten für die Migration großer MySQL- und MariaDB-Datenbanken mit mehreren Terabyte, unabhängig vom ausgewählten Tool:

- Stellen Sie sicher, dass in den Quell- und Zieldatenbanken ausreichend Speicherplatz vorhanden ist, um das Backup zu erstellen und wiederherzustellen.
- Erstellen Sie keine sekundären Indizes auf der Zieldatenbank-Instance, bis die Migration abgeschlossen ist. Sekundäre Indizes erhöhen den Wartungsaufwand beim Import und können den Importvorgang verlangsamen.
- Wenn Sie einen Multithread-Ansatz verwenden, wählen Sie die richtige Anzahl von Threads. Für den Export empfehlen wir, einen Thread für jeden CPU-Kern zu verwenden. Für den Import empfehlen wir, einen Thread für jeweils zwei CPU-Kerne zu verwenden.
- Datendumps werden häufig von aktiven Datenbankservern aus ausgeführt, die Teil einer unternehmenskritischen Produktionsumgebung sind. Wenn der Datendump die Leistung erheblich beeinträchtigt und dies in Ihrer Umgebung nicht akzeptabel ist, sollten Sie eine der folgenden Optionen in Betracht ziehen:
  - Der Quellserver verfügt über Replikate. Sie können Daten von einem der Replikate sichern.
  - Der Quellserver wird durch regelmäßige Backup-Verfahren abgedeckt:
    - Wenn das Backup-Format für den direkten Import in die Zieldatenbank geeignet ist, verwenden Sie die Backup-Daten als Eingabe für den Importvorgang.
    - Wenn das Backup-Format nicht für den direkten Import in die Zieldatenbank geeignet ist, verwenden Sie das Backup, um eine temporäre Datenbank bereitzustellen und Daten daraus zu sichern.
  - Wenn Replikate und Backups nicht verfügbar sind:
    - Führen Sie Dumps außerhalb der Spitzenzeiten durch, wenn der Produktionsverkehr am geringsten ist.
    - Reduzieren Sie die Parallelität von Dump-Vorgängen, sodass der Server über genügend freie Kapazität für den Produktionsdatenverkehr verfügt.
- Erstellen Sie nur Dumps von vom Benutzer erstellten Datenbanken.

- Erstellen Sie die Benutzer in der Zieldatenbank neu und konfigurieren Sie ihre Berechtigungen. Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für Amazon RDS, Identitäts- und Zugriffsmanagement für Amazon Aurora oder Identitäts- und Zugriffsmanagement für Amazon EC2.
- Wenn Sie einen großen Datenbankserver migrieren, der aus mehreren unabhängigen Datenbanken besteht, erstellen Sie für jede Datenbank eine separate Instanz. Auf diese Weise können Sie die Datenbank effizienter verwalten und die Ressourcenbereitstellung verbessern, und die separaten Rechenressourcen können die Datenbankleistung verbessern.

Ressourcen

AWS Präskriptive Leitlinien

- Portfolio-Leitfaden für große Migrationen AWS
- Migrationsstrategie für relationale Datenbanken
- Migrieren Sie eine lokale MySQL-Datenbank zu Amazon RDS for MySQL
- Richten Sie die Datenreplikation zwischen Amazon RDS for MySQL und MySQL auf Amazon EC2 mithilfe von GTID ein
- Migrieren Sie eine lokale MariaDB-Datenbank mit nativen Tools zu Amazon RDS for MariaDB

AWS Blog-Beiträge

- Bewährte Sicherheitsmethoden für Amazon RDS for MySQL- und MariaDB-Instances
- Migrieren Sie selbstverwaltetes MariaDB zu Amazon Aurora MySQL

Ressourcen für die Wiederherstellung des Backups

- Einen Bucket erstellen (Amazon S3 S3-Dokumentation)
- Mit SSH eine Verbindung zu Ihrer Linux-Instance herstellen (EC2 Amazon-Dokumentation)
- Konfiguration der AWS CLI (AWS CLI Dokumentation)
- Befehl sync (AWS CLI Befehlsreferenz)
- Erstellen einer IAM-Richtlinie für den Zugriff auf Amazon S3 S3-Ressourcen (Aurora-Dokumentation)
- Voraussetzungen für DB-Cluster (Aurora-Dokumentation)
- Arbeiten mit DB-Subnetzgruppen (Aurora-Dokumentation)
- Erstellen einer VPC-Sicherheitsgruppe für einen privaten DB-Cluster (Aurora-Dokumentation)
- Wiederherstellen eines Aurora MySQL-DB-Clusters aus einem Amazon S3 S3-Bucket (Aurora-Dokumentation)
- Einrichtung der Replikation mit MySQL oder einem anderen Aurora-DB-Cluster (Aurora-Dokumentation)
- rds_set_external_master-Verfahren (Amazon RDS-Dokumentation)

- rds_start_replication-Verfahren (Amazon RDS-Dokumentation)

AWS Marketing

- Amazon Aurora
- Amazon RDS für MariaDB
- Amazon RDS für MySQL
- Amazon S3 File Gateway

Sonstige Ressourcen

- Percona XtraBackup
- MyDumper
- Mein SQL-Dump
- mysql-Pumpe

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
Verfügbarkeit von mysqlpump	mysqlpump wurde in MySQL Version 8.4 entfernt. Wir haben die Abschnitte mysqldump und mysqlpump aktualisiert, um dieser Änderung der Verfügbarkeit Rechnung zu tragen.	21. November 2024
MyDumper bewährte Verfahren	Wir haben die bewährten Methoden aktualisiert MyDumper , um Informationen zur Migration verschlüsselter Datenbanktabellen hinzuzufügen.	24. Oktober 2024
Percona-Versionen XtraBackup	Im XtraBackup Abschnitt Percona haben wir die Anweisungen aktualisiert, sodass sie den Versionen von Percona entsprechen, die von Amazon Aurora MySQL und Amazon RDS XtraBackup unterstützt werden.	3. August 2023
Erste Veröffentlichung	—	06. April 2023

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.