Migrar bancos de dados para o Amazon Aurora

Junho de 2016

Avisos

Este documento é fornecido apenas para fins informativos. Ele relaciona as atuais ofertas de produtos e práticas da AWS na data de emissão deste documento, que estão sujeitas a alterações sem aviso prévio. Os clientes são responsáveis por fazer sua própria avaliação independente das informações deste documento e de qualquer uso dos produtos ou serviços da AWS, e cada um deles é fornecido "como está", sem garantia de qualquer tipo, expressa ou implícita. Este documento não cria quaisquer garantias, representações, compromissos contratuais, condições ou seguros da AWS, suas afiliadas, fornecedores ou licenciadores. As responsabilidades da AWS com seus clientes são controladas pelos contratos da AWS, e este documento não modifica nem faz parte de qualquer contrato entre a AWS e seus clientes.

Índice

Resumo

O Amazon Aurora é um mecanismo de banco de dados relacional de nível corporativo e compatível com o MySQL. O Amazon Aurora é um banco de dados nativo em nuvem que supera muitas das limitações dos tradicionais mecanismos de banco de dados relacional. O objetivo deste whitepaper é destacar as melhores práticas de migração de bancos de dados existentes para o Amazon Aurora. Ele apresenta considerações sobre a migração e o passo-a-passo para migrar bancos de dados comerciais e de código aberto para o Amazon Aurora com o mínimo de interrupção para os aplicativos.

Introdução ao Amazon Aurora

Durante décadas, os tradicionais bancos de dados relacionais foram a principal escolha para o armazenamento e a persistência de dados. Esses sistemas de banco de dados continuam a se valer de arquiteturas monolíticas e não foram projetados para aproveitar a infraestrutura de nuvem. Essas arquiteturas monolíticas apresentam muitos desafios, especialmente em áreas como custo, flexibilidade e disponibilidade. Para solucionar esses desafios, a AWS reprojetou o banco de dados relacional para a infraestrutura de nuvem e introduziu o Amazon Aurora.

O Amazon Aurora é um mecanismo de banco de dados relacional compatível com o MySQL, que combina a velocidade, a disponibilidade e a segurança de bancos de dados comerciais avançados com a simplicidade e a economia dos bancos de dados de código aberto. O Aurora proporciona um desempenho até cinco vezes melhor do que o MySQL e desempenho comparável aos bancos de dados comerciais avançados. O Amazon Aurora é oferecido por um décimo do preço dos mecanismos comerciais.

O Amazon Aurora está disponível por meio da plataforma Amazon Relational Database Service (Amazon RDS). Assim como ocorre com outros bancos de dados da plataforma Amazon RDS, o Aurora é um banco de dados gerenciado. Com a plataforma Amazon RDS, a maioria das tarefas de gerenciamento de banco de dados, como provisionamento de hardware, correção de software, instalação, configuração, monitoramento e backup, é completamente automatizada.

O Amazon Aurora foi criado para lidar com cargas de trabalho essenciais e possui uma alta disponibilidade por padrão. Um cluster de banco de dados do Aurora abrange várias Zonas de disponibilidade em uma região, fornecendo durabilidade imediata e tolerância a falhas para seus dados em datacenters físicos. Uma Zona de disponibilidade é composta por um ou mais datacenters de alta disponibilidade operados pela Amazon. As Zonas de disponibilidade são isoladas umas das outras e são conectadas por meio de links de baixa latência. Cada segmento do volume de seu banco de dados é replicado seis vezes por essas Zonas de disponibilidade.

Os volumes de cluster do Aurora crescem automaticamente à medida que a quantidade de dados em seu banco de dados aumenta sem qualquer impacto no desempenho ou na disponibilidade. Assim, não há necessidade de estimar e provisionar uma grande quantidade de armazenamento do banco de dados antecipadamente. Um volume de cluster do Aurora pode crescer até um tamanho máximo de 64 terabytes (TB). Você só paga pelo espaço que usar em um volume de cluster do Aurora.

O recurso de backup automatizado do Aurora oferece suporte à recuperação point-in-time de seus dados, permitindo que você restaure o banco de dados a qualquer instante durante o período de retenção, até os últimos cinco minutos. Os backups automáticos são armazenados no Amazon Simple Storage Service (Amazon S3), que foi projetado para fornecer uma durabilidade de 99,999999999%. Os backups do Amazon Aurora são automáticos, incrementais e contínuos, e não afetam o desempenho do banco de dados.

Para aplicativos que precisam de réplicas somente leitura, você pode criar até 15 réplicas do Aurora por banco de dados do Aurora com um atraso de réplica muito baixo. Essas réplicas compartilham o mesmo armazenamento subjacente da instância de origem, reduzindo os custos e evitando a necessidade de executar gravações com os nós de réplica.

O Amazon Aurora é altamente seguro e permite criptografar bancos de dados usando chaves que você cria e controlar por meio do AWS Key Management Service (AWS KMS). Em uma instância de banco de dados em execução com a criptografia do Amazon Aurora, os dados ociosos no armazenamento subjacente são criptografados, bem como os backups automatizados, os snapshots e as réplicas no mesmo cluster. O Amazon Aurora usa SSL (AES-256) para proteger os dados em trânsito.

Para obter uma lista completa de recursos do Aurora, consulte a página de produtos do Amazon Aurora.[1] Considerando o rico conjunto de recursos e a economia proporcionada pelo Amazon Aurora, esta solução tem sido cada vez mais vista como o banco de dados preferencial para aplicativos essenciais.

Considerações sobre a migração do banco de dados

Um banco de dados representa um componente fundamental na arquitetura da maioria dos aplicativos. A migração do banco de dados para uma nova plataforma é um evento importante no ciclo de vida de um aplicativo e pode ter um impacto na funcionalidade, no desempenho e na confiabilidade dele. Você deve levar em conta alguns pontos importantes antes de embarcar em seu primeiro projeto de migração para o Amazon Aurora.

Fases de migração

Como as migrações de banco de dados tendem a ser complexas, recomendamos uma abordagem iterativa e em fases.

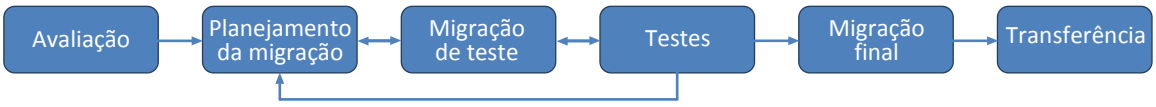

Figura 1: fases de migração

Considerações sobre aplicativos

Avaliar os recursos do Aurora

Embora a maioria dos aplicativos possa ser projetada para funcionar com vários mecanismos de banco de dados relacional, você deve se certificar de que o seu aplicativo funciona com o Amazon Aurora. O Amazon Aurora foi projetado para ser compatível com fio com o MySQL 5.6. Portanto, a maior parte do código,

aplicativos, drivers e ferramentas que são usados atualmente com bancos de dados do MySQL podem ser usados com o Aurora com pouca ou nenhuma alteração.

No entanto, alguns recursos do MySQL, como o mecanismo de armazenamento MyISAM, não estão disponíveis com o Amazon Aurora. Além disso, devido à natureza gerenciada do serviço do Aurora, o acesso SSH aos nós de banco de dados é restrito, o que pode afetar sua capacidade de instalar ferramentas ou plug-ins de terceiros no host do banco de dados.

Considerações sobre desempenho

O desempenho do banco de dados é uma consideração fundamental ao migrar um banco de dados para uma nova plataforma. Portanto, muitos projetos de migração de banco de dados bem-sucedidos começam com avaliações de desempenho da nova plataforma de banco de dados. Embora a execução de referências TPC-C e Sysbench ofereça a você uma ideia razoável do desempenho geral do banco de dados, essas referências não emulam os padrões de acesso a dados de seus aplicativos. Para obter resultados mais úteis, teste o desempenho do banco de dados para cargas de trabalho urgentes executando suas consultas (ou subconjunto de suas consultas) diretamente na nova plataforma.

Considere estas estratégias:

- Se o seu banco de dados atual for o MySQL, migre para o Amazon Aurora deixando um tempo de inatividade e teste o desempenho do banco de dados com uma versão de teste ou intermediária de seu aplicativo ou reproduzindo a carga de trabalho de produção.

- Se você estiver em um mecanismo compatível com o MySQL, poderá copiar seletivamente as tabelas mais ocupadas para o Amazon Aurora e testar suas consultas para essas tabelas. Assim, você terá um bom ponto de partida. Naturalmente, a realização de testes após concluir a migração de dados proporciona uma visão completa do desempenho real de seu aplicativo na nova plataforma.

O Amazon Aurora oferece um desempenho comparável ao de mecanismos comerciais e uma melhoria considerável sobre o desempenho do MySQL. Isso se deve à integração total do mecanismo de banco de dados com uma camada de armazenamento virtualizada com base em SSD criada para cargas de trabalho de banco de dados. Como resultado, você reduz as gravações no sistema de

armazenamento, minimiza a contenção de bloqueio e elimina atrasos gerados pelos threads de processo do banco de dados. Nossos testes com o SysBench em instâncias r3.8xlarge mostram que o Amazon Aurora entrega mais de 585.000 leituras por segundo e 107.000 gravações por segundo – cinco vezes mais do que o MySQL executando a mesma referência no mesmo hardware.

Uma área em que o Amazon Aurora oferece melhorias significativas em relação ao MySQL tradicional é nas cargas de trabalho altamente simultâneas. Para maximizar a taxa de transferência de sua carga de trabalho no Amazon Aurora, recomendamos que você projete seus aplicativos para gerar um grande número de consultas simultâneas.

Considerações sobre réplica de leitura e fragmentação (sharding)

Se o seu banco de dados atual estiver fragmentado em vários nós, você pode ter a oportunidade de combinar esses fragmentos em um único banco de dados do Aurora durante a migração. Uma única instância do Amazon Aurora pode ser dimensionada para até 64 TB, oferecer suporte a milhares de tabelas e processar um número significativamente maior de leituras e gravações em comparação com um banco de dados padrão do MySQL.

Se o seu aplicativo exige muita leitura/gravação, considere usar réplicas de leitura do Aurora para transferir a carga de trabalho somente leitura do nó mestre do banco de dados. Isso pode melhorar a simultaneidade de seu banco de dados primário para gravações e o desempenho geral dos processos de leitura e gravação. O uso de réplicas de leitura também pode reduzir seus custos em uma configuração Multi-AZ, uma vez que você pode usar instâncias menores para sua instância principal enquanto adiciona recursos de recuperação de falhas ao seu cluster de banco de dados. As réplicas de leitura do Aurora oferecem um atraso de replicação quase nulo e você pode criar até 15 réplicas de leitura.

Considerações sobre confiabilidade

Uma consideração importante com os bancos de dados é a alta disponibilidade e recuperação de desastres. Determine o objetivo de tempo de recuperação (RTO – Recovery Time Objective) e o objetivo de ponto de recuperação (RPO – Recovery Point Objective) de seu aplicativo. Com o Amazon Aurora, você pode melhorar esses dois fatores consideravelmente.

O Amazon Aurora reduz os tempos de reinicialização do banco de dados para menos de 60 segundos na maioria dos cenários de falha de banco de dados. Além disso, o Aurora move o cache do buffer para fora do processo do banco de dados e o disponibiliza imediatamente no momento da reinicialização. Em raros cenários de falhas em hardware e na Zona de disponibilidade, a recuperação é automaticamente controlada pela plataforma de banco de dados.

O Aurora foi projetado para oferecer recuperação de RPO nulo dentro de uma região da AWS, o que é uma melhoria muito importante em sistemas de banco de dados no local. Ele também mantém seis cópias de seus dados em três Zonas de disponibilidade e tenta recuperar automaticamente seu banco de dados em uma Zona de disponibilidade saudável sem perda de dados. No caso improvável de seus dados estarem indisponíveis no armazenamento do Amazon Aurora, você pode restaurar a partir de um snapshot do DB ou realizar uma operação de restauração point-in-time para uma nova instância.

Considerações sobre licenças e custos

A aquisição e a execução de bancos de dados têm custos associados. Antes de planejar uma migração de banco de dados, é fundamental realizar uma análise do custo total de propriedade (TCO – Total Cost of Ownership) da nova plataforma de banco de dados. A migração para uma nova plataforma de banco de dados deve, idealmente, reduzir o custo total de propriedade e, ao mesmo tempo, fornecer aos seus aplicativos recursos semelhantes ou melhores. Se você estiver executando um mecanismo de banco de dados de código aberto (MySQL, Postgres), seus custos devem estar principalmente relacionados a hardware, gerenciamento de servidores e atividades de gerenciamento de banco de dados. No entanto, se você estiver executando um mecanismo de banco de dados comercial (Oracle, SQL Server, DB2 etc.), uma parte significativa de seu custo deve estar destinada ao licenciamento do banco de dados.

Como o Aurora está disponível por um décimo do custo de mecanismos comerciais, muitos dos aplicativos que estão migrando para o Aurora podem reduzir significativamente seu TCO. Mesmo que você esteja executando um mecanismo de código aberto, como o MySQL ou o Postgres, com o alto desempenho e as réplicas de leitura de dupla finalidade do Aurora, você poderá obter uma economia significativa ao migrar para o Amazon Aurora. Para saber mais, consulte a página de Definição de Preço do Amazon RDS para Aurora.[2]

Outras considerações sobre a migração

Depois de levar em conta os fatores de adequação do aplicativo, desempenho, TCO e confiabilidade, você deve pensar sobre o que seria necessário para migrar para a nova plataforma.

Estimar o esforço da mudança de código

É importante estimar a quantidade de mudanças que você precisa fazer no código e no schema ao migrar seu banco de dados para o Amazon Aurora. Se estiver migrando de bancos de dados compatíveis com o MySQL, as alterações de código serão insignificantes. No entanto, se estiver migrando de mecanismos não compatíveis com o MySQL, talvez você precise fazer alterações no código e no schema. A AWS Schema Conversion Tool, discutida mais adiante neste documento, pode ajudar a estimar esse esforço.

Disponibilidade do aplicativo durante a migração

Você tem a opção de migrar para o Amazon Aurora adotando uma abordagem de inatividade previsível com seu aplicativo ou adotando uma abordagem de inatividade quase nula. A abordagem escolhida depende do tamanho de seu banco de dados e dos requisitos de disponibilidade de seus aplicativos. Seja qual for o caso, é uma boa ideia considerar o impacto do processo de migração em seu aplicativo e em sua empresa antes de iniciar uma migração do banco de dados. As seções a seguir explicam ambas as abordagens em detalhes.

Planejar o processo de migração do banco de dados

A seção anterior discutiu algumas das principais considerações a serem feitas ao migrar bancos de dados para o Amazon Aurora. Depois que você determina que o Aurora é a melhor opção para seu aplicativo, a próxima etapa é escolher uma abordagem de migração preliminar e criar um plano de migração do banco de dados.

Migração homogênea

Se o seu banco de dados de origem for um banco de dados compatível com o MySQL 5.6 (MySQL, MariaDB, Percona etc.), então a migração para o Aurora será bastante simples.

Migração homogênea com tempo de inatividade

Se o seu aplicativo puder acomodar um tempo de inatividade previsível em horários fora de pico, a migração com tempo de inatividade será a opção mais simples e uma abordagem altamente recomendada. A maioria dos projetos de migração de banco de dados se encaixa nessa categoria, dado que boa parte dos aplicativos já tem uma janela de manutenção bem definida. Você tem as seguintes opções para migrar seu banco de dados com tempo de inatividade.

- **Migração de snapshot do RDS** – Se o seu banco de dados de origem estiver em execução no Amazon RDS MySQL 5.6, você poderá simplesmente migrar um snapshot do banco de dados para o Amazon Aurora. Para migrações com tempo de inatividade, ou você precisa interromper o aplicativo ou deve interromper a gravação no banco de dados enquanto o snapshot e a migração estão em andamento. O tempo para migrar depende principalmente do tamanho do banco de dados e pode ser determinado antes da migração da produção executando-se um teste de migração. A opção de migração de snapshot é explicada na seção Migração de snapshot do RDS. Você também pode fazer uma migração com tempo de inatividade quase nulo usando a migração de snapshot e a replicação binlog, descrita na próxima seção.
- **Migração usando ferramentas nativas do MySQL.** Você pode usar ferramentas nativas do MySQL para migrar seus dados e schema para o Aurora. Essa é uma ótima opção quando você precisa de mais controle sobre o processo de migração do banco de dados, se estiver mais à vontade para usar as ferramentas nativas do MySQL e outros métodos de migração não estiverem funcionando bem para seu caso de uso. Para melhores práticas ao usar esta opção, faça o download do whitepaper Amazon RDS for Aurora Export/Import Performance Best Practices. [3]
- **Migração usando o AWS Database Migration Service (AWS DMS).** Uma migração única usando o AWS DMS é outra forma de mover o banco de dados de origem para o Amazon Aurora. Antes que você possa usar o AWS DMS para mover os dados, é preciso copiar o schema do banco de dados da origem para o destino usando as ferramentas nativas do MySQL. Para ver o processo passo a passo, consulte a seção Migrar dados. O AWS DMS é uma ótima opção a ser usada quando você não tem experiência com as ferramentas nativas do MySQL.

Migração homogênea com tempo de inatividade quase nulo

Em alguns casos, convém migrar seu banco de dados para o Aurora com um tempo de inatividade mínimo. Aqui estão dois exemplos:

- Quando seu banco de dados é relativamente grande e o tempo de migração usando as opções de tempo de inatividade é mais do que sua janela de manutenção de aplicativos.
- Quando você deseja executar bancos de dados de origem e destino em paralelo para fins de teste.

Em tais casos, você pode replicar as alterações de seu banco de dados de origem do MySQL para o Aurora em tempo real usando a replicação. Você tem algumas opções:

- **Migração com tempo de inatividade quase nulo usando a replicação binlog do MySQL.** O Amazon Aurora oferece suporte à replicação binlog tradicional do MySQL. Se você estiver executando o banco de dados do MySQL, é possível que já esteja familiarizado com a configuração de replicação binlog clássica. Se esse for o caso e você quiser ter mais controle sobre o processo de migração, um carregamento único do banco de dados usando ferramentas nativas em conjunto com a replicação binlog oferecerá a você uma opção familiar para migrar para o Aurora.
- **Migração com tempo de inatividade quase nulo usando o AWS Database Migration Service (AWS DMS).** Além de oferecer suporte à migração única, o AWS DMS também é compatível com a replicação de dados em tempo real usando a captura de dados de alterações (CDC – Change Data Capture) da origem para o destino. O AWS DMS cuida das complexidades relacionadas à cópia de dados inicial, configurando instâncias de replicação e monitorando a replicação. Depois que a migração inicial do banco de dados é concluída, o banco de dados de destino permanece sincronizado com a origem pelo tempo que você quiser. Se você não estiver familiarizado com a replicação binlog, o AWS DMS será a próxima melhor opção para uma migração homogênea e com tempo de inatividade quase nulo para o Amazon Aurora. Consulte a seção Introdução e abordagem geral do AWS DMS.
- **Migração com tempo de inatividade quase nulo usando a migração de snapshot do RDS e a replicação binlog do MySQL.** Se o seu banco de dados de origem estiver em execução no Amazon RDS MySQL 5.6, você poderá migrar um snapshot desse banco de dados para o Amazon Aurora e

começar a replicação binlog entre origem e destino após a migração do snapshot. Para obter detalhes sobre essa opção de migração, consulte Replication with Amazon Aurora no *Amazon RDS User Guide*[4].

Migração heterogênea

Se você quiser migrar um banco de dados não compatível com o MySQL (Oracle, SQL Server, PostgreSQL etc.) para o Amazon Aurora, existem várias opções para ajudá-lo a realizar essa migração de forma rápida e fácil.

Migração de schema

É possível realizar a migração de schema de um banco de dados não compatível com o MySQL para o Amazon Aurora usando a AWS Schema Conversion Tool. Esta ferramenta consiste em um aplicativo de área de trabalho que ajuda você a converter seu schema de banco de dados do Oracle, Microsoft SQL Server ou PostgreSQL em uma instância de banco de dados do Amazon RDS MySQL ou um cluster de banco de dados do Amazon Aurora. Nos casos em que o schema de seu banco de dados de origem não puder ser completamente convertido de forma automática, a AWS Schema Conversion Tool fornecerá orientação sobre como criar o schema equivalente em seu banco de dados do Amazon RDS de destino. Para obter detalhes, consulte a seção Migrar o schema de banco de dados.

Migração de dados

Embora ofereça suporte a migrações homogêneas com tempo de inatividade quase nulo, o AWS Database Migration Service (AWS DMS) também oferece suporte à replicação contínua em bancos de dados heterogêneos e é uma opção preferencial para mover o banco de dados de origem para o banco de dados de destino, tanto em migrações com tempo de inatividade como em migrações com tempo de inatividade quase nulo. Uma vez que a migração foi iniciada, o AWS DMS gerencia todas as complexidades do processo de migração, como transformação de tipo de dados, compactação e transferência paralela (para uma transferência de dados mais rápida) e, ao mesmo tempo, garante que as mudanças de dados no banco de dados de origem que ocorrem durante o processo de migração sejam automaticamente replicadas no destino.

Além de usar o AWS DMS, você pode usar várias ferramentas de terceiros, como o Attunity Replicate, Tungsten Replicator, Oracle Golden Gate etc. para migrar seus dados para o Amazon Aurora. Qualquer que seja a ferramenta escolhida,

você deve levar em conta o desempenho e os custos de licenciamento antes de finalizar seu conjunto de ferramentas para migração.

Migrar bancos de dados grandes para o Amazon Aurora

A migração de grandes conjuntos de dados apresenta desafios únicos em cada projeto de migração do banco de dados. Muitos projetos bem-sucedidos de migração de banco de dados grandes usam uma combinação das seguintes estratégias:

- **Migração com replicação contínua.** Bancos de dados grandes normalmente têm requisitos maiores de tempo de inatividade ao transferir os dados da origem para o destino. Para reduzir o tempo de inatividade, você pode primeiro carregar os dados de base da origem para o destino e, em seguida, ativar a replicação (usando ferramentas nativas do MySQL, o AWS DMS ou ferramentas de terceiros) para as alterações.
- **Copiar tabelas estáticas primeiro.** Se o seu banco de dados se valer de grandes tabelas estáticas com dados de referência, você poderá migrar essas tabelas grandes para o banco de dados de destino antes de migrar seu conjunto de dados ativo. Você pode usar o AWS DMS para copiar tabelas seletivamente ou para exportar e importar essas tabelas manualmente.
- **Migração multifásica.** A migração de um banco de dados grande com milhares de tabelas pode ser dividida em várias fases. Por exemplo, você pode mover um conjunto de tabelas sem consultas de ligações cruzadas a cada semana até que o banco de dados de origem seja totalmente migrado para o banco de dados de destino. Lembre-se de que para isso você precisa fazer alterações em seu aplicativo para se conectar a dois bancos de dados simultaneamente enquanto o conjunto de dados estiver em dois nós distintos. Embora esse não seja um padrão de migração comum, ainda é uma opção.
- **Limpeza de banco de dados.** Muitos bancos de dados grandes contêm dados e tabelas que permanecem inutilizados. Em muitos casos, os desenvolvedores e administradores de bancos de dados (DBAs – Database Administrators) mantêm cópias de backup das tabelas no mesmo banco de dados ou simplesmente se esquecem de transferir as tabelas não usadas. Qualquer que seja o motivo, um projeto de migração de banco de dados oferece uma oportunidade para limpar o banco de dados existente antes da

migração. Se algumas tabelas não estiverem sendo usadas, você poderá transferi-las ou arquivá-las em outro banco de dados. Você também poderá excluir dados antigos de tabelas grandes ou arquivar esses dados em arquivos simples.

Consolidação de fragmentos e partições no Amazon Aurora

Se você estiver executando vários fragmentos ou partições funcionais de seu banco de dados para obter um alto desempenho, poderá ter a oportunidade de consolidar essas partições ou fragmentos em um único banco de dados do Aurora. Uma única instância do Amazon Aurora pode ser dimensionada para até 64 TB, oferecer suporte a milhares de tabelas e processar um número significativamente maior de leituras e gravações em comparação com um banco de dados padrão do MySQL. A consolidação dessas partições em uma única instância do Aurora não apenas reduz o custo total de propriedade e simplifica o gerenciamento do banco de dados, mas também melhora significativamente o desempenho das consultas de partição cruzada.

- **Partições funcionais.** O particionamento funcional significa dedicar nós diferentes a tarefas diferentes. Por exemplo, em um aplicativo de comércio eletrônico, você pode ter um nó do banco de dados dedicado aos dados do catálogo de produtos e outro nó dedicado à captura e ao processamento de pedidos. Como resultado, essas partições normalmente têm esquemas distintos e não sobrepostos.

  Estratégia de consolidação. Migre cada partição funcional como um schema distinto para sua instância de destino do Aurora. Se o seu banco de dados de origem for compatível com o MySQL, use as ferramentas nativas do MySQL para migrar o schema e, em seguida, use o AWS DMS para migrar os dados, uma vez ou continuamente por meio da replicação. Se o seu banco de dados de origem não for compatível com o MySQL, use a AWS Schema Conversion Tool para migrar os esquemas para o Aurora e o AWS DMS para fazer um carregamento único ou replicação contínua.

- **Fragmentos de dados.** Se você possui o mesmo schema com diferentes conjuntos de dados em vários nós, então está usando a fragmentação de banco de dados. Por exemplo, um serviço de blogs de alto tráfego pode fragmentar os dados e a atividade de usuários em vários fragmentos de banco de dados enquanto mantém o mesmo schema de tabela.

Estratégia de consolidação. Uma vez que todos os fragmentos compartilham o mesmo schema de banco de dados, você só precisa criar o schema de destino uma vez. Se você estiver usando um banco de dados compatível com o MySQL, use ferramentas nativas para migrar o schema do banco de dados para o Aurora. Se você estiver usando um banco de dados não compatível com o MySQL, use a AWS Schema Conversion Tool para migrar o schema do banco de dados para o Aurora. Assim que o schema do banco de dados tiver sido migrado, será melhor interromper as gravações nos fragmentos do banco de dados e usar ferramentas nativas ou um carregamento de dados único do AWS DMS para migrar um fragmento individual para o Aurora. Se as gravações no aplicativo não puderem ser interrompidas por um período de tempo prolongado, você ainda poderá usar o AWS DMS com replicação, mas somente após o devido planejamento e testes.

Resumo das opções de migração

Tipo de banco de dados de origem	Migração com tempo de inatividade	Migração com tempo de inatividade quase nulo
Amazon RDS MySQL	Opção 1: migração de snapshot do RDS Opção 2: migração manual usando ferramentas nativas* Opção 3: migração de schema usando ferramentas nativas e carregamento de dados do AWS DMS	Opção 1: migração usando ferramentas nativas + replicação binlog Opção 2: migração de snapshot do RDS + replicação binlog Opção 3: migração de schema usando ferramentas nativas + AWS DMS para mover os dados
MySQL Amazon EC2 ou local	Opção 1: migração usando ferramentas nativas Opção 2: migração de schema usando ferramentas nativas + AWS DMS para carregamento de dados	Opção 1: migração usando ferramentas nativas + replicação binlog Opção 2: migração de schema usando ferramentas nativas + AWS DMS para mover os dados
Oracle/SQL Server	Opção 1: AWS Schema Conversion Tool + AWS DMS (recomendado) Opção 2: processo manual ou ferramenta de terceiros para conversão de schema + processo manual ou ferramenta de terceiros para carregar os dados no destino	Opção 1: AWS Schema Conversion Tool + AWS DMS (recomendado) Opção 2: processo manual ou ferramenta de terceiros para conversão de schema + processo manual ou ferramenta de terceiros para carregar os dados no destino + ferramenta de terceiros para replicação

Tipo de banco de dados de origem	Migração com tempo de inatividade	Migração com tempo de inatividade quase nulo
Outros bancos de dados que não são MySQL	Opção: processo manual ou ferramenta de terceiros para conversão de schema + processo manual ou ferramenta de terceiros para carregar os dados no destino	Opção: processo manual ou ferramenta de terceiros para conversão de schema + processo manual ou ferramenta de terceiros para carregar os dados no destino + ferramenta de terceiros para replicação (GoldenGate etc.)

*Ferramentas nativas do MySQL: mysqldump, SELECT INTO OUTFILE; ferramentas de terceiros como mydumper/myloader

Migração de snapshot do RDS

Para usar a migração de snapshot do RDS para migrar para o Aurora, seu banco de dados do MySQL deve estar em execução no Amazon RDS MySQL 5.6 e você deve fazer um snapshot do RDS do banco de dados. Este método de migração não funciona com bancos de dados no local ou bancos de dados em execução no Amazon Elastic Compute Cloud (Amazon EC2). Além disso, se você estiver executando o banco de dados do Amazon RDS MySQL em uma versão anterior a 5.6, será preciso fazer o upgrade para a versão 5.6 como pré-requisito.

A maior vantagem deste método de migração é que ele é o mais simples e requer o menor número de etapas. Mais especificamente, ele migra todos os objetos de schema, índices secundários e procedimentos armazenados junto com todos os dados do banco de dados.

Durante a migração de snapshot sem replicação binlog, seu banco de dados de origem deve estar off-line ou em um modo somente leitura (para que nenhuma alteração seja feita no banco de dados de origem durante a migração). Para estimar o tempo de inatividade, você pode simplesmente usar o snapshot existente de seu banco de dados para fazer um teste de migração. Se o tempo da migração estiver de acordo com seus requisitos de tempo de inatividade, esse pode ser o melhor método para você. Observe que, em alguns casos, a migração usando o AWS DMS ou ferramentas de migração nativas pode ser mais rápida do que a migração de snapshot.

Se você não puder tolerar um período de inatividade prolongado, ainda poderá fazer uma migração com tempo de inatividade quase nulo migrando primeiro um snapshot do RDS do banco de dados para o Amazon Aurora enquanto permite

que o banco de dados de origem continue a ser atualizado. Em seguida, você o atualizará usando a replicação binlog do MySQL para o Aurora.

Você pode migrar um snapshot do DB manual ou automático. As etapas gerais a serem realizadas são as seguintes:

1. Determine a quantidade de espaço necessária para migrar sua instância do Amazon RDS MySQL 5.6 para um cluster de banco de dados do Aurora. Para obter mais informações, consulte a próxima seção.
2. Use o console do Amazon RDS para criar o snapshot na região onde está localizada a instância do Amazon RDS MySQL 5.6.
3. Use o recurso **Migrar banco de dados** no console para criar um cluster de banco de dados do Amazon Aurora que será preenchido usando o snapshot do DB da instância de banco de dados original do MySQL 5.6.

Observação: Algumas tabelas MyISAM não podem não ser convertidas sem erros e talvez precisem de alterações manuais. Por exemplo, o mecanismo InnoDB não permite que um campo de incrementação automática faça parte de uma chave composta. Além disso, no momento não oferecemos suporte a índices espaciais.

Estimar requisitos de espaço para a migração de snapshot

Quando você migra um snapshot de uma instância de banco de dados do MySQL para um cluster de banco de dados do Aurora, o Aurora usa um volume do Amazon Elastic Block Store (Amazon EBS) para formatar os dados do snapshot antes de migrá-los. Há alguns casos que exigem um espaço adicional para formatar os dados para migração. Os dois recursos que podem causar problemas de espaço durante a migração são as tabelas MyISAM e o uso da opção `ROW_FORMAT=COMPRESSED`. Se você não estiver usando esses recursos em seu banco de dados de origem, pode ignorar esta seção porque provavelmente não terá problemas de espaço. Durante a migração, as tabelas MyISAM serão convertidas para InnoDB e todas as tabelas compactadas serão descompactadas. Consequentemente, deverá haver um espaço adequado para as cópias adicionais dessas tabelas.

O tamanho do volume de migração baseia-se no tamanho alocado para o banco de dados de origem do MySQL do qual o snapshot foi feito. Portanto, se você tiver tabelas MyISAM ou compactadas que compõem uma pequena porcentagem do tamanho total do banco de dados e houver espaço disponível no banco de dados

original, sua migração deverá acontecer sem que você encontre problemas de espaço. No entanto, se o banco de dados original não tiver espaço suficiente para armazenar uma cópia das tabelas MyISAM convertidas, bem como outras cópias (descompactadas) de tabelas compactadas, o volume de migração não será grande o suficiente. Nesse caso, você precisa modificar o banco de dados de origem do Amazon RDS MySQL, aumentando o tamanho dele para criar espaço para as cópias adicionais dessas tabelas, fazer um novo snapshot do banco de dados e, em seguida, migrar o novo snapshot.

Ao migrar os dados para seu cluster de banco de dados, observe as seguintes diretrizes e limitações:

- Embora o Amazon Aurora ofereça até 64 TB de espaço de armazenamento, o processo de migração de um snapshot para um cluster de banco de dados do Aurora é limitado pelo tamanho do volume do Amazon EBS do snapshot e, portanto, limita-se a um tamanho máximo de 6 TB.
- As tabelas no banco de dados de origem que não são MyISAM podem ter um tamanho de até 6 TB. No entanto, devido aos requisitos de espaço adicional durante a conversão, verifique se nenhuma das tabelas MyISAM e compactadas que estão sendo migradas de sua instância de banco de dados do MySQL excede 3 TB.

Você pode modificar o schema de banco de dados (converter as tabelas MyISAM para InnoDB e remover `ROW_FORMAT = COMPRESSED`) antes de migrá-lo para o Amazon Aurora. Isso pode ser útil nos seguintes casos:

- Você quer acelerar o processo de migração.
- Você não tem certeza de quanto espaço precisa para provisionar.
- Você tentou migrar seus dados e a migração falhou devido à falta de espaço provisionado.

Verifique se você não está fazendo essas alterações em seu banco de dados de produção do Amazon RDS MySQL, mas sim em uma instância de banco de dados que foi restaurada de seu snapshot de produção. Para obter mais informações sobre isso, consulte Reducing the Amount of Space Required to Migrate Data into Amazon Aurora no *Amazon RDS User Guide*.[5]

Migrar um snapshot do DB usando o console

Você pode migrar um snapshot do DB de uma instância de banco de dados do Amazon RDS MySQL para criar um cluster de banco de dados do Aurora. O novo cluster de banco de dados será preenchido com os dados da instância de banco de dados original do Amazon RDS MySQL. O snapshot do DB deve ter sido feito a partir de uma instância de banco de dados do RDS executando o MySQL 5.6 e não deve estar criptografado. Para obter mais informações sobre como criar um snapshot do banco de dados, consulte Creating a DB Snapshot no *Amazon RDS User Guide*.[6]

Se o snapshot do DB não estiver na região onde você deseja localizar seu cluster de banco de dados do Aurora, use o console do Amazon RDS para copiar o snapshot do banco de dados para essa região. Para obter mais informações sobre como copiar um snapshot do banco de dados, consulte Copying a DB Snapshot no *Amazon RDS User Guide*.[7]

Para migrar um snapshot do DB do MySQL 5.6 usando o console, faça o seguinte:

1. Faça login no Console de Gerenciamento da AWS e abra o console do Amazon RDS em https://console.aws.amazon.com/rds/.
2. Selecione **Snapshots**.
3. Na página **Snapshots**, selecione o snapshot do Amazon RDS MySQL 5.6 que você deseja migrar para um cluster de banco de dados do Aurora.
4. Selecione **Migrate Database**.
5. Na página **Migrate Database**, especifique os valores correspondentes ao seu ambiente e os requisitos de processamento, conforme mostrado na ilustração a seguir. Para obter descrições dessas opções, consulte Migrating a DB Snapshot by Using the Console no *Amazon RDS User Guide*.[8]

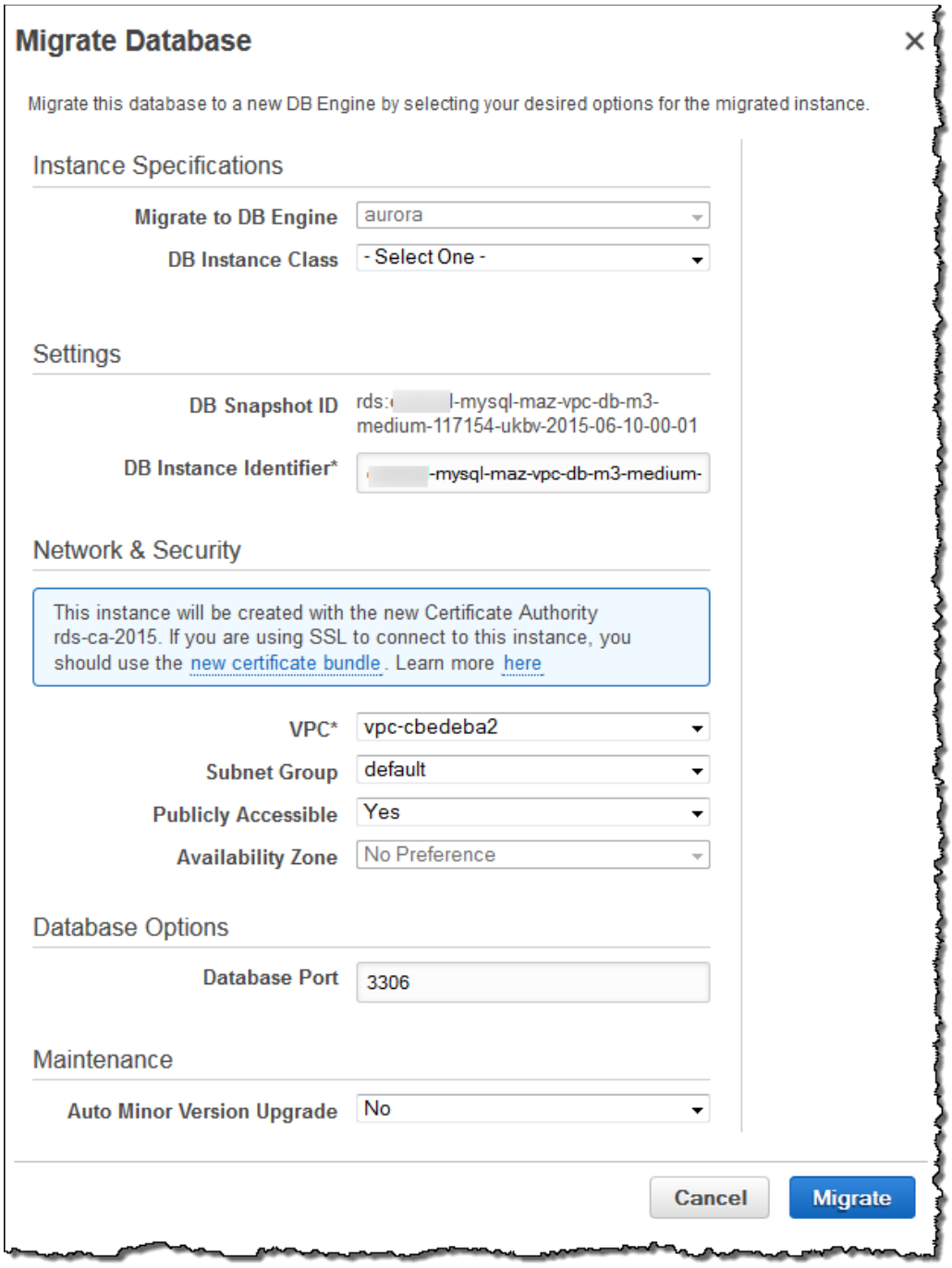

Figura 2: migração de snapshot

6. Clique em **Migrar** para migrar o snapshot do DB.

Na lista de instâncias, clique no ícone de seta apropriado para mostrar os detalhes do cluster de banco de dados e monitorar o progresso da migração. Este painel de detalhes exibe o endpoint do cluster usado para se conectar à instância principal do banco de dados do cluster. Para obter mais informações sobre como

se conectar a um cluster de banco de dados do Amazon Aurora, consulte Connecting to an Amazon Aurora DB Cluster no *Amazon RDS User Guide*.[9]

Migrar o schema de banco de dados

A migração do snapshot do DB do RDS migra os dados e o schema completos para a nova instância do Aurora. No entanto, se a localização de seu banco de dados de origem ou seus requisitos de tempo de atividade do aplicativo não permitirem o uso da migração de snapshot do RDS, primeiro será preciso migrar o schema do banco de dados de origem para o banco de dados de destino para que você possa mover os dados reais. Schema de banco de dados é uma estrutura de esqueleto que representa a visualização lógica de todo o banco de dados e geralmente inclui o seguinte:

- Objetos de armazenamento de banco de dados: tabelas, colunas, restrições, índices, sequências, tipos definidos pelo usuário e tipos de dados.
- Objetos de código do banco de dados: funções, procedimentos, pacotes, gatilhos, exibições, visualizações materializadas, eventos, funções escalares SQL, funções em linha SQL, funções de tabela SQL, atributos, variáveis, constantes, tipos de tabela, tipos públicos, tipos privados, cursores, exceções, parâmetros e outros objetos.

Na maioria dos casos, o schema de banco de dados permanece relativamente estático e, portanto, não é necessário programar um tempo de inatividade durante a etapa de migração do schema de banco de dados. O schema do banco de dados de origem pode ser extraído enquanto este está ativo e em execução sem afetar o desempenho. Se o seu aplicativo ou os desenvolvedores fizerem alterações frequentes no schema do banco de dados, verifique se essas alterações estão pausadas enquanto a migração está em andamento ou se são levadas em conta durante o processo de migração de schema.

Dependendo do tipo de banco de dados de origem, você poderá usar as técnicas discutidas nas seções a seguir para migrar o schema do banco de dados. Como pré-requisito para a migração de schema, você deve ter um banco de dados de destino do Aurora criado e disponível.

Migração de schema homogênea

Se o seu banco de dados de origem for compatível com o MySQL 5.6 e estiver em execução no Amazon RDS, no Amazon EC2 ou fora da AWS, você pode usar as ferramentas nativas do MySQL para exportar e importar o schema.

- **Exportar o schema do banco de dados.** Você pode usar o utilitário de cliente mysqldump para exportar o schema do banco de dados. Para executar esse utilitário, conecte-se ao seu banco de dados de origem e redirecione a saída do comando `mysqldump` para um arquivo simples. A opção `-no-data` garante que apenas o schema do banco de dados seja exportado sem quaisquer dados reais da tabela. Para ver a referência completa do comando `mysqldump`, consulte mysqldump—A Database Backup Program.[10]

```
mysqldump –u source_db_username –p --no-data --routines --
triggers –databases source_db_name > DBSchema.sql
```

- **Importar o schema do banco de dados para o Aurora.** Para importar o schema para sua instância do Aurora, conecte-se ao banco de dados do Aurora por um cliente de linha de comando do MySQL (ou um cliente correspondente do Windows) e direcione o conteúdo do arquivo de exportação para o MySQL.

```
mysql –h aurora-cluster-endpoint -u username -p <
DBSchema.sql
```

Observe o seguinte:

- Se o seu banco de dados de origem contiver procedimentos armazenados, gatilhos e exibições, será preciso remover a sintaxe `DEFINER` de seu arquivo de dump. Um simples comando Perl para fazer isso é indicado abaixo. Essa ação cria todos os gatilhos, exibições e procedimentos armazenados com o usuário conectado no momento como `DEFINER`. Lembre-se de avaliar as possíveis implicações de segurança dessa ação.

```
$perl -pe 's/\sDEFINER=`[^`]+`@`[^`]+`//' < DBSchema.sql >
DBSchemaWithoutDEFINER.sql
```

- O Amazon Aurora é compatível somente com tabelas InnoDB. Se você tiver tabelas MyISAM em seu banco de dados de origem, o Aurora alterará automaticamente o mecanismo para InnoDB quando o comando `CREATE TABLE` for executado.
- O Amazon Aurora não oferece suporte a tabelas compactadas (ou seja, tabelas criadas com `ROW_FORMAT=COMPRESSED`). Se você tiver tabelas compactadas em seu banco de dados de origem, o Aurora alterará automaticamente `ROW_FORMAT` para `COMPACT` quando o comando `CREATE TABLE` for executado.

Depois que você tiver importado com êxito o schema do banco de dados de origem compatível com o MySQL 5.6 para o Amazon Aurora, a próxima etapa será copiar os dados reais da origem para o destino. Para obter mais informações, consulte Introdução e abordagem geral do AWS DMS mais adiante neste documento.

Migração de schema heterogênea

Se o seu banco de dados de origem não for compatível com o MySQL, será preciso converter seu schema em um formato compatível com o Amazon Aurora.
A conversão do schema de um mecanismo de banco de dados para outro é uma tarefa incomum e pode ser preciso reescrever determinadas partes de seu banco de dados e código do aplicativo. Você tem duas opções principais para converter e migrar seu schema para o Amazon Aurora:

- **AWS Schema Conversion Tool.** A AWS Schema Conversion Tool facilita as migrações de banco de dados heterogêneas convertendo automaticamente o schema do banco de dados de origem e a maior parte do código personalizado, incluindo exibições, procedimentos armazenados e funções, em um formato compatível com o banco de dados de destino.[11] Qualquer código que não possa ser convertido automaticamente será deixado em evidência para que possa ser convertido manualmente. Você pode usar esta ferramenta para converter bancos de dados de origem em execução no Oracle ou no Microsoft SQL Server para um banco de dados de destino do Amazon Aurora, MySQL ou PostgreSQL no Amazon RDS ou no

Amazon EC2. Usar a AWS Schema Conversion Tool para converter seu schema do Oracle, SQL Server ou PostgreSQL em um formato compatível com o Aurora é o método preferencial.

- **Migração de schema manual e ferramentas de terceiros.** Se o seu banco de dados de origem não for Oracle, SQL Server ou PostgreSQL, será possível migrar manualmente seu schema de banco de dados de origem para o Aurora ou usar ferramentas de terceiros para migrar o schema em um formato compatível com o MySQL 5.6. A migração de schema manual pode ser um processo razoavelmente complicado dependendo do tamanho e da complexidade de seu schema de origem. Na maioria dos casos, no entanto, a conversão de schema manual compensa o esforço por conta da economia de custos, dos benefícios de desempenho e de outras melhorias oferecidas pelo Amazon Aurora.

Migração de schema usando a AWS Schema Conversion Tool

A AWS Schema Conversion Tool oferece uma interface de usuário com base em projeto para converter automaticamente o schema do banco de dados de origem em um formato compatível com o Amazon Aurora. É altamente recomendável que você use a AWS Schema Conversion Tool para avaliar o esforço da migração do banco de dados e para fazer um piloto da migração antes da migração de produção real.

A descrição a seguir mostra as etapas de alto nível do uso da AWS Schema Conversion Tool. Para obter instruções detalhadas, consulte a seção Getting Started do *AWS Schema Conversion Tool User Guide*.[12]

1. Primeiro, instale a ferramenta. A AWS Schema Conversion Tool está disponível para o Microsoft Windows, Mac OS X, Ubuntu Linux e Fedora Linux.

   Instruções detalhadas para download e instalação podem ser encontradas na seção Installation and Update do guia do usuário.[13] É importante refletir sobre onde a AWS Schema Conversion Tool será instalada. A ferramenta precisa se conectar diretamente a ambos os bancos de dados de origem e de destino para converter e aplicar o schema. Certifique-se de que a área de trabalho onde você pretende instalar a AWS Schema Conversion Tool tem conectividade de rede com os bancos de dados de origem e de destino.

2. Instale os drivers JDBC. A AWS Schema Conversion Tool usa drivers JDBC para se conectar aos bancos de dados de origem e de destino. Para usar essa ferramenta, você deve fazer download desses drivers JDBC para o seu desktop local. As instruções para fazer download do driver podem ser encontradas em Required Database Drivers no *AWS Schema Conversion Tool User Guide*.[14] Além disso, consulte as instruções de configuração dos drivers JDBC para diferentes mecanismos de banco de dados no fórum da AWS para AWS Schema Conversion Tool.[15]
3. Crie um banco de dados de destino. Crie um banco de dados de destino do Amazon Aurora. Para obter instruções sobre a criação de um banco de dados do Amazon Aurora, consulte Creating an Amazon Aurora DB Cluster no *Amazon RDS User Guide*[16].
4. Abra a AWS Schema Conversion Tool e inicie o **New Project Wizard**.

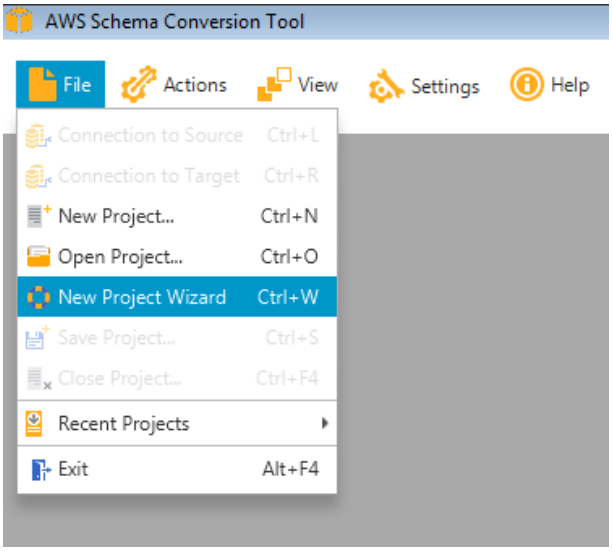

Figura 3: criar um novo projeto da AWS Schema Conversion Tool

5. Configure o banco de dados de origem e teste a conectividade entre a AWS Schema Conversion Tool e o banco de dados de origem. Para que isso funcione, seu banco de dados de origem deve ser acessível de sua área de trabalho, portanto, verifique se você possui uma rede apropriada e configurações de firewall aplicadas.

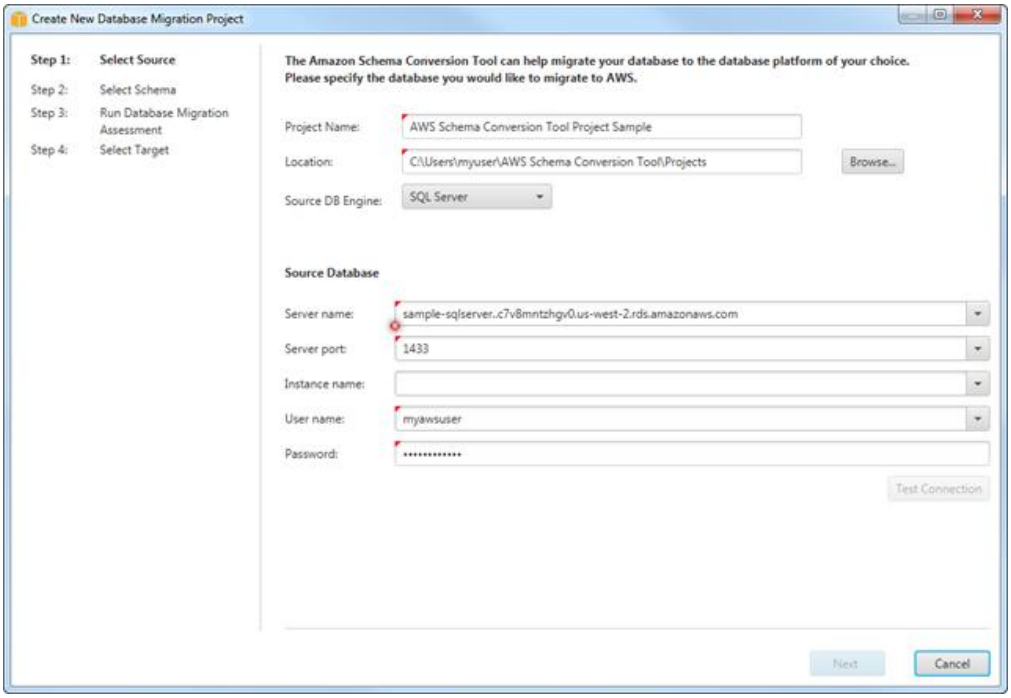

Figura 4: assistente Create New Database Migration Project

6. Na próxima tela, selecione o schema do banco de dados de origem que você deseja converter para o Amazon Aurora.

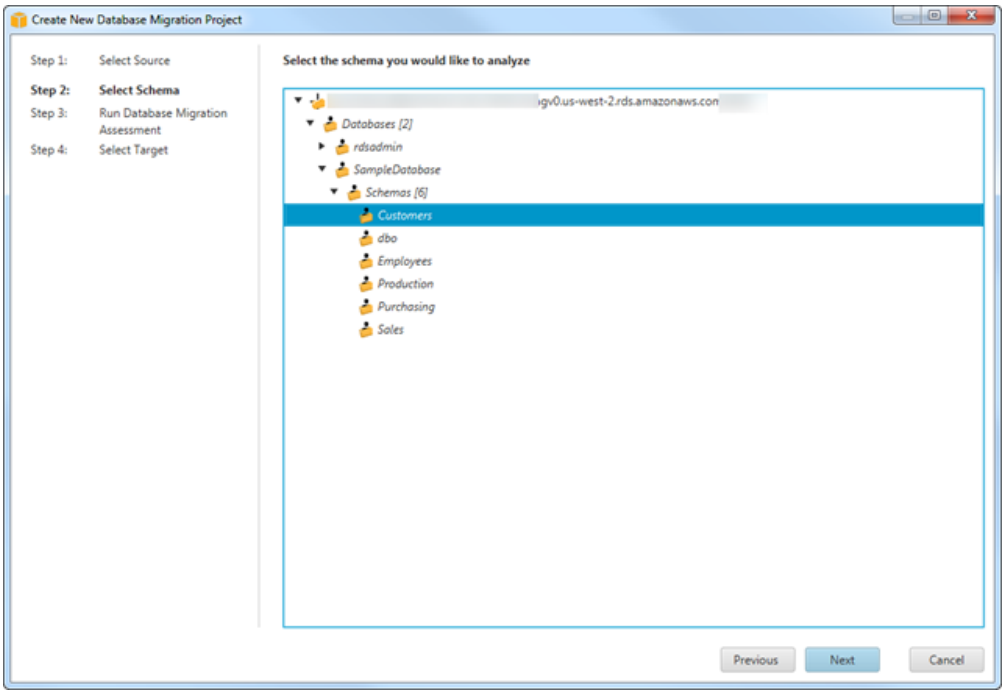

Figura 5: etapa Select Schema do assistente de migração

7. Execute o relatório de avaliação da migração do banco de dados. Esse relatório fornece informações importantes sobre a conversão do schema de seu banco de dados de origem para sua instância do Amazon Aurora. Ele resume todas as tarefas de conversão de schema e detalha os itens de ação para partes do schema que não podem ser automaticamente convertidas para o Aurora. O relatório também inclui estimativas do esforço que levará para gravar no banco de dados de destino o código equivalente que não pôde ser convertido automaticamente.

   Clique em **Next** para configurar o banco de dados de destino. Você pode visualizar esse relatório de migração novamente mais tarde.

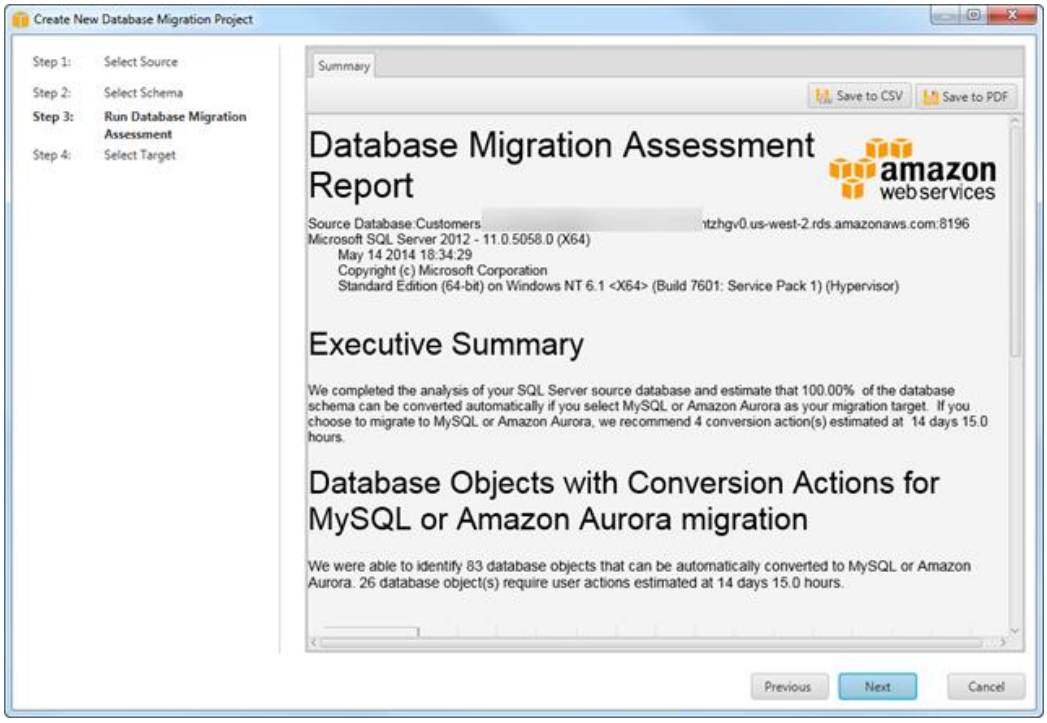

Figura 6: relatório de migração

8. Configure o banco de dados de destino do Amazon Aurora e teste a conectividade entre a AWS Schema Conversion Tool e o banco de dados de origem. Para que isso funcione, seu banco de dados de destino deve ser acessível de sua área de trabalho, portanto, verifique se você possui uma rede apropriada e configurações de firewall aplicadas. Clique em **Finish** para acessar a janela do projeto.
9. Quando você estiver na janela do projeto, significa que a conexão com os bancos de dados de origem e de destino já foi estabelecida e agora você está pronto para avaliar o relatório detalhado e migrar o schema.

10. No painel esquerdo que exibe o schema de seu banco de dados de origem, escolha um objeto de schema para criar um relatório de avaliação. Clique com o botão direito do mouse no objeto e selecione **Create Report**.

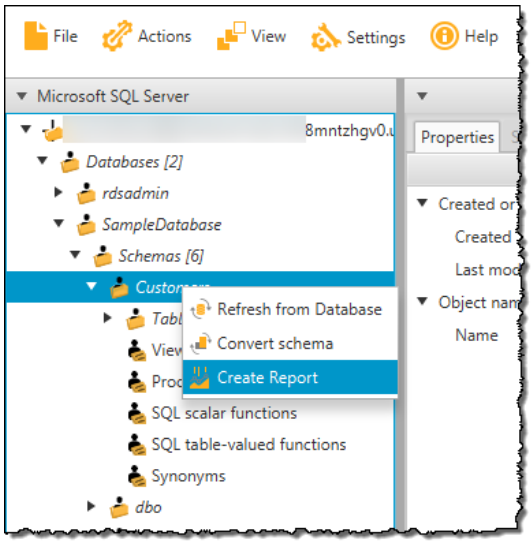

Figura 7: criar relatório de migração

A guia **Summary** exibe as informações de resumo do relatório de avaliação da migração do banco de dados. Ele mostra os itens que foram convertidos automaticamente e os itens que não puderam ser convertidos automaticamente.

Para itens de schema que não puderam ser convertidos automaticamente para o mecanismo de banco de dados de destino, o resumo inclui uma estimativa do esforço que seria necessário para criar um schema equivalente ao banco de dados de origem em sua instância de banco de dados de destino. O relatório categoriza a estimativa de tempo para converter esses itens de schema da seguinte forma:

- **Simples** – ações que podem ser concluídas em menos de uma hora.
- **Médio** – ações que são mais complexas e podem ser concluídas entre uma e quatro horas.
- **Significativo** – ações que são muito complexas e levarão mais de quatro horas para serem concluídas.

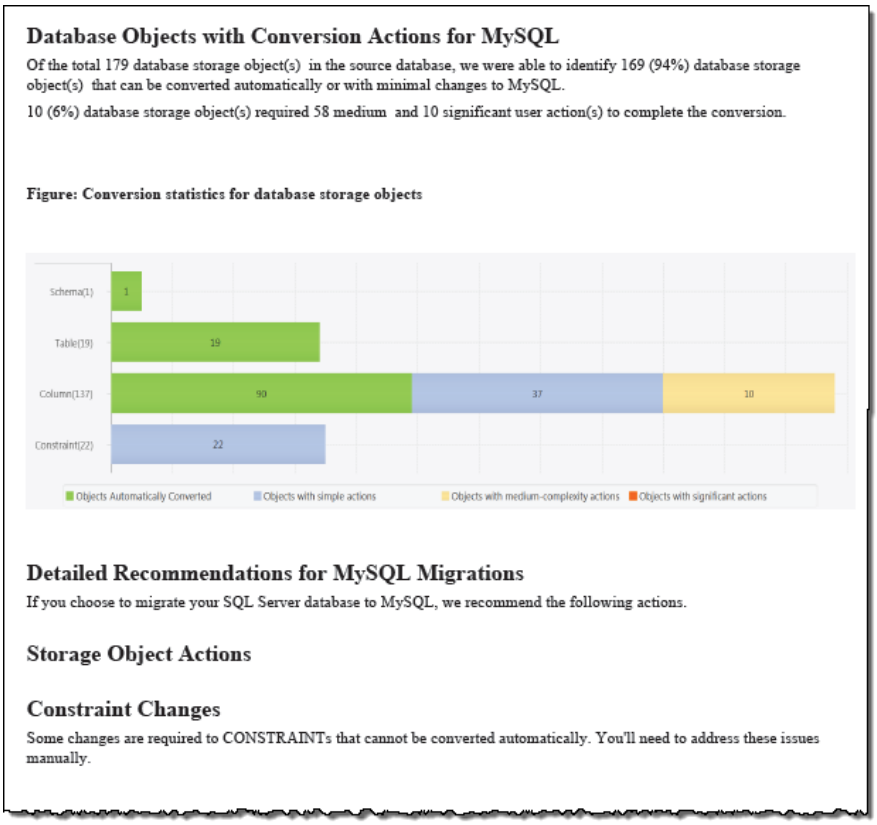

Figura 8: relatório de migração

Importante: se você estiver avaliando o esforço necessário para o seu projeto de migração de banco de dados, este relatório de avaliação é um importante artefato a se considerar. Estude o relatório de avaliação em detalhes para determinar quais alterações de código são necessárias no schema do banco de dados e o impacto que as alterações podem ter na funcionalidade e no design de seu aplicativo.

11. A próxima etapa é converter o schema. O schema convertido não é aplicado imediatamente ao banco de dados de destino. Em vez disso, ele é armazenado localmente até que você aplique explicitamente o schema convertido ao banco de dados de destino. Para converter o schema do banco de dados de origem, escolha um objeto de schema para converter no painel esquerdo de seu projeto. Clique com o botão direito do mouse no objeto e selecione **Convert schema**, como mostrado na ilustração a seguir.

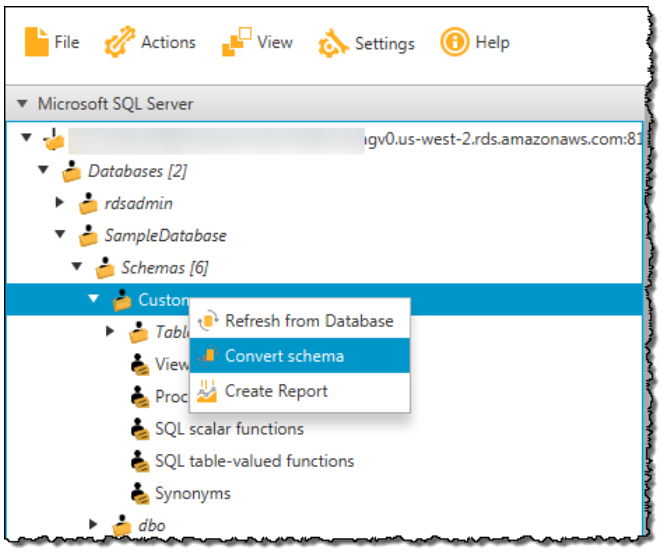

Figura 9: Convert schema

Essa ação adiciona o schema convertido ao painel direito da janela do projeto e mostra os objetos que foram convertidos automaticamente pela AWS Schema Conversion Tool.

12. Você pode responder aos itens de ação no relatório de avaliação de diferentes maneiras:

- Adicione o schema equivalente de forma manual. É possível gravar em sua instância do banco de dados de destino a parte do schema que pode ser automaticamente convertida selecionando **Apply to database** no painel direito do projeto. O schema gravado na instância do banco de dados de destino não incluirá os itens que não puderam ser convertidos automaticamente. Esses itens são listados no relatório de avaliação da migração do banco de dados.

  Depois de aplicar o schema à instância do banco de dados de destino, você pode criar manualmente o schema na instância do banco de dados de destino para os itens que não puderam ser convertidos automaticamente. Em alguns casos, talvez você não consiga criar um schema equivalente na instância do banco de dados de destino. Pode ser preciso reprojetar uma parte de seu aplicativo e do banco de dados para usar a funcionalidade que está disponível no mecanismo de banco de dados para sua instância do banco de dados de destino. Em outros casos, você pode simplesmente ignorar o schema que não pôde ser convertido automaticamente.

Atenção: se você criar o schema manualmente em sua instância do banco de dados de destino, não selecione **Apply to database** até que tenha salvo uma cópia de qualquer trabalho manual que você tenha feito. Se você aplicar o schema de seu projeto em sua instância do banco de dados de destino, o schema com o mesmo nome na instância do banco de dados de destino será substituído e você perderá todas as atualizações realizadas manualmente.

- Modifique o schema do banco de dados de origem e atualize o schema em seu projeto. Para alguns itens, talvez seja melhor modificar o schema em seu banco de dados de origem para o schema compatível com a arquitetura de seu aplicativo e que também poderá ser convertido automaticamente para o mecanismo de banco de dados de sua instância do banco de dados de destino. Após atualizar o schema no banco de dados de origem e verificar se as atualizações são compatíveis com seu aplicativo, selecione **Refresh from Database** no painel esquerdo de seu projeto para atualizar o schema do banco de dados de origem. Em seguida, você pode converter o schema atualizado e gerar novamente o relatório de avaliação da migração do banco de dados. O item de ação para o schema atualizado não aparecerá mais.

13. Quando você estiver pronto para aplicar o schema convertido à sua instância do Aurora, selecione o elemento do schema no painel direito do projeto. Clique com o botão direito do mouse no elemento do schema e selecione **Apply to database**, conforme mostrado na figura a seguir.

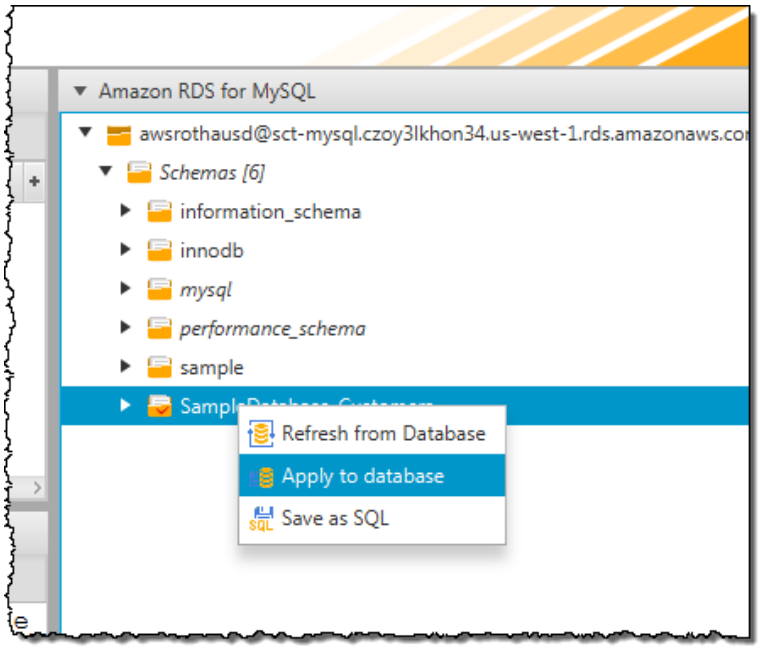

Figura 10: aplicar schema ao banco de dados

Observação: Na primeira vez que você aplicar o schema convertido à sua instância do banco de dados de destino, a AWS Schema Conversion Tool adicionará um schema adicional (`AWS_ORACLE_EXT` ou `AWS_SQLSERVER_EXT`) em sua instância do banco de dados de destino. Esse schema implementa funções do sistema do banco de dados de origem que são necessárias ao gravar o schema convertido em sua instância do banco de dados de destino. Não modifique esse schema, pois você pode encontrar resultados inesperados no schema convertido que foi gravado em sua instância do banco de dados de destino. Quando o schema for totalmente migrado para sua instância do banco de dados de destino e você não precisar mais da AWS Schema Conversion Tool, poderá excluir o schema `AWS_ORACLE_EXT` ou `AWS_SQLSERVER_EXT`.

A AWS Schema Conversion Tool é um complemento fácil de usar para seu kit de ferramentas de migração. Para saber outras melhores práticas relacionadas à AWS Schema Conversion Tool, consulte o tópico Best Practices no *AWS Schema Conversion Tool User Guide*.[17]

Migrar dados

Depois que o schema do banco de dados tiver sido copiado do banco de dados de origem para o banco de dados de destino do Aurora, a próxima etapa será migrar os dados reais da origem para o destino. Embora a migração de dados possa ser realizada usando ferramentas diferentes, recomendamos que você transfira os dados usando o AWS Database Migration Service (AWS DMS), já que ele oferece a simplicidade e os recursos necessários para a tarefa a ser realizada.

Introdução e abordagem geral do AWS DMS

O AWS Database Migration Service (AWS DMS) facilita a migração de bancos de dados de produção para a AWS com um tempo de inatividade mínimo. Você pode manter seus aplicativos em execução durante a migração do banco de dados. Além disso, o AWS Database Migration Service garante que as mudanças de dados no banco de dados de origem que ocorrem durante e após o processo de migração sejam continuamente replicadas no destino. As tarefas de migração podem ser configuradas em questão de minutos no Console de Gerenciamento da AWS. O AWS Database Migration Service pode migrar seus dados de e para plataformas de banco de dados amplamente usadas, como Oracle, SQL Server, MySQL, PostgreSQL, Amazon Aurora, MariaDB e Amazon Redshift.

O serviço oferece suporte a migrações homogêneas, como de Oracle para Oracle, e para migrações heterogêneas entre diferentes plataformas de banco de dados, como de Oracle para Amazon Aurora ou de SQL Server para MySQL. Você pode realizar migrações ocasionais ou pode manter a replicação contínua entre bancos de dados sem que um cliente precise instalar ou configurar um software complexo.

O AWS DMS funciona com bancos de dados que estão no local, em execução no Amazon EC2 ou em execução no Amazon RDS. No entanto, o AWS DMS não funciona em situações em que o banco de dados de origem e o banco de dados de destino estão no local; um dos endpoints deve estar na AWS.

O AWS DMS é compatível com versões específicas do Oracle, SQL Server, Amazon Aurora, MySQL e PostgreSQL. Para saber as versões atualmente compatíveis, consulte o AWS Database Migration Service User Guide.[18] No entanto, este whitepaper está voltado apenas para o Amazon Aurora como um destino de migração.

Métodos de migração

O AWS DMS oferece três métodos para migrar dados:

Migrar dados existentes. Este método cria as tabelas no banco de dados de destino, define automaticamente os metadados necessários no destino e preenche as tabelas com dados do banco de dados de origem (também conhecidos como "carregamento total"). Os dados das tabelas são carregados em paralelo para melhorar a eficiência. As tabelas serão criadas somente se as migrações homogêneas e os índices secundários não forem criados automaticamente pelo AWS DMS. Continue a leitura para obter mais detalhes.

Migrar dados existentes e replicar as alterações contínuas. Este método faz um carregamento total, conforme descrito acima, além de capturar todas as alterações contínuas feitas no banco de dados de origem durante o carregamento total e armazená-las na instância de replicação. Assim que a o carregamento total é concluído, as alterações armazenadas são aplicadas no banco de dados de destino até que ele seja atualizado com o banco de dados de origem. Além disso, todas as alterações contínuas feitas no banco de dados de origem continuam a ser replicadas no banco de dados de destino para mantê-los sincronizados. Este método de migração é muito útil quando você deseja executar uma migração de banco de dados com um tempo de inatividade muito curto.

Replicar somente alterações de dados. Este método simplesmente lê as alterações no arquivo de log de recuperação do banco de dados de origem e as aplica no banco de dados de destino de forma contínua. Se o banco de dados de destino não estiver disponível, essas alterações serão armazenadas em buffer na instância de replicação até que o destino se torne disponível.

Quando o AWS DMS está executando uma migração de carregamento total, o processamento coloca uma carga nas tabelas no banco de dados de origem, o que pode afetar o desempenho de aplicativos que estão tentando acessar esse banco de dados ao mesmo tempo. Se isso for um problema e você não puder encerrar seus aplicativos durante a migração, pense na possibilidade de adotar as seguintes abordagens:

- Execute a migração num momento em que carga do aplicativo no banco de dados esteja em seu ponto mais baixo.
- Crie uma réplica de leitura de seu banco de dados de origem e, em seguida, execute a migração do AWS DMS a partir da réplica de leitura.

Procedimento de migração

A descrição geral para usar o AWS DMS é fornecida a seguir:

1. Crie um banco de dados de destino.
2. Copie o schema.
3. Crie uma instância de replicação do AWS DMS.
4. Defina os endpoints dos bancos de dados de origem e de destino.
5. Crie e execute uma tarefa de migração.

Criar banco de dados de destino

Crie seu cluster de banco de dados de destino do Amazon Aurora usando o procedimento descrito em Creating an Amazon Aurora DB Cluster no *Amazon RDS User Guide*.[19] Você deve criar o banco de dados de destino na região e com um tipo de instância que atenda às suas necessidades de negócios. Além disso, para melhorar o desempenho da migração, verifique se o banco de dados de destino não tem a Implantação Multi-AZ habilitada; você pode habilitá-la depois que o carregamento for concluído.

Copiar schema

Além disso, você deve criar o schema neste banco de dados de destino.
O AWS DMS oferece suporte à migração de schema básica, incluindo a criação de tabelas e chaves primárias. No entanto, o AWS DMS não cria automaticamente índices secundários, chaves estrangeiras, procedimentos armazenados, contas de usuários etc. no banco de dados de destino. Para obter todos os detalhes da migração de schema, consulte a seção Migrar o schema do banco de dados.

Criar uma instância de replicação do AWS DMS

Para usar o AWS DMS, você deve criar uma instância de replicação do AWS DMS que seja executada em sua VPC. Essa instância lê os dados do banco de dados de origem, executa os mapeamentos da tabela especificada e grava os dados no banco de dados de destino. Em geral, o uso de uma instância de replicação maior acelera a migração do banco de dados (embora a migração também possa ser limitada por outros fatores, como a capacidade dos bancos de dados de origem e de destino, a latência da conexão etc.). Além disso, a instância de replicação pode ser interrompida assim que a migração do banco de dados for concluída.

Figura 11: AWS Database Migration Service

O AWS DMS atualmente oferece suporte às classes de instância T2 e C4 para instâncias de replicação. As classes de instância T2 são instâncias padrão de baixo custo projetadas para fornecer um nível básico de desempenho da CPU com capacidade de intermitência acima da linha de base. Elas são adequadas para desenvolver, configurar e testar o processo de migração do banco de dados, bem como para tarefas de migração de dados periódicas que podem se beneficiar do recurso de intermitência da CPU. As classes de instância C4 foram projetadas para proporcionar o mais alto nível de desempenho de processador e atingir uma performance de pacotes por segundo (PPS – Packet Per Second) significativamente mais alta, além de menor variação e latência de rede. Você deve usar as classes de instância C4 se estiver migrando bancos de dados grandes e quiser minimizar o tempo de migração.

Normalmente, o carregamento total não requer uma quantidade significativa de instance storage em sua instância de replicação do AWS DMS. No entanto, se você estiver fazendo a replicação junto com o carregamento total, as alterações no banco de dados de origem serão armazenadas na instância de replicação do AWS DMS enquanto o carregamento total estiver sendo realizado. Portanto, se você estiver migrando um banco de dados de origem muito grande que também está recebendo muitas atualizações durante a migração, então é possível que você use uma quantidade significativa de instance storage. A família de instâncias C4 é fornecida com 100 GB de instance storage, e a família de instâncias T2 vem com 50 GB. Normalmente, essas quantidades de armazenamento devem ser mais do que suficiente para a maioria dos cenários de migração.

Além disso, em alguns casos extremos em que bancos de dados muito grandes e com taxas de transação muito elevadas estão sendo migrados com a replicação habilitada, é possível que a replicação do AWS DMS não consiga acompanhar o ritmo. Se você se deparar com essa situação, talvez precise interromper as alterações no banco de dados de origem durante alguns minutos para que a

replicação se recupere antes que você redirecione seu aplicativo para o banco de dados de destino do Aurora.

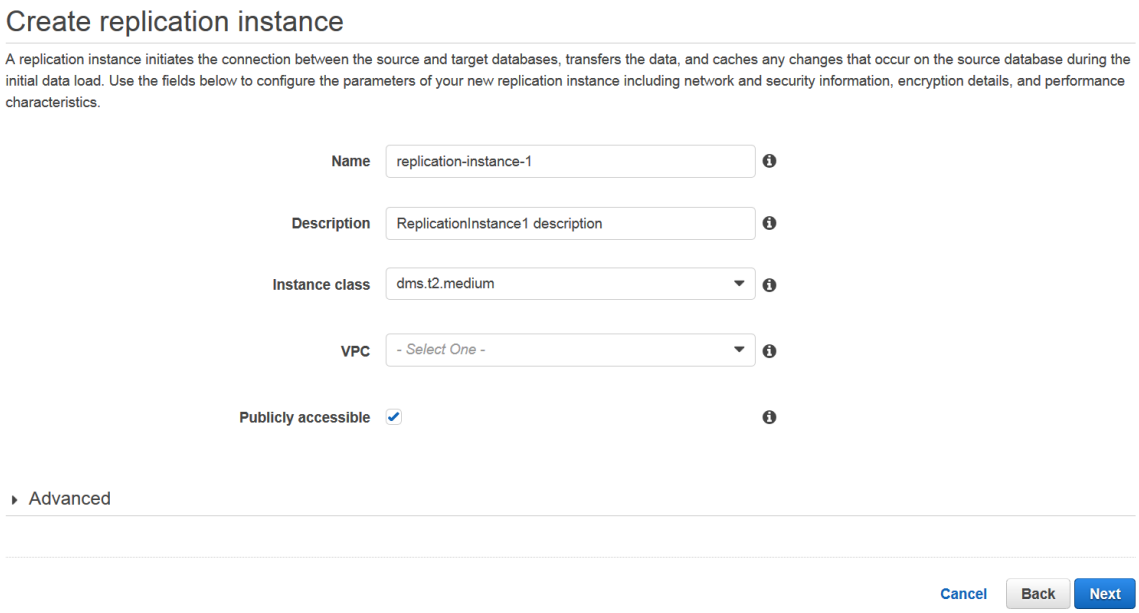

Figura 12: criar uma página de instância de replicação no console do AWS DMS

Definir endpoints de origem e de destino do banco de dados

Um endpoint de banco de dados é usado pela instância de replicação para se conectar a um banco de dados. Para executar uma migração de banco de dados, você deve criar *tanto* um endpoint de banco de dados de origem *como* um endpoint de banco de dados de destino. Os endpoints de banco de dados especificados podem estar no local, em execução no Amazon EC2 ou em execução no Amazon RDS, desde que a origem e o destino não estejam ambos no local.

É altamente recomendável que você teste a conexão do endpoint do banco de dados depois de defini-lo. A mesma página usada para criar um endpoint de banco de dados também pode ser usada para testá-lo, conforme explicado posteriormente neste documento.

Observação: se você tiver restrições de chave estrangeira em seu schema de origem, ao criar o endpoint de destino, será preciso informar o seguinte para **Extra connection attributes** na seção **Advanced**:

```
initstmt=SET FOREIGN_KEY_CHECKS=0
```

Isso desabilita as verificações de chave estrangeira enquanto as tabelas de destino estão sendo carregadas. Por sua vez, isso acaba impedindo que o carregamento seja interrompido por verificações de chave estrangeira com falha em tabelas parcialmente carregadas.

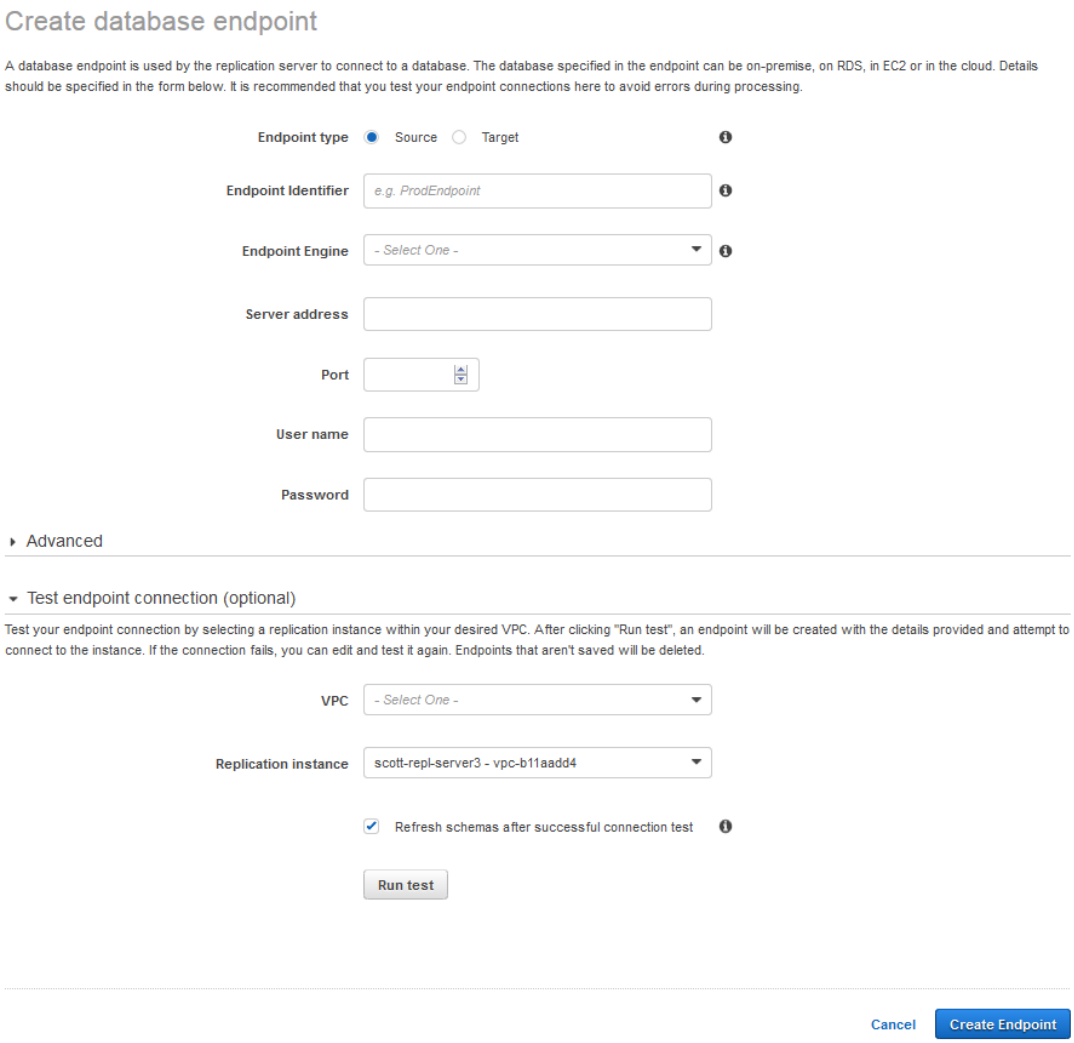

Figura 13: página Create database endpoint no console do AWS DMS

Criar e executar uma tarefa de migração

Agora que você criou e testou os endpoints dos bancos de dados de origem e de destino, pode criar uma tarefa para fazer a migração de dados. Quando você cria uma tarefa, especifica a instância de replicação que criou, o tipo de método de migração de banco de dados (discutido anteriormente), o endpoint do banco de dados de origem e o endpoint do banco de dados de destino para seu cluster do banco de dados do Amazon Aurora.

Além disso, em **Task Settings**, se você já tiver criado o schema completo no banco de dados de destino, será preciso alterar **Target table preparation mode** para **Do nothing**, em vez de usar o valor padrão **Drop tables on target**. O último pode fazer com que você perca aspectos de sua definição de schema, como restrições de chave estrangeira, quando ele transfere e recria tabelas.

Ao criar uma tarefa, você pode criar mapeamentos de tabela que especificam o schema de origem junto com tabelas correspondentes a serem migradas para o endpoint de destino. O método de mapeamento padrão migra todas as tabelas de origem para tabelas de destino com o mesmo nome, se existirem. Caso contrário, ele cria a(s) tabela(s) de origem no destino (dependendo de suas configurações de tarefas). Além disso, você pode criar mapeamentos personalizados (usando um arquivo JSON) se quiser migrar apenas determinadas tabelas ou se quiser ter mais controle sobre o processo de mapeamento de tabela e campo. Você também pode optar por migrar apenas um schema ou todos os esquemas de seu endpoint de origem.

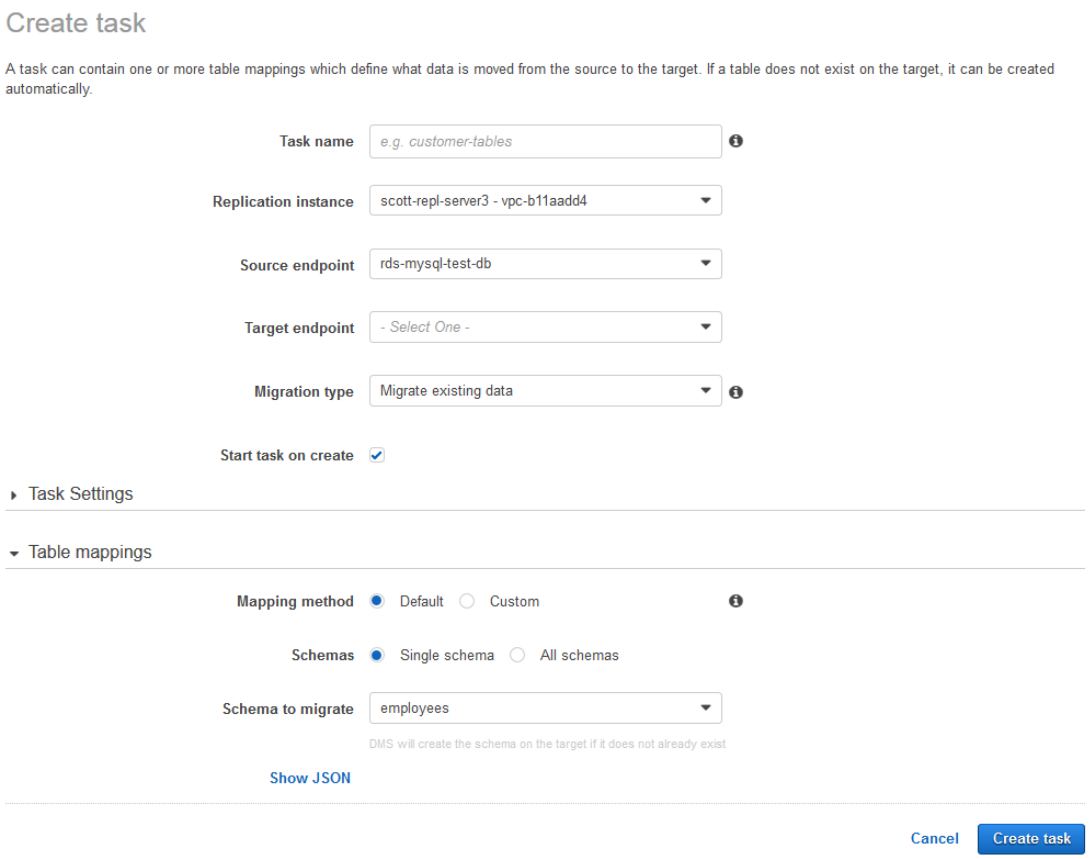

Figura 14: página Create task no console do AWS DMS

Você pode usar o Console de Gerenciamento da AWS para monitorar o progresso de suas tarefas do AWS Database Migration Service (AWS DMS). Você também pode monitorar os recursos e a conectividade de rede usada. O console do AWS DMS mostra as estatísticas básicas para cada tarefa, incluindo o status da tarefa, a porcentagem de conclusão, o tempo decorrido e as estatísticas da tabela, conforme mostrado na imagem a seguir.

Além disso, você pode selecionar uma tarefa e exibir métricas de desempenho para ela, incluindo a taxa de transferência, os registros por segundo migrados, o uso de memória e espaço de disco e a latência.

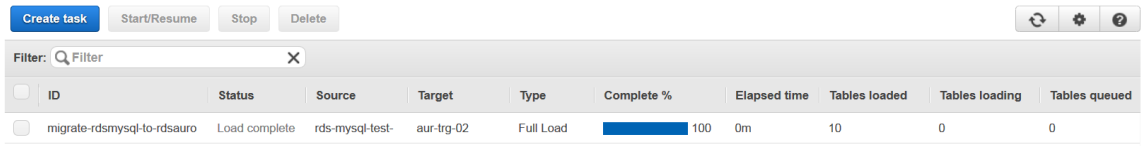

Figura 15: status da tarefa no console do AWS DMS

Teste e cutover

Depois que o schema e os dados tiverem sido migrados com sucesso do banco de dados de origem para o Amazon Aurora, você estará pronto para realizar o teste completo do processo de migração. A abordagem de teste deve ser refinada após cada migração de teste, e o plano de migração final deve incluir um plano de teste que garante testes adequados do banco de dados migrado.

Testes de migração

Categoria do teste	Finalidade
Testes de aceitação básicos	Esses testes pre-cutover devem ser executados automaticamente após a conclusão do processo de migração de dados. O objetivo principal deles é verificar se a migração dos dados foi bem-sucedida. Veja a seguir alguns resultados comuns desses testes: • Número total de itens processados • Número total de itens importados • Número total de itens ignorados • Número total de avisos • Número total de erros

Categoria do teste	Finalidade
	Se algum desses totais relatados pelos testes desviarem dos valores esperados, isso significa que a migração não foi bem-sucedida e que os problemas devem ser resolvidos antes que você passe para a próxima etapa do processo ou para a próxima rodada de testes.
Testes funcionais	Esses testes post-cutover exercitam a funcionalidade do(s) aplicativo(s) usando o Aurora para armazenamento de dados. Eles incluem uma combinação de testes automatizados e manuais. O objetivo principal dos testes funcionais é identificar problemas no aplicativo causados pela migração dos dados para o Aurora.
Testes não funcionais	Esses testes post-cutover avaliam as características não funcionais do aplicativo, como o desempenho em níveis variáveis de carregamento.
Testes de aceitação do usuário	Esses testes post-cutover devem ser executados pelos usuários finais do aplicativo assim que a migração e o cutover final dos dados forem concluídos. O propósito desses testes é que os usuários finais decidam se o aplicativo é aproveitável o suficiente para cumprir com sua função principal na organização.

Cutover

Depois de concluir a migração final e os testes, é hora de direcionar seu aplicativo para o banco de dados do Amazon Aurora. Essa fase da migração é conhecida como *cutover*. Se as fases de planejamento e teste tiverem sido devidamente executadas, o cutover não deve apresentar problemas inesperados.

Ações pre-cutover

- Escolha uma janela de cutover: identifique um período de tempo em que você pode realizar cutover para o novo banco de dados com o mínimo de interrupção para os negócios. Normalmente, você deve escolher um período de baixa atividade para o banco de dados (normalmente, noites e/ou fins de semana).
- Verifique quais alterações estão funcionando: se uma abordagem de migração com tempo de inatividade quase nulo tiver sido usada para replicar as alterações no banco de dados de origem para o banco de dados de destino, verifique se todas as alterações estão sendo detectadas e se o banco de dados de destino não está significativamente defasado em relação ao banco de dados de origem.
- Prepare scripts para fazer as alterações na configuração do aplicativo: para realizar o cutover, é preciso modificar os detalhes da conexão do banco de dados nos arquivos de configuração de seu aplicativo. Aplicativos

grandes e complexos podem exigir atualizações nos detalhes de conexão em vários locais. Verifique se você tem os scripts necessários prontos para atualizar a configuração de conexão de forma rápida e confiável.

- Pare o aplicativo: pare os processos de aplicativo no banco de dados de origem e coloque-o em modo somente leitura para que nenhuma outra gravação seja feita no banco de dados de origem. Se as alterações no banco de dados de origem não forem totalmente replicadas no banco de dados de destino, aguarde um momento enquanto essas alterações são totalmente propagadas para o banco de dados de destino.
- Execute testes de pre-cutover: execute testes de pre-cutover automatizados para garantir que a migração de dados tenha sido bem-sucedida.

Cutover

- Execute o cutover: se as verificações pre-cutover tiverem sido concluídas com êxito, agora você pode direcionar seu aplicativo para o Amazon Aurora. Execute scripts criados na fase pre-cutover para alterar a configuração do aplicativo para direcioná-lo para o novo banco de dados do Aurora.
- Inicie seu aplicativo: nesse ponto, você pode iniciar seu aplicativo. Se você tiver como impedir que os usuários acessem o aplicativo enquanto ele é executado, faça isso até que tenha realizado as verificações post-cutover.

Verificações post-cutover

- Execute os testes post-cutover: execute casos de teste manuais ou automatizados predefinidos para garantir que seu aplicativo funcione conforme o esperado com o novo banco de dados. Uma boa estratégia é começar a testar a funcionalidade somente leitura do banco de dados antes de executar os testes que fazem gravações nele.
- Habilite o acesso dos usuários e monitore atentamente: se os casos de teste tiverem sido executados com êxito, você pode dar acesso aos usuários para que o aplicativo conclua o processo de migração. Nesse momento, tanto o aplicativo como o banco de dados devem ser monitorados atentamente.

Conclusão

O Amazon Aurora é um banco de dados de alto desempenho, altamente disponível e de nível empresarial criado para a nuvem. O uso do Amazon Aurora pode resultar em um melhor desempenho e uma maior disponibilidade do que

outros bancos de dados de código aberto, além de custos mais baixos do que a maioria dos bancos de dados de nível comercial. Este documento propõe estratégias para identificar o melhor método de migração de bancos de dados para o Amazon Aurora e detalha os procedimentos para planejar e executar essas migrações. Mais especificamente, as ferramentas recomendadas para os cenários de migração heterogênea são o AWS Database Migration Service (AWS DMS) e a AWS Schema Conversion Tool. Essas ferramentas eficientes podem reduzir significativamente os custos e a complexidade das migrações de bancos de dados.

Colaboradores

As seguintes organizações e pessoas contribuíram para este documento:

- Puneet Agarwal, arquiteto de soluções, Amazon Web Services
- Scott Williams, arquiteto de soluções, Amazon Web Services

Outras fontes de leitura

Para obter mais ajuda, consulte estas fontes:

- Detalhes do produto Amazon Aurora
- Perguntas frequentes sobre o Amazon Aurora
- Amazon Database Migration Service
- Perguntas frequentes sobre o Amazon Database Migration Service

Observações

[1] https://aws.amazon.com/rds/aurora/

[2] http://aws.amazon.com/rds/aurora/pricing/

[3] https://d0.awsstatic.com/product-marketing/Aurora/Aurora_Export_Import_Best_Practices_v1-3.pdf

[4] http://docs.aws.amazon.com/pt_br/AmazonRDS/latest/UserGuide/Aurora.Replication.html#Aurora.Overview.Replication.MySQLReplication

[5] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Migrate.html#USER_ImportAurora.PreImport

[6] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html

[7] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html

[8] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Migrate.html#USER_ImportAuroraCluster.Console

[9] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Connect.html

[10] https://dev.mysql.com/doc/refman/5.6/en/mysqldump.html

[11] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/Welcome.html

[12] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.GettingStarted.html

[13] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.Installing.html

[14] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.Installing.html#CHAP_SchemaConversionTool.Installing.JDBCDrivers

[15] https://forums.aws.amazon.com/forum.jspa?forumID=208

[16] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.CreateInstance.html

[17] http://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/CHAP_SchemaConversionTool.BestPractices.html

[18] http://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.html

[19] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.CreateInstance.html