Como migrar MongoDB para Autonomous Database em 3 passos simples e sem refactoring

Thamires Samira Ferreira

21.03.2022 - 10h00

https://github.com/oracledatabr/datamodernization

Índice

Introdução

Um banco de dados convergente é um banco de dados com suporte nativo a todos os tipos de dados modernos e os mais recentes paradigmas de desenvolvimento integrados num único produto.

Bancos de dados convergentes suportam dados espaciais para reconhecimento de localização, JSON para armazenamento de documentos, IoT para integração de dispositivos, tecnologias in-memory para análises em tempo real e, claro, os dados relacionais tradicionais.

Ao fornecer suporte a todos esses tipos dados, o Banco de Dados Convergente pode executar todas as espécies de cargas de trabalho, de IoT a Blockchain até a Analytics e a Machine Learning. Ela também consegue operar com qualquer paradigma de desenvolvimento, incluindo Microservices, Events, REST, SaaS e CI/CD, para citar alguns.

MongoDB

Lançado em 2009, o MongoDB é um dos bancos de dados não-relacionais mais comuns do mercado. Ele tem como linguagem C++ e utiliza o Java Script para facilitar os recursos de pesquisas.

Ele também é open source e orientado por documentos (document database) em formato JSON. Por ser não relacional, não demanda a utilização de tabelas com colunas e linhas para fazer a armazenagem de dados. O MongoDB funciona em Windows, Linux e OSX.

Seus dados são armazenados dentro de documentos semelhantes a JSON, porém utilizando uma versão binário do JSON chamada BSON, que retém os dados usando pares de chave/valo.

Por exemplo:

Elementos principais no MongoDB:

- **Database**: Este é o elemento de nível superior. Sendo um recipiente físico de uma estrutura chamado de coleção (collections). Cada banco de dados tem o seu próprio conjunto de arquivos no seus sistemas de arquivos, porem o MongoDB tem um único servidor normalmente tem vários banco de dados.
- **Collection**: Este é um conjunto formado por documentos do MongoDB. Somente pode haver uma coleção com esse nome no banco de dados.
- **Documents**: Esta é a unidade básica de dado no MongoDB. Basicamente, ela é composta por conjuntos pares de chave/valor. Documentos são dinâmicos e fazem parte de uma mesma coleção sem precisar ter o mesmo conjunto de campos.

Uma das grandes diferenças entre candos de dados convergentes como Oracle Autonomous Database e bancos de dados não relacionais como o MongoDB é que bancos de dados não relacionais em geral abrem mão de propriedades ACID para funcinar.

O que é ACID ?

ACID é um conceito que se refere às quatro propriedades de transação de um sistema de banco de dados: Atomicidade, Consistência, Isolamento e Durabilidade.

- **Atomicidade:** Em uma transação envolvendo duas ou mais partes de informações discretas, ou a transação será executada totalmente ou não será executada, garantindo assim que as transações sejam atômicas.
- **Consistência:** A transação cria um novo estado válido dos dados ou em caso de falha retorna todos os dados ao seu estado antes que a transação foi iniciada.
- **Isolamento:** Uma transação em andamento mas ainda não validada deve permanecer isolada de qualquer outra operação, ou seja, garantimos que a transação não será interferida por nenhuma outra transação concorrente.
- **Durabilidade:** Dados validados são registados pelo sistema de tal forma que mesmo no caso de uma falha e/ou reinício do sistema, os dados estão disponíveis em seu estado correto.

As propriedades ACID das transações permitem que você escreva aplicações sem considerar o ambiente complexo em que o aplicativo é executado.

Com transações ACID você pode se concentrar na lógica da aplicação e não na detecção de falhas, recuperação e sincronização do acesso aos dados compartilhados.

O Oracle Autonomous JSON Database é um serviço de banco de dados de documentos em nuvem que simplifica o desenvolvimento de aplicativos centrados em JSON. Ele apresenta APIs de documentos simples, dimensionamento sem servidor, transações ACID de alto desempenho, segurança abrangente e baixo preço de pagamento por uso.

Automatiza o provisionamento, configuração, ajuste, dimensionamento, patching, criptografia e reparo de bancos de dados, eliminando o gerenciamento de banco de dados e oferecendo disponibilidade de 99,995%.

Para saber mais acesse e aqui um comparative completo entre MongoDB e Autonomous Database: https://www.oracle.com/autonomous-database/autonomous-json-database/oracle-json-vs-mongodb-atlas/

O laboratório de hoje têm como intuito ensinar como habilitar umpa API para o seu Autonomous Database Always Free ou não que permite o acesso e manipulação de databases MongoDB com SQL diretamente no banco Oracle.

1. Considerações iniciais e pré-requisitos

Recursos usados:

OCI (all free tier)

- Conta OCI free tier
- Armazenamento Oracle Object Storage
- Oracle Autonomous Database – ATP
- Oracle Database API for MongoDB

Tópicos não cobertos:

- Configuração MongoDB Compass
- Instalação e data load em Mongo DB
- Conta gratuita no Oracle Cloud

Provisionar um Autonomous Database

1. Abra o menu de navegação. Em **Oracle Database**, clique em **Autonomous Data Warehouse**, **Autonomous Transaction Processing** ou **Autonomous JSON Database**. Para provisionar o APEX Application Development, você pode clicar nos tipos de carga de trabalho mencionados anteriormente ou navegar para **Serviços do Desenvolvedor** e, em seguida, clicar em **APEX Application Development**.

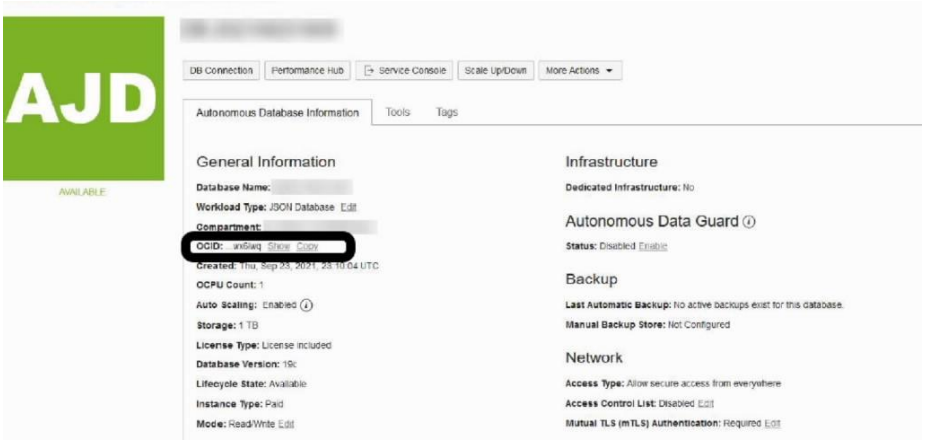
2. Forneça as seguintes informações do Autonomous Database:
   - **Compartimento:** Selecione o compartimento do Autonomous Database.
   - **Nome para exibição:** Uma descrição amigável do usuário ou outras informações que ajudam a identificar facilmente o recurso. O nome para exibição não precisa ser exclusivo. Evite inserir informações confidenciais.
   - **Nome do Banco de Dados:** O nome do banco de dados deve consistir apenas em letras e números, começando por uma letra. O tamanho máximo é de 14 caracteres.
3. Escolha **Autonomous Transaction Processing**
4. Escolha o tipo de implantação da **Infraestrutura Compartilhada**.
5. Configure o banco de dados:
   - **Always Free:** Mova esse seletor para a direita para que o workflow de provisionamento mostre apenas as opções de configuração Always Free. Observe que os campos de configuração **Contagem de núcleos de CPU** e **Armazenamento** estão desativados ao provisionar um Autonomous Database Always Free. Seu banco de dados terá 1 OCPU, 8 GB de memória e 20 GB de armazenamento.
   - **Escolher versão do banco de dados:** Selecione uma versão do banco de dados entre as versões disponíveis.
6. **Criar credenciais do administrador**: Defina a senha do usuário ADMIN do Autonomous Database digitando uma senha que atenda aos critérios a seguir. Essa senha é usada ao acessar a console de serviço do Autonomous Database e ao usar uma ferramenta cliente SQL.

   Critérios de senha:

   - Contém de 12 a 30 caracteres e inclui pelo menos uma letra maiúscula, uma letra minúscula e um caractere numérico.
   - Não contém a string "admin", independentemente de maiúsculas e minúsculas
   - Não é uma das quatro últimas senhas usadas para o usuário ADMIN
   - Não contém aspas duplas (")
   - Não pode ser a mesma senha definida há menos de 24 horas
7. Escolha o tipo de acesso à rede.
   - **Permitir acesso seguro de qualquer lugar:** Por padrão, conexões seguras são permitidas de todos os lugares. Com essa opção, a autenticação TLS (mTLS) mútua é necessária para se conectar ao banco de dados.

Habilitar a API

Acesse o Autonomous Transaction Processing:



Selecione Service Console

Siga para "**Database Action**" e execute:

```
SQL> CREATE TESTUSER IDENTIFIED BY <sua senha>;
SQL> GRANT SODA_APP to TESTUSER;
```

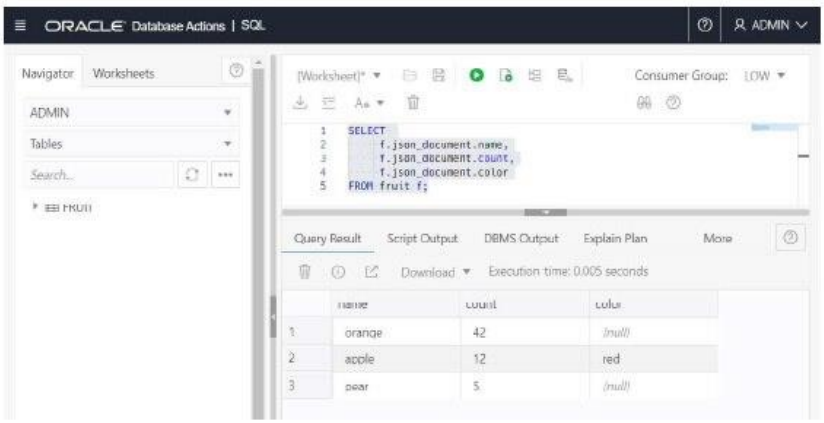
Acesse o MongoDB e utiize o commando abaixo para conectar o Autonomous ao seu MongoDB

```
$ mongosh --tls --tlsAllowInvalidCertificates 'mongodb://
TESTUSER:<PASSWORD>@<database URL>.
<OCI region>.oraclecloudapps.com:27016/admin?
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false'
Current Mongosh Log ID: 614c9e2a01e3575c8c0b2ec7
Connecting to:          mongodb://<credentials>@<database
URL>.<OCIregion>.oraclecloudapps.com:27016/admin?
authMechanism=PLAIN&authSource=$external&tls=true&loadBalanced=false
Using MongoDB:                  3.6.2
Using Mongosh:                  1.0.7
For mongosh info see: https://docs.mongodb.com/mongodb-shell/admin
> show dbs
testuser    0 B
>
```

Crie uma coleção:

```
testuser> db.createCollection( 'fruit' )
{ ok: 1 }
testuser> show collections
fruit
testuser> db.fruit.insertOne( {name:"orange", count:42} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca31fdab254f63e4c6b47")
}
testuser> db.fruit.insertOne( {name:"apple", count:12, color:
"red"} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca340dab254f63e4c6b48")
}
testuser> db.fruit.insertOne( {name:"pear", count:5} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca351dab254f63e4c6b49")
}
```

Volte para database actions e faça a query na coleção que acabou de criar:



A partir desse momento você poderá utilizar SQL para fazer querys e ter todos os beneficios de um banco de dados convergente e Oracle Autonomous Database.

Migrar dados de aplicativos do MongoDB para o banco de dados Oracle

Existem algumas maneiras de exportar seus dados JSON do MongoDB e depois importá-los para Oracle Database e Oracle Autonomous Database .

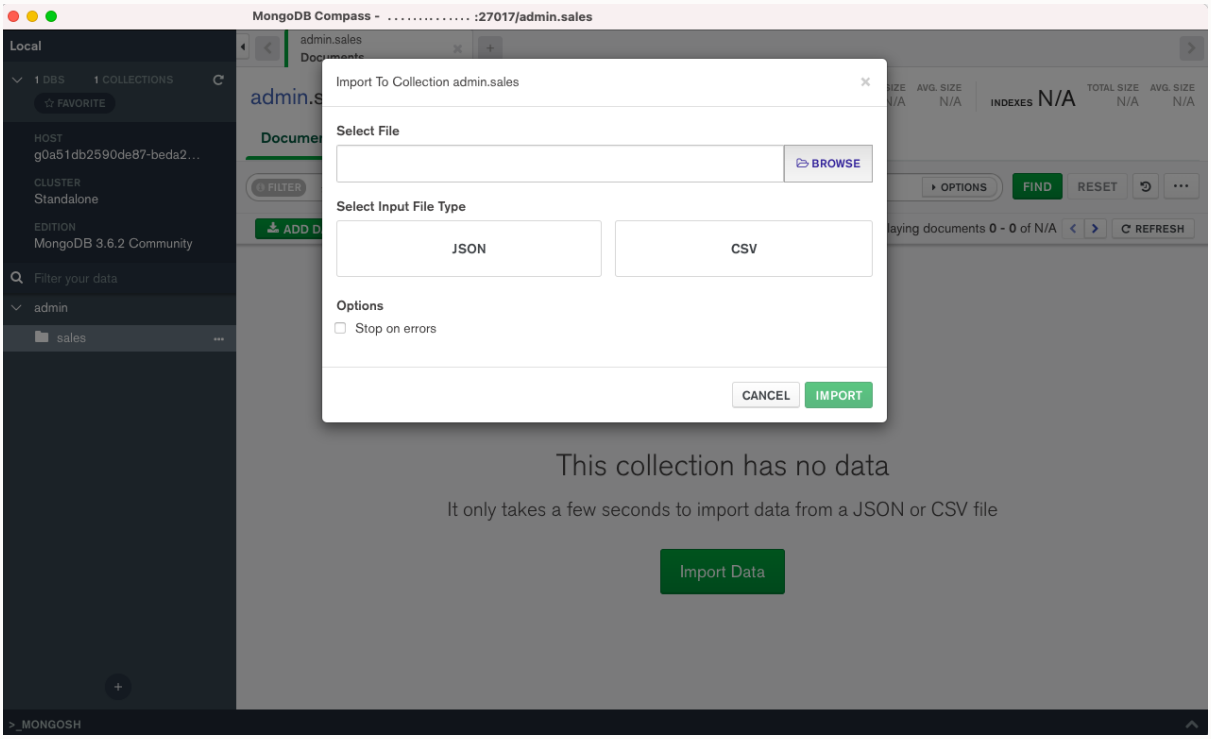
Você pode migrar seus dados de aplicativo de qualquer uma dessas maneiras:

1. Use as ferramentas de linha de comando MongoDB `mongoexport` e `mongoimport`.

   `a mongoexport` exporta dados de uma instância do MongoDB para o seu sistema de arquivos, e `a mongoimport` importa os dados exportados do seu sistema de arquivos para o Oracle Database. Forneça informações de conexão ao seu banco de dados ao usar `o mongoimport`. O exemplo 2-5 ilustra isso.

2. Use uma ferramenta MongoDB, como o Compass, para importar dados para o Banco de Dados Oracle depois de conectar essa ferramenta ao banco de dados. Selecione o nome da sua coleção JSON e selecione **ADICIONAR DADOS**.

   Isso exibe uma caixa de diálogo pop-up onde você navega e importa o arquivo JSON contendo seus dados de coleta. Para maiores informações visite: MongoDB Compass.



Description of the illustration mongodb_compass.png

1. Depois de exportar dados JSON para o seu sistema de arquivos, importe-os para a Oracle Cloud Object Store e, em seguida, carregue-os de lá para uma coleção usando o procedimento PL/SQL `DBMS_CLOUD.copy_collection`. O exemplo 2-6 ilustra isso.

   Isso processa os dados em paralelo, por isso é tipicamente mais rápido do que `o mongoimport`.

2. Escreva um programa que leia documentos JSON de uma conexão com o MongoDB e escreva-os para uma conexão com o Banco de Dados Oracle.

Exemplo 2-5 Migrar dados JSON para o banco de dados Oracle usando mongoexport e mongoimport

Este exemplo exporta `vendas` de coleção do MongoDB para o file-system file `sales.json`. Em seguida, importa os dados desse arquivo para o Oracle Database como `vendas` de coleta. O usuário está conectado ao host *<host>* como esquema de banco de dados *<usuário>* com senha *<password>*.

```
mongoexport --collection=sales --out sales.json


mongoimport
'mongodb://<user>:<password>@<host>:27017/<user>?authMechanism=PLAIN&authSource=$external&ssl=true' --collection=sales --
file=sales.json
```

Nota: Use codificação por cento URI para substituir quaisquer caracteres reservados em seu URI de string de conexão — em particular, caracteres em seu nome de usuário e senha. Estes são os caracteres reservados e suas codificações com um sinal de porcentagem conforme apresentado no quadro abaixo:

!	#	$	%	&	'	(	)	*	+
%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B

,	/	:	;	=	?	@	[	]
%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D

Por exemplo, se seu nome de usuário é `RUTH` e sua senha é `@least1/2#?` , sua sequência de conexão MongoDB para servidor *<server>* será:

```
Copy'mongodb://RUTH:%40least1%2F2%23%3F@<server>:27017/ruth/ ...'
```

Dependendo das ferramentas ou drivers que você usa, você pode ser capaz de fornecer um nome de usuário e senha como parâmetros separados, em vez de como parte de uma sequência de conexão URI. Nesse caso, você provavelmente não precisará codificar nenhum caractere reservado que eles contenham.

Exemplo 2-6 carregando dados JSON em uma coleção usando DBMS_CLOUD.COPY_COLLECTION

Este exemplo carrega dados da Oracle Cloud Object Store em uma nova coleção,`newCollection`, usando o procedimento PL/SQL `DBMS_CLOUD.copy_collection`. Ele assume que os dados foram exportados do MongoDB para o seu sistema de arquivos e, em seguida, importados de lá para o local passado como o valor do parâmetro `file_uri_list`.

O valor `copy_collection` do parâmetro `FORMAT` é um objeto JSON com campos `recorddelimiter` e `type`:

1. O campo `recorddelimiter` especifica que os registros nos dados de entrada são separados por caracteres de linha nova. Um documento JSON é criado para cada registro, ou seja, para cada linha nos dados de entrada delimitados pela nova linha.
1. O campo `type` especifica que os dados de entrada JSON podem conter objetos estendidos EJSON e que estes devem ser interpretados.

```
BEGIN
  DBMS_CLOUD.copy_collection(
    collection_name => 'newCollection',
    file_uri_list   => 'https://objectstorage.../data.json',
    format          => json_object(
                         'recorddelimiter' : '''\n''',
                         'type'            : 'ejson'));
END;
/
```

Veja mais informações em:

Using the Oracle Database API for MongoDB